Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 269 (2011) 71–82
www.elsevier.com/locate/entcs

The Descriptive Complexity of the Deterministic Exponential Time Hierarchy

Cibele Matos Freire[1,3] Ana Teresa Martins[2,4]

Departamento de Computação
Universidade Federal do Ceará
Fortaleza, Brasil

Abstract

In Descriptive Complexity, we investigate the use of logics to characterize computational complexity classes. Since 1974, when Fagin proved that the class NP is captured by existential second-order logic, considered the first result in this area, other relations between logics and complexity classes have been established. Well-known results usually involve first-order logic and its extensions, and complexity classes in polynomial time or space. Some examples are that the first-order logic extended by the least fixed-point operator captures the class P and the second-order logic extended by the transitive closure operator captures the class PSPACE. In this paper, we will analyze the combined use of higher-order logics of order i, HO^i, for $i \geq 2$, extended by the least fixed-point operator, and we will prove that each level of this hierarchy captures each level of the deterministic exponential time hierarchy. As a corollary, we will prove that the hierarchy of $\mathrm{HO}^i(\mathrm{LFP})$, for $i \geq 2$, does not collapse, that is, $\mathrm{HO}^i(\mathrm{LFP}) \subset \mathrm{HO}^{i+1}(\mathrm{LFP})$.

Keywords: Descriptive Complexity, Higher-Order Logics, Deterministic Exponential Time Hierarchy, Fixed-Point Operator.

1 Introduction

In Computational Complexity, the measure of the efficiency of an algorithm is usually based on the amount of time and space consumed in its execution. The investigation about the efficiency of an algorithm allows us to define a hierarchy of classes of problems. When this hierarchy is based on time and space, we identify important classes as the class P that contains all problems which have a polynomial deterministic Turing machine that solves them, and the class NP that contains all problems which have a polynomial nondeterministic Turing machine that solves them.

[1] This research is partially supported by CNPq (Master's degree scholarship)

[2] This research is partially supported by CNPq (PQ, Universal 2008, "Casadinho" 2008) and CAPES(PROCAD)

[3] Email:cibelemf@lia.ufc.br

[4] Email:ana@lia.ufc.br

doi:10.1016/j.entcs.2011.03.006

The measures of time and space in Computational Complexity, although natural from the point of view of engineering since they reflect physical resources necessary for computing, are not intuitive from the mathematical perspective.

In 1974, Fagin showed that the complexity class NP is exactly the one where problems in it can be described by the existential fragment of second-order logic ($\exists$SO), that is, NP=$\exists$SO [1]. This result is very important since it indicates that the complexity of a problem can be characterized independently of the machine, through the expressiveness of a logical language used to specify all and only the problems of a class.

After Fagin's initial result, many others were obtained leading us to believe that all complexity classes can be precisely captured by a logical language. This area is known as Descriptive Complexity [6].

In the definition of a complexity class, decision problems play an essential role and, in the area of Descriptive Complexity, we refer to them as boolean queries. We can describe a boolean query using a formula φ and we say that a boolean query is computable if there is a Turing machine M that computes it, i.e., $\mathcal{A} \models \varphi$ iff M accepts $\mathcal{A}$, for all finite relational structures $\mathcal{A}$.

An example of an important boolean query is REACHABILITY, or simply REACH, thus defined: Let G be a graph and s and t be two vertices in G. Is there a path between s and t in G? If we want to express this query in first-order logic (FO), we will see that it is not possible due to its limited expressive power. Indeed, we know that FO=LH, i.e., FO captures the logarithmic time hierarchy and, as REACH is in P$-$LH, FO cannot express it [6].

By the fact that FO=LH, we can notice that the expressive power of FO is too weak. There are several ways to extend FO in order to get stronger logics. For instance, we can increase the order of the variables of the logic obtaining higher-order logics, we can add operators not expressible in FO as the transitive closure and the fixed-point ones, and we can allow formulas of infinite size obtaining infinitary logics.

In this paper, we are interested in investigating the combined use of the least fixed-point operator with higher-order logics. In section 2, we will present the definition of higher-order logics and some expressiveness results already established in the literature. In section 3, we will introduce the least fixed-point operator and we will show how it increases the expressive power of FO. In section 4, we will prove our main result: the characterization of the complexity classes captured by each higher-order logic extended with the least fixed-point operator. Some corollaries about the expressive power of these logics are also presented. We will end with the Conclusion section. Throughout the text, the basics of Computational Complexity are from [12].

2 Higher-Order Logics

As said in the Introduction, there are several ways to increase the expressive power of FO. In this section, we will introduce the higher-order logics, HO, whose language

extend that of FO with relation variables of any order. At the end of this section, we will see how big is their expressive power when compared to the one of FO already mentioned.

We begin with the definition of formulas in higher-order logics based on a vocabulary σ, a finite set of constant, function and relation symbols. For our purposes, we will only consider relational vocabularies, the ones which just contain relation symbols. For every $i \geq 2$, the alphabet of a higher-order logic of order i, HO^i, contains the usual logical and punctuation symbols, the symbols in σ, a countable infinite set of individual variables $x_1, x_2, x_3, \ldots$ and, for every $r \geq 1$ and $2 \leq j \leq i$, a countable infinite set of r-ary relation variables $X_1, X_2, X_3, \ldots$ of order j. Henceforth, we will often use x and y for individual variables, X and Y for relation variables and R for relation symbols, with or without indexes.

Definition 2.1 Let σ be a relational vocabulary. The set of *σ-formulas of HO^i* is inductively defined as follows:

(i) If x and y are individual variables, then $x = y$ is a σ-formula.

(ii) If R is an r-ary relation symbol in σ, $r \geq 1$, and $x_1, \ldots, x_r$ are individual variables, then $R(x_1, \ldots, x_r)$ is a σ-formula.

(iii) If X is an r-ary relation variable of order 2, $r \geq 1$, and $x_1, \ldots, x_r$ are individual variables, then $X(x_1, \ldots, x_r)$ is a σ-formula.

(iv) If X is an r-ary relation variable of order j, $r \geq 1$ and $3 \leq j \leq i$, and $Y_1, \ldots, Y_r$ are r-ary relation variables of order $j-1$, then $X(Y_1, \ldots, Y_r)$ is a σ-formula.

(v) If X and Y are relation variables of the same order and of the same arity, then $X = Y$ is a σ-formula.

(vi) If φ and ψ are σ-formulas, then $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ are σ-formulas.

(vii) If φ is a σ-formula and x is an individual variable, then $\exists x\varphi$ and $\forall x\varphi$ are σ-formulas.

(viii) If φ is a σ-formula and X is a relation variable, then $\exists X\varphi$ and $\forall X\varphi$ are σ-formulas.

Whenever the reference to σ is either clear or unimportant, we will just speak of formulas, instead of σ-formulas. The atomic formulas are the ones introduced above by items (i) to (v).

The interpretation of a formula is based on the following definitions. A relational σ-structure $\mathcal{A}$ is a tuple $(A,\ R_1^{\mathcal{A}},\ \ldots,\ R_n^{\mathcal{A}})$, where A is a nonempty finite set, the domain of $\mathcal{A}$, and $R_i^{\mathcal{A}}$, $1 \leq i \leq n$, is an r-ary relation on A that interprets each r-ary relation symbol $R_i \in \sigma$. A *valuation* v on a σ-structure $\mathcal{A}$ is a function which assigns to each individual variable x an element of A, and to each r-ary relation variable X of order j, $r \geq 1$ and $2 \leq j \leq i$, an r-ary relation of order j on A. The notion of relation of order j can be defined as: Let R be a relation of order j and arity r. If $j = 2$ then R is a relation on A^r, i.e., $R \in A_2$, where $A_2 = \mathcal{P}(A^r)$ and $\mathcal{P}(A^r)$ is the powerset of A^r. If $j = i$ then R is a relation on the set of relations of order $i-1$, i.e., $R \in A_i$, where $A_i = \mathcal{P}(A_{i-1}^r)$. Let v_0 and v_1 be two valuations on

a σ-structure $\mathcal{A}$, and let V be a variable of whichever kind, we say that v_0 and v_1 are V-equivalent if they coincide in every variable of whichever kind, but on V.

Definition 2.2 Let $\mathcal{A}$ be a σ-structure and let v be a valuation on $\mathcal{A}$. The notion of *satisfaction on HO^i* is inductively defined as follows:

(i) $\mathcal{A}, v \models (x = y)$:iff $v(x) = v(y)$, where x and y are individual variables.

(ii) $\mathcal{A}, v \models R(x_1, \ldots, x_r)$:iff $(v(x_1), \ldots, v(x_r)) \in R^{\mathcal{A}}$, where R is an r-ary relation symbol in σ, $r \geq 1$, and $x_1, \ldots, x_r$ are individual variables.

(iii) $\mathcal{A}, v \models X(x_1, \ldots, x_r)$:iff $(v(x_1), \ldots, v(x_r)) \in v(X)$, where X is an r-ary relation variable of order 2, $r \geq 1$, and $x_1, \ldots, x_r$ are individual variables.

(iv) $\mathcal{A}, v \models X(Y_1, \ldots, Y_r)$:iff $(v(Y_1), \ldots, v(Y_r)) \in v(X)$, where X is an r-ary relation variable of order j, $r \geq 1$ and $3 \leq j \leq i$, and $Y_1, \ldots, Y_r$ are r-ary relation variables of order $j - 1$.

(v) $\mathcal{A}, v \models (X = Y)$:iff $v(X) = v(Y)$, where X and Y are relation variables of order j, for some $2 \leq j \leq i$, and of arity r, for some $r \geq 1$.

(vi) $\mathcal{A}, v \models (\neg\varphi)$:iff $\mathcal{A}, v \not\models \varphi$, where φ is a formula.

(vii) $\mathcal{A}, v \models (\varphi \wedge \psi)$:iff $\mathcal{A}, v \models \varphi$ and $\mathcal{A}, v \models \psi$, where φ and ψ are formulas.

(viii) $\mathcal{A}, v \models (\varphi \vee \psi)$:iff $\mathcal{A}, v \models \varphi$ or $\mathcal{A}, v \models \psi$, where φ and ψ are formulas.

(ix) $\mathcal{A}, v \models \exists x\varphi$:iff there is a valuation v' that is x-equivalent to v and $\mathcal{A}, v' \models \varphi$, where x is an individual variable and φ is a formula.

(x) $\mathcal{A}, v \models \forall x\varphi$:iff for every valuation v' that is x-equivalent to v we have that $\mathcal{A}, v' \models \varphi$, where x is an individual variable and φ is a formula.

(xi) $\mathcal{A}, v \models \exists X\varphi$:iff there is a valuation v' that is X-equivalent to v and $\mathcal{A}, v' \models \varphi$, where X is a relation variable and φ is a formula.

(xii) $\mathcal{A}, v \models \forall X\varphi$:iff for every valuation v' that is X-equivalent to v we have that $\mathcal{A}, v' \models \varphi$, where X is a relation variable and φ is a formula.

As for FO, we can define the class Σ^i_j, $i, j \geq 1$, of formulas $\varphi \in \mathrm{HO}^{i+1}$ of the form $\exists X_{11} \ldots \exists X_{1s_1} \forall X_{21} \ldots \forall X_{2s_2} \ldots QX_{j1} \ldots QX_{js_j} \psi$, where $\psi \in \mathrm{HO}^i$, and Q is either $\exists$ or $\forall$ depending on whether j is odd or even, respectively. That is, Σ^i_j is the class of HO^{i+1} formulas with $j - 1$ alternations of quantifiers block of variables of order $i + 1$, starting with an existential quantifier. The class Π^i_j is dually defined. We say that a formula φ is in *Generalized Skolem Normal Form*, GSNF, if it belongs to either Σ^i_j or Π^i_j, for some $i, j \geq 1$. The following well known lemma is true.

Lemma 2.3 *[3] For all $i \geq 2$, and for every formula $\varphi \in \mathrm{HO}^i$, there is a formula $\hat{\varphi} \in \mathrm{HO}^i$ which is in GSNF and equivalent to φ.*

Some results concerning the descriptive complexity of HO over finite structures were already established in [14], [4], [8], [10] and [9]. The following one gives a well-known correspondence between SO and the polynomial-time hierarchy.

Theorem 2.4 *[14]* SO = PH, *where* PH *is the polynomial-time hierarchy defined*

as: $\mathrm{PH} = \bigcup_{i\geq 0} \Sigma_i\mathrm{P}$, $\Sigma_0\mathrm{P} = \mathrm{P}$ *and* $\Sigma_{i+1}\mathrm{P} = \mathrm{NP}^{\Sigma_i P}$, *for every* i.

The following theorems give results for all levels of HO. The first theorem shows that each level of HO is more expressive than the level before, and the second shows that the same is true when we consider the existential and universal fragments.

Theorem 2.5 *[4] For all* $i \geq 2$, $\mathrm{HO}^i \subset \mathrm{HO}^{i+1}$.

Theorem 2.6 *[8] For all* $i \geq 2$,

(i) $\exists\mathrm{HO}^i \subset \exists\mathrm{HO}^{i+1}$.

(ii) $\forall\mathrm{HO}^i \subset \forall\mathrm{HO}^{i+1}$.

In [9] and [10] a characterization of HO is presented. Some criticisms about this characterization are presented in [3]. We follow the late characterization. Before presenting this result, we need the definition of the function *exp* and of the nondeterministic exponential time hierarchy, NEXPH.

Definition 2.7 Let $f(n)$ be a function on natural numbers. We define $exp(0, f(n)) = f(n)$ and, for $i \geq 1$, $exp(i, f(n)) = 2^{exp(i-1, f(n))}$.

Definition 2.8 For $i \geq 0$ and $j \geq 1$, we define the *Nondeterministic Exponential Time Hierarchy* as:

(i) $\mathrm{NEXPH}_i^0 = \bigcup_{c\in\mathbb{N}} \mathrm{NTIME}(exp(i, n^c))$

(ii) $\mathrm{NEXPH}_i^j = \mathrm{NEXPH}_i^{0^{\Sigma_{j-1}P}}$, where $\Sigma_{j-1}\mathrm{P}$ is defined as in the polynomial-time hierarchy.

The following theorem establishes that each fragment of HO in the prenex normal form, i.e. Σ_j^i and Π_j^i, captures a level in the Nondeterministic Exponential Time Hierarchy. The Theorem 2.4 is a straightforward corollary of Theorem 2.9.

Theorem 2.9 *[2] [3] For* $i, j \geq 1$*:*

(i) $\Sigma_j^i = \mathrm{NEXPH}_{i-1}^{j-1}$.

(ii) $\Pi_j^i = \mathrm{coNEXPH}_{i-1}^{j-1}$.

By Lemma 2.3 and Theorem 2.9, we get as a corollary the characterization of the higher-order logics.

Corollary 2.10 *[2] [3] For every* $i \geq 2$, $\mathrm{HO}^i = \bigcup_{j\geq 0} (\mathrm{NEXPH}_{i-2}^j \cup \mathrm{coNEXPH}_{i-2}^j)$.

3 The Least Fixed-Point Logic

In the first section, we said that FO is not powerful enough to express the boolean query REACH, although it can express the following one: "Let G be a finite graph with two vertices, s and t, in it. Is there a path from s to t in G with length n?" How can we explain this fact? From a computational point of view, we may say

that FO can express loops with a precise number of iterations (*for-loops*), but not the ones where you do not know in advance the number of steps needed to finish them (*while-loops*). From a mathematical point of view, we may say that many inductively defined relations are not FO expressible.

Another possibility of extending FO and increasing its expressive power is with an operator. In this section, we will add the least fixed-point operator to FO which allows one to define new relations by induction. This is the Least Fixed-Point logic, FO(LFP). We will close this section by also analysing its expressive power.

Let A be a finite set, $\mathcal{P}(A)$ be its powerset, $F : \mathcal{P}(A) \to \mathcal{P}(A)$ be an operator on A. We say that F is monotone if $X \subseteq Y$ then $F(X) \subseteq F(Y)$, for $X, Y \subseteq A$. A set $X \subseteq A$ is a fixed-point of F if $F(X) = X$. A set $X \subseteq A$ is the least fixed-point of F, $\mathbf{lfp}(F)$, if it is a fixed-point and, for every other fixed-point Y of F, we have $X \subseteq Y$. F gives rise to a sequence of sets $\emptyset, F(\emptyset), F(F(\emptyset)), \ldots$. Let us call its members by $F_0, F_1, F_2, \ldots$, i.e., $F_0 = \emptyset$ and $F_{n+1} = F(F_n)$. F_n will denote the n-th stage of F. Suppose that there is an n_0 such that $F(F_{n_0}) = F_{n_0}$. By the definition of this sequence, $F_m = F_{n_0}$, for all $m \geq n_0$ and, by the definition of fixed-point, F_{n_0} is a fixed-point of F. We will denote F_{n_0} by F_∞. We can not garantee that the fixed-point F_∞ exists, however, in some cases, we can say that it exists and is the least fixed-point of F.

Lemma 3.1 *[7][15] If F is monotone, then F_∞ is the least fixed-point of F.*

Besides the fact that F_∞ is the least fixed-point when F is monotone, its clear that the fixed-point is achieved at most in the n-th stage, where $|A| = n$.

Let σ be a relation vocabulary, R a k-ary relation symbol not in σ and $\varphi(R, x_1, \ldots, x_k)$ a $\sigma \cup \{R\}$-formula. For each finite σ-structure $\mathcal{A}$, $\varphi(R, x_1, \ldots, x_k)$ induces the operator $F^\varphi : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ defined as

$$F^\varphi(X) = \{(a_1, \ldots, a_k) \mid \mathcal{A} \models \varphi(X/R, a_1, \ldots, a_k)\},$$

where $\varphi(X/R, a_1, \ldots, a_k)$ means that R is interpreted as X in φ.

FO(LFP) is defined as an extension of FO with formulas for computing the least fixed-point of operators F^φ. By Lemma 3.1, if F^φ is monotone then $\mathbf{lfp}(F^\varphi) = F^\varphi_\infty$ and the least fixed-point is achieved no later than the n^k-th stage of F^φ. In order to assure this property to F^φ, we have to impose some syntactic restriction to φ. Let φ be a formula that may contain an occurrence of a relation R. We say that an occurence of R is positive in φ if it is under the scope of an even number of negations. Otherwise, it is negative. A formula φ is positive in R if all occurrences of R are positive, or there are none at all.

Lemma 3.2 *Let σ be a relational vocabulary, R a k-ary relation symbol not in σ and $\varphi(R, x_1, \ldots, x_k)$ a $\sigma \cup \{R\}$-formula. If φ is positive in R, then F^φ is monotone.*

The proof of this lemma is by induction on formulas. The intuitive idea of this lemma is as follows: We want to prove that if $X \subseteq Y \subseteq A^k$ then $F^\varphi(X) \subseteq F^\varphi(Y)$. However, for it to be false there would be a tuple $(a_1, \ldots, a_k)$ in $F^\varphi(X)$ that was not in $F^\varphi(Y)$. However, this is not possible. Since $X \subseteq Y$, the only possibility for

$(a_1, \ldots, a_k) \notin F^{\varphi}(Y)$ would be if an occurrence of R were negative in φ, which is not the case.

Definition 3.3 Let σ be a relational vocabulary, R a k-ary relation symbol not in σ, $\varphi(R, x_1, \ldots, x_k)$ a $\sigma \cup \{R\}$-formula and $\mathcal{A}$ a finite σ-structure. The *language of FO(LFP)* extends that of FO with the following formation rule:

- if $\varphi(R, x_1, \ldots, x_k)$ is positive in R and $(t_1, \ldots, t_k)$ is a tuple of terms, then $[\mathbf{lfp}_{R,x_1,\ldots,x_k}\varphi(R, x_1, \ldots, x_k)](t_1, \ldots, t_k)$ is a formula whose free variables are those of $(t_1, \ldots, t_k)$.

The *satisfiability relation of FO(LFP)* extends that of FO with the following definition:

- $\mathcal{A} \models [\mathbf{lfp}_{R,x_1,\ldots,x_k}\varphi(R, x_1, \ldots, x_k)](a_1, \ldots, a_k)$ iff $(a_1, \ldots, a_k) \in \mathbf{lfp}(F^{\varphi})$.

As an example, let $\sigma = \{E\}$, E and R binary relations, $\mathcal{A}$ a finite σ-structure, and $\varphi_t(R, x_1, x_2) = E(x_1, x_2) \vee \exists z(E(x_1, z) \wedge R(z, x_2))$. Note that φ_t is positive in R. Now consider $[\mathbf{lfp}_{R,x_1,x_2}\varphi_t(R, x_1, x_2)]$. What does this formula define? By the above definition, the answer is $\mathbf{lfp}(F^{\varphi_t})$. But what is F^{φ_t} in this case? For a set $X \subseteq A^2$, $F^{\varphi_t}(X) = E \cup (E \circ X)$, where $E \circ X = \{(a, b) \mid (a, c) \in R, (c, b) \in R$, for some $c \in A\}$. Hence, $[\mathbf{lfp}_{R,x_1,x_2}\varphi_t(R, x_1, x_2)]$ defines the transitive closure of E.

Now, we can define the query REACH in FO(LFP) as follows:

$$\text{REACH} = [\mathbf{lfp}_{R,x_1,x_2}\varphi_t(R, x_1, x_2)](s, t).$$

As mentioned in the Introduction, FO = LH, the logarithmic-time hierarchy. We saw above that the addition of the least fixed-point operator to FO really increases its expressivity. The next theorem measures how much is this gain.

Theorem 3.4 *[5][16][11] Over finite and ordered structures,* FO(LFP) *captures* P*, the class of problems decidable in determinstc polynomial time, that is,* FO(LFP) = P.

The proof of this theorem is based on the fact that FO=LH and REACH_a is complete for P via first-order reductions. REACH_a is a variation of REACH where the path between s and t must be alternating. We can express REACH_a in FO(LFP) as follows: $\text{REACH}_\text{a} = [\mathbf{lfp}_{R,x_1,x_2}\varphi_a(R, x_1, x_2)](s, t)$, where $\varphi_a(R, x, y) \equiv (x = y) \vee (\exists z(E(x, z) \wedge R(z, y)) \wedge (A(x) \rightarrow \forall z(E(x, z) \rightarrow R(z, y))))$ (see [6] for further details). In order to prove that FO(LFP) $\subseteq$ P, we must construct a polynomial time Turing machine M that computes the query defined by a FO(LFP) sentence φ. The proof is by induction on the structure of φ. The difficult case is when φ is of the form $[\mathbf{lfp}_{R,x_1,\ldots,x_k}\varphi(R, x_1, \ldots, x_k)]$. In this case, the machine will calculate the new relation defined by the least fixed point operator. In each stage of F^{φ}, the machine will verify if φ is satisfied. As the fixed-point is achieved no later than the n^k-th stage, where n is the size of the structure and k the arity of the new relation, and the first-order sentence φ can be evaluated in logarithmic time, the new relation is computed in polynomial time, so M is in P. For the other direction, since REACH_a is complete for P, all problems in this class can be reduced to it. Moreover, since

REACH_a is definable in FO(LFP), it is easy to see that all problems in P are definable in FO(LFP) too.

4 Higher-Order Logics with the Least Fixed-Point Operator

As mentioned in the previous sections, one can increase the expressive power of logics in several ways. Here, we will investigate if the addition of the least fixed-point operator to higher-order logics increases their expressive power and we will identify which are the complexity classes captured by such logics. This is theorem 4.4 and it is the main result of this paper.

There are other fixed-points as the inflationary (IFP) and partial (PFP) ones. In [13], the authors added inflationary and partial fixed-points to higher-order logics and they showed that, for every order, it is sufficient to increase the order of the given logic by one to capture inflationary fixed-points, and by two to capture partial fixed-points. We get similar results as straightforward corollaries of Theorem 4.4, although a similar result to Theorem 4.4 is not proved in [13].

Definition 4.1 Let σ be a relational vocabulary, R a k-ary relation variable of order $i+1$, $\varphi(R, X_1, \ldots, X_k)$ a formula of *HO^i(LFP)*, where $X_1, \ldots, X_k$ are k-ary relation variables of order i, and $\mathcal{A}$ a finite σ-structure. The *language of HO^i(LFP)* extends that of HO^i with the following formation rule:

- if $\varphi(R, X_1, \ldots, X_k)$ is positive in R and $(V_1, \ldots, V_k)$ is a tuple of relations of order i, then $[\mathbf{lfp}_{R,X_1,\ldots,X_k}\varphi(R, X_1, \ldots, X_k)](V_1, \ldots, V_k)$ is a formula whose free variables are those of $(V_1, \ldots, V_k)$.

The *satisfiability relation of HO^i(LFP)* extends that of HO^i with the following definition:

- $\mathcal{A} \models [\mathbf{lfp}_{R,X_1,\ldots,X_k}\varphi(R, X_1, \ldots, X_k)](R_1, \ldots, R_k)$ iff $(R_1, \ldots, R_k) \in \mathbf{lfp}(F^{\varphi})$.

The following theorem shows which complexity class the first level of HO^i(LFP) captures. Although this fact is cited in the literature, we did not find any proof or reference for this proof.

Theorem 4.2 *Over finite and ordered structures,* SO(LFP) *captures* EXP*, the class of problems decidable in exponential time, that is,* SO(LFP) = EXP*.*

In attempt to prove the theorem above, we analyze a more general result. Instead of investigating only the first level of the hierarchy of higher-order logics with the least fixed-point, we obtained a proof that deals with all the hierarchy. In fact, we prove that each level of this hierarchy is captured by a level of the deterministic exponential time hierarchy, as we will see below.

Definition 4.3 Let i-EXP = TIME$(exp(i, n^k))$, for all k. The *Deterministic Exponential Time Hierarchy* is defined as: $\text{EXPH} = \bigcup_{i \geq 1} i\text{-EXP}$.

Theorem 4.4 *For all $i \geq 2$,* $\mathrm{HO}^i(\mathrm{LFP})$ *captures* $(i-1)$-EXP *over finite and ordered structures.*

Proof.

(a) $\mathrm{HO}^i(\mathrm{LFP}) \subseteq (i-1)$-EXP.

By induction on formulas φ. Every atomic formula of $\mathrm{HO}^i(\mathrm{LFP})$ is a formula without the operator **lfp**, thus it is a formula of HO^i. By Corollary 2.10 and by the fact that $\bigcup_{j\geq 0}(\mathrm{NEXPH}^j_{i-2} \cup \mathrm{coNEXH}^j_{i-2}) \subseteq (i-1)$-EXP [12], we can say that atomic formulas can be evaluated with an $(i-1)$-EXP machine.

$\varphi = \neg\psi$. By inductive hypothesis, there is a machine $M_\psi \in (i-1)$-EXP that evaluates ψ. The machine M_φ uses M_ψ to verify if $\mathcal{A}$ satisfies ψ. If M_ψ accepts, then M_φ rejects and vice-versa.

$\varphi = \psi_1 \wedge \psi_2$. Let M_{ψ_1} and M_{ψ_2} be the machines that evaluate ψ_1 and ψ_2. Therefore, M_φ uses these machines and, if both accept, M_φ accepts. Otherwise, it rejects.

$\varphi = \psi_1 \vee \psi_2$. Similar to the previous case.

$\varphi = \exists x\psi$. By inductive hypothesis, there is a machine $M_\psi \in (i-1)$-EXP that evaluates ψ. Note that ψ has one free variable x. Then, the machine M_φ writes on a work tape the binary encoding of $j = 0, \ldots, n-1$ and, using M_ψ, it verifies if $\mathcal{A} \models \psi(j/x)$. If M_ψ accepts, for some j, then M_φ accepts. Otherwise, M_φ rejects. As the size of domain is n, this will be done at most n times and, therefore, $M_\varphi \in (i-1)$-EXP.

$\varphi = \forall x\psi$. Similar to the previous case.

$\varphi = \exists X\psi$. X is a relational variable of order j, $2 \leq j \leq i$, and arity k. By induction hypothesis, there is a machine $M_\psi \in (i-1)$-EXP. Note that ψ has a free variable X. M_φ uses a work tape to code relations of order j and arity k. This tape uses $exp(i-2, n^k)$ cells and it can encode $exp(i-1, n^k)$ different relations. To each relation R described in the tape, M_ψ is used to verify if $\mathcal{A} \models \psi(R)$. If M_ψ accepts for some R, then M_φ accepts. Otherwise, M_φ rejects. As at most $exp(i-1, n^k)$ calls to M_ψ will be done, then M_φ will execute in time $exp(i-1, n^k).exp(i-1, O(n^c)) = exp(i-1, O(n^c))$. Thus, $M_\varphi \in (i-1)$-EXP.

$\varphi = \forall X\psi$. Similar to the previous case.

$\varphi = [\mathbf{lfp}_{R,X_1,\ldots,X_k}\psi(R, X_1, \ldots, X_k)](R_1, \ldots, R_k)$. By inductive hypothesis, there is $M_\psi \in (i-1)$-EXP. The machine M_φ uses two work tapes to write a encoding of relations of order $i+1$ that will be assigned to the relational variable R. As these relations are of order $i+1$, their maximum size is the number of different tuples that can be formed with relations of order i. If the relation is of order 2 and arity r, then the maximum size is given by $|A^r| = exp(0, n^r)$. If the order is 3, the maximum size is $|A^r_2| = exp(1, n^r)$ and, in the general case, if order is i, then the maximum size is $|A^r_{i-1}| = exp(i-2, n^r)$. Thus, we need $exp(i-1, n^k)$ cells to encode a relation of order $i+1$. In the first step, $R = \emptyset$ is the relation encoded on tape 1. Using the machine M_ψ, M_φ computes the following relation: $R' = \{(V_1, \ldots, V_k) \mid \mathcal{A} \models \psi(R, V_1, \ldots, V_k)\}$. If $R = R'$ then the fixed-point is achieved and it is sufficient to test whether $(X_1, \ldots, X_k) \in R$. Otherwise, it copies

the contents of tape 1 into tape 2, it erases the contents of tape 1, and it computes R' again. As ψ is positive in R, with an argument like in Lemma 3.2 and by the Lemma 3.1, we know that the fixed-point of F^{ψ} will be achieved in at most $exp(i-1, n^k)$ steps. So, M_{ψ} will be called at most $exp(i-1, n^k)$ times and M_{φ} will execute in time $exp(i-1, n^k).exp(i-1, O(n^c)) = exp(i-1, O(n^c))$. Therefore, M_{φ} $\in (i-1)$-EXP.

(b) $(i-1)$-EXP $\subseteq$ HO^i(LFP).
Let $M \in (i-1)$-EXP be a Turing machine that executes in time $exp(i-1, n^k)$, for some k. As a consequence, we can conclude that it uses at most $exp(i-1, n^k)$ cells in its work tapes. We want to define a formula φ such that M accepts $\mathcal{A}$ iff $\mathcal{A} \models \varphi$. Our formula describes the computation of the machine and, as the computation of a machine is described as a sequence of configurations, we will use a relation C to encode such sequence of configurations. A machine configuration is defined by the current state, the current position of the head of each tape and the current content of each work tape. We can encode this data with a relation C of order $i+1$ and arity 3, as we can see next. We use the first component of C as a time stamp to indicate which is the position of the configuration in the sequence, i.e., in which moment of the computation the configuration described by that tuple was reached. As the machine executes in time $exp(i-1, n^k)$, we can encode this with relations of order i. The second component indicates what kind of data is encoded in the third component of C. The data can be the current state, the current head position of tape m or the current content of tape m. As the number of states and tapes of the machine is finite, this can be encoded using individual variables, easily with a relation of order i. The third component encodes the data itself. In this case, when we indicate the position of the head or the contents of work tape, we need to encode an exponential number of positions, because, as we previously mentioned, the machine uses at most $exp(i-1, n^k)$ cells for each tape. Therefore, we can use relations of order i again. Now, we can see that each tuple of C encodes a part of the configuration of machine M and, in fact, if we want to know what is the configutation of M in time t, we have to look for all tuples whose the first component is "t". As the tuples of the relation C are formed by relations of order i, C is a relation of order $i+1$. As we cannot quantify relations of order $i+1$, we need to use the least fixed-point operator to define the relation C. To do this, we define a formula ψ such that $C = [\mathbf{lfp}_{R,X_1,X_2,X_3}\psi(R, X_1, X_2, X_3)]$. The formula ψ is built based on the instructions of the machine in a way that in the t-th step of the least fixed-point operator the tuples added to the relation are the ones that encode the configuration of M in the instant t. In the end, it is enough to verify if there is a tuple in C in which the first component indicates the time $exp(i-1, n^k)$, the second component indicates that the data is the current state, and the third component indicates the accepting state. □

Theorem 4.2 is a straightforward corollary of Theorem 4.4. The last one follows the same argument of well known results as, for example, FO=LH, FO(LFP)=P and SO=PH, although they differ in technical details. The interesting fact is the close

relation between the addition of the least fixed-point operator and the exponential gain in time. For example, FO captures the logarithmic hierarchy and, when we add the operator to obtain FO(LFP), the new logic captures the class P, i.e., an exponential gain in time with respect to LH. This gain also occurs when we add the least fixed-point operator to higher-order logics. Below, see some reasonable corollaries that follow from our main result.

Corollary 4.5 $\mathrm{HO}^i(\mathrm{LFP}) \subset \mathrm{HO}^{i+1}(\mathrm{LFP})$, *for all* $i \geq 2$.

Corollary 4.6 $\mathrm{HO}^i(\mathrm{LFP}) \subseteq \mathrm{HO}^{i+1}$, *for all* $i \geq 2$.

Corollary 4.7 $\bigcup_{i\geq 2} \mathrm{HO}^i(\mathrm{LFP}) = \bigcup_{i\geq 2} \mathrm{HO}^i$.

In the Corollary 4.5 we see that the hierarchy of higher-order logics with the least fixed-point does not collapse. In fact, the proof of this corollary is a consequence of Theorem 4.4 and of the Time Hierarchy Theorem [12] which has as a corollary the fact that $(i-1)$-EXP $\subset$ i-EXP. Corollary 4.6 is similar to the theorem proved in [13] to $\mathrm{HO}^i(\mathrm{IFP})$, and Corollary 4.7 states that there is no gain in expressivity when we consider the union of all levels of the hierarchy.

5 Conclusions

In Descriptive Complexity, we are interested in characterizing complexity classes using logics. Most results were obtained involving FO, and its extensions, and polynomial complexity classes. In this paper, we investigated the addition of the least fixed-point operator to higher-order logics.

Our central result is that $\mathrm{HO}^i(\mathrm{LFP})$ captures the $(i-1)$-EXP complexity class, for $i \geq 2$. Hence, we characterized all levels of the Deterministic Exponential Time Hierarchy. From this, we obtained interesting corollaries. The first one states that it is sufficient to increase the order of the given higher-order logic by one to capture the least fixed-point operator, that is, $\mathrm{HO}^i(\mathrm{LFP}) \subseteq \mathrm{HO}^{i+1}$, for all $i \geq 2$. We also proved that the $\mathrm{HO}^i(\mathrm{LFP})$ hierarchy does not collapse since each level is more expressive than the previous one, that is, $\mathrm{HO}^i(\mathrm{LFP}) \subset \mathrm{HO}^{i+1}(\mathrm{LFP})$, for all $i \geq 2$. Finally, despite of the gain of expressivity in each level of the $\mathrm{HO}^i(\mathrm{LFP})$ hierarchy, there is no gain in expressivity when we consider the union of all levels, that is, $\bigcup_{i\geq 2} \mathrm{HO}^i(\mathrm{LFP}) = \bigcup_{i\geq 2} \mathrm{HO}^i$.

The idea of adding some fixed-point operators to higher-order logics was first formulated by [13], but for inflationary and partial fixed-points. They get similar results to our corollaries, although not for theorem 4.4.

References

[1] Fagin, R., *Generalized first-order spectra and polynomial-time recognizable sets*, , **7**, AMS Bookstore, 1974 pp. 43–73.

[2] Hella, L. and J. M. Turull Torres, *Expressibility of higher order logics*, Eletronic Notes in Theoretical Computer Science **84** (2003), pp. 129–140.

[3] Hella, L. and J. M. Turull Torres, *Computing queries with higher order logics*, Theoretical Computer Science (2006), pp. 197–214.

[4] Hull, R. and J. Su, *On the expressive power of database queries with intermediate types*, Journal of Computer and System Sciences **43** (1991), pp. 219–267.

[5] Immerman, N., *Relational queries computable in polynomial time (extended abstract)*, in: *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing* (1982), pp. 147–152.

[6] Immerman, N., "Descriptive Complexity," Springer, 1999.

[7] Knaster, B., *Un théoréme sur les fonctions d'ensembles.*, Annales de la Société Polonaise de Mathématique. **6** (1928), pp. 133–134.

[8] Kuper, G. M. and M. Y. Vardi, *On the complexity of queries in the logical data model*, Lecture Notes in Computer Science (1988), pp. 267–280.

[9] Leivant, D., *Characterization of complexity classes in higher-order logic*, in: *Proceedings of the 2nd. Annual Conference Structure in Complexity Theory*, (1987), pp. 203–217.

[10] Leivant, D., *Descriptive characterizations of computational complexity*, Journal of Computer and System Sciences **39** (1989), pp. 51–83.

[11] Livchak, A. B., *Languages for polynomial-time queries*, Computer-Based Modeling and Optimization of Heat-Power and Electrochemical (1982), p. 41.

[12] Papadimitriou, C. H., "Computational Complexity," Addison Wesley Logman, 1994.

[13] Schewe, K.-D. and J. M. T. Torres, *Fixed-point quantifiers in higher order logics*, in: *Proceeding of the 2006 conference on Information Modelling and Knowledge Bases XVII* (2006), pp. 237–244.

[14] Stockmeyer, L. J., *The polynomial-time hierarchy*, Theoretical Computer Science **3** (1977), pp. 1–22.

[15] Tarski, A., *A lattice-theoretical fixpoint theorem and its applications.*, Pacific Journal oh Mathematics **5** (1955), pp. 285–309.

[16] Vardi, M. Y., *The complexity of relational query languages (extended abstract)*, in: *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing* (1982), pp. 137–146.