Expressibility of Higher-Order Logics on Relational Databases: Proper Hierarchies

A dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Systems at Massey University, Wellington, New Zealand.

Flavio Antonio Ferrarotti

2008

Supervisor:
Associate Prof. Dr. José María Turull Torres

Co-Supervisor:
Prof. Dr. Klaus-Dieter Schewe

Internal Examiner:
Prof. Dr. Sven Hartmann

New Zealand Examiner:
Prof. Dr. Robert Goldblatt

Overseas Examiner:
Prof. Dr. Lauri Hella

Date of Examination:
12.06.2008

Dedicated to the memory of my grandfather, Reinaldo

Abstract

We investigate the expressive power of different fragments of higher-order logics over finite relational structures (or equivalently, relational databases) with special emphasis in higher-order logics of order greater than or equal three. Our main results concern the study of the effect on the expressive power of higher-order logics, of simultaneously bounding the arity of the higher-order variables and the alternation of quantifiers. Let $AA^i(r, m)$ be the class of $(i + 1)$-th order logic formulae where all quantifiers are grouped together at the beginning of the formulae, forming m alternating blocks of consecutive existential and universal quantifiers, and such that the maximal-arity (a generalization of the concept of arity, not just the maximal of the arities of the quantified variables) of the higher-order variables is bounded by r. Note that, the order of the quantifiers in the prefix may be mixed. We show that, for every $i \geq 1$, the resulting AA^i hierarchy of formulae of $(i + 1)$-th order logic is proper. This extends a result by Makowsky and Pnueli who proved that the same hierarchy in second-order logic is proper. In both cases the strategy used to prove the results consists in considering the set $AUTOSAT(F)$ of formulae in a given logic F which, represented as finite structures, satisfy themselves. We then use a similar strategy to prove that the classes of $\Sigma^i_m \cup \Pi^i_m$ formulae in which the higher-order variables of all orders up to $i+1$ have maximal-arity at most r, also induce a proper hierarchy in each higher-order logic of order $i \geq 3$. It is not known whether the correspondent hierarchy in second-order logic is proper. Using the concept of finite model truth definitions introduced by M. Mostowski, we give a sufficient condition for that to be the case. We also study the complexity of the set $AUTOSAT(F)$ and show that when F is one of the prenex fragments Σ^1_m of second-order logic, it follows that $AUTOSAT(F)$ becomes a complete problem for the corresponding prenex fragment Σ^2_m of third-order logic. Finally, aiming to provide the background for a future line of research in higher-order logics, we take a closer look to the restricted second-order logic SO^ω introduced by Dawar. We further investigate its connection with the concept of relational complexity studied by Abiteboul, Vardi and Vianu. Dawar showed that the existential fragment of SO^ω is equivalent to the nondeterministic inflationary fixed-point logic NFP. Since NFP captures *relational* NP, it follows that the existential fragment of SO^ω captures *relational* NP. We give a direct proof, in the style of the proof of Fagin's theorem, of this fact. We then define formally the concept of relational machine with *relational oracle* and prove the exact correspondence between the prenex fragments of SO^ω and the levels of the *relational* polynomial-time hierarchy. This allows us to stablish a direct connection between the *relational* polynomial hierarchy and SO^ω without using the Abiteboul and Vianu normal form for relational machines.

Acknowledgements

It is very difficult to express how much I owe to José María Turull-Torres in just a few lines. I have been truly fortunate to have him as supervisor. He taught me not only most of what I know of finite model theory, but also to put first things first, to be precise, to call things by their name, to be rigorous and to be honest. Frequently, he would come over to my office and invite me for a coffee to not only discuss my work, but also to chat about life. I always found after these discussions with him the strength and motivation needed to continue with this project. He helped me immensely in every aspect of my work. More importantly, he was always very careful to give me just the right amount of guidance that would allow me to evolve as a researcher by finding my own path and research questions. Apart from being such an exceptional supervisor, he also gave me the invaluable gift of his friendship for which I am eternally grateful.

I am also deeply grateful to my co-supervisor, Klaus-Dieter Schewe. He made it possible for me to pursue my academic dream in New Zealand. He is a very generous man. He supported our work in every possible way, yet he never asked anything in return. He has been a guide to me from both an academic and a personal perspective.

I want to express my profound gratitude to Lauri Hella, Robert Goldblatt and Sven Hartmann, for serving as examiners, for a careful reading of the dissertation, and for their valuable comments.

I am very grateful to Leszek Kolodziejczyk for his important and detailed observations after a careful reading of a preliminary version of the work in Chapters 4 and 5. His remarks were very helpful. I have also learnt a lot from his excellent thesis. I thank Michal Krynicki for first drawing my attention to it.

I would also like to thank the anonymous referees for the Annals of Mathematics and Artificial Intelligence - their valuable observations have greatly helped me to shape the final version of fundamental parts of this dissertation.

I would like to thank all members of the Information Science Research Centre - it has been an extraordinary group to work with. A special thanks is due to the head of the department of Information Systems, Chris Freyberg, and to the secretaries, Merle Richardson and Julie Lyons. To Chris because of his constant support, and to Merle and July because they always had a solution for my problems.

My thanks also goes to the members of the doctoral subcommittee of the department of computer sciences at the University of Buenos Aires, especially to Alejandro Ríos. After careful evaluation of my application for admission as PhD student, they recommended me to take a paper on First Order Logic and Computability Theory taught by Roberto Cignoli, and a paper on Theory of Languages taught by Juan Miguel Santos. I benefited greatly from taking those papers and it was then that I developed my interest in the fascinating world of mathematical logic. I also thank Alejandro Vaisman who, without knowing me, accepted to act as my PhD advisor during the time I spent in the University of Buenos Aires.

My Friends in Argentina have always been with me through these years. They never complained for the long periods of silence on my part. Thanks are due to them for receiving me each time I went back and making me feel as if I never left. A very special thanks goes to Jorge Peri for putting me in contact with José María when I was starting my PhD. Jorge was the most extraordinary professor that I had during all my years as an undergraduate student and I will be always in debt to him for that. Also a special thanks goes to Stella Perillo and Fernando Lorge. When I decided to start my PhD, I had the good fortune of being working with them in my paid job. They strongly supported me in every imaginable way during that time. I also thank Gustavo Croch who has always been there when I needed a hand, and to Fernando Bordignon whom I admire for his enthusiastic and honest attitude toward research, which has been an important source of motivation for me.

I thank my dear friends, Peter and Mercedes Baxter and Raquel Direnzo, for all their support. They always made me and my partner feel at home in New Zealand and adopted us as part of their family here.

I thank my mother, Antonia, who encouraged me to pursue my dreams.

Last but not least, I am deeply in debt with my partner Lorena. I could not have done this work without her love, constant encouragement and support during all these years. She and our newborn son Luca are the two best things that ever happened to me.

Contents

Chapter 1

Introduction

Since the irruption of the relational database model proposed by Codd in the seventies [Cod70], the interaction between database theory and finite model theory has been a very fruitful one. Logics over finite structures have become the standard starting point for developing the theory of database query languages, and finite model theory techniques are used for proving results about their expressiveness and complexity [AHV94, EF99, Imm99, Lib04].

Due to the well known limitations of first-order logic (or equivalently, relational calculus) as a query language, this logic loses the dominating role it has always played in classical model theory when it comes to the study of finite relational structures or databases. Instead, various extensions of first-order logic have been explored as a means to build logics more expressive in finite structures. Among the extensions of first-order which have received more attention we can mention fixed-point logics, logics with counting quantifiers, infinitary logics, and second-order logic.

A fundamental underlying question in this regard is: which kind of syntactic restrictions have impact on the expressive power of such logics over finite structures?, or equivalently: when can we really express more queries over relational databases? The answer to this question is in many cases related to important open problems in complexity theory. An example in second-order logic of this fact, can be found in the well known Fagin-Stockmeyer characterization [Sto76] of the polynomial time-hierarchy. Asking whether increasing the number of alternating blocks of quantifiers in the prenex fragments of second-order logic allows us to express more queries, is equivalent to asking whether the polynomial time-hierarchy is a proper hierarchy.

Even though the seminal result of Fagin [Fag74] which established that the complexity class NP –those problems computable in nondeterministic polynomial time– is exactly the set of problems describable in existential second-order logic, is usually considered the starting point in the consistent development of finite model theory, and since then a considerable amount of effort was devoted to the study of different fragments of second-order logic, when it comes to the topic of this dissertation, we note that higher-order logics beyond second-order have not received the same level of attention in the context of finite model theory. However, judging by some of the recent work done in the area [HT03, Kol04b, Kol05, HT06b, HT06a], we think this could be starting to change. Of course those recent works have some important antecedents as [Lei89, HS91].

In the present work, we try to make a contribution to the knowledge in this area of research in finite model theory and relational databases. We investigate the expressive power of different fragments of higher-order logics over finite relational structures with special emphasis in higher-order logics of order greater than or equal three. Our main results were inspired by a work of Makowsky and Pnueli in second-order logic [MP96], and concern the study of the effect on the expressive power of higher-order logics, of simultaneously bounding the arity of the higher-order variables and the alternation of quantifiers. We call the resulting hierarchies of formulae, arity and alternation hierarchies, and show that they are proper in higher-order logics.

This dissertation is organized as follows. In Chapter 2, we provide the theoretical background and introduce notation and terminology. In Chapter 3 we present the central subject of the thesis, namely higher-order logics. This chapter also serves as a review of known result regarding the expressive power of these logics over finite relational structures. Our contributions are presented in Chapters 4, 5, 6, and 7. We outline the results included in these chapters in the next section. Finally, in Chapter 8 we conclude our exposition with some conjectures, and comment possible lines to continue this work in the future.

1.1 Outline of Contributions

We study two different kind of arity and alternation hierarchies in higher-order logics –the AA^i hierarchies and the HAA^i hierarchies. A given layer $AA^i(r, m)$ in the AA^i hierarchy of higher-order logic formulae of order $i + 1$, is formed by the class of prenex $(i+1)$-th order logic formulae in which the maximal-arity of the higher-order variables is bounded by r and the number of alternating blocks of quantifiers is bounded by m. There are two important subtleties which should be taken into account. Firstly, by maximal-arity we refer to a generalization for higher-order variables of the notion of arity of a second-order variable, not just the maximal of the arities of the quantified variables. We introduce this notion formally in Chapter 3. Secondly, the formulae in the AA^i hierarchies are in prenex normal form, but they are *not* necessarily in generalized Skolem normal form (GSNF). This means that the order of the quantifiers in the prefix of the formulae in the AA^i hierarchies may be mixed and that all alternations of quantifiers of whichever order are considered. For instance, let φ be a third-order formula of the form $\exists x \forall X \exists \mathcal{X}(\psi)$ where x, X and $\mathcal{X}$ are first-, second- and third-order variables, respectively, both X and $\mathcal{X}$ have maximal-arity 4, and ψ is a quantifier free third-order subformula. It follows that φ is in prenex normal form and it is precisely in the level $AA^2(4,3)$ of the AA^2 hierarchy of third-order logic formulae, but it is not in GSNF. In order for a higher-order logic formula of order i to be in GSNF, all the quantifiers of order i in the formula must precede all the remaining quantifiers in the prefix.

The higher-order logic formulae which are in GSNF comprise well known hierarchies whose levels are denoted Σ^i_m and Π^i_m. The class Σ^i_m consists of those higher-order logic formulae in GSNF in which the quantifiers of order $i+1$ are arranged into at most m alternating blocks. Π^i_m is defined dually. The levels $HAA^i(r,m)$ of the second kind of hierarchies of arity and alternation that we study in this dissertation –the HAA^i hierarchies– are defined as the restriction of the classes $\Sigma^i_m \cup \Pi^i_m$ to formulae in which the higher-order variables of all orders up to $i+1$ have maximal-arity at most r. That is, in the HAA^i hierarchies the formulae are in GSNF, and the alternations are counted as in Σ^i_m and Π^i_m. For instance, the formula $\exists \mathcal{X} \forall \mathcal{Y} \exists X x(\varphi)$, where x, X, $\mathcal{X}$ and $\mathcal{Y}$ are first-, second- and third-order variables, respectively, $\mathcal{X}$, $\mathcal{Y}$, and X have maximal-arity 4, and ψ is a quantifier free third-order subformula, is in the level $HAA^2(4,2)$ of the HAA^2 hierarchy of third-order logic formulae. The same formula is however in the level $AA^2(4,3)$ of the AA^2 hierarchy of second-order logic formulae.

In Chapters 4 and 5 we present our main contributions regarding the AA^i hierarchies and the HAA^i hierarchies, respectively. The main results presented in these chapters appear in [FT06, FT07].

Generalizing a result of Makowsky and Pnueli [MP96] who proved that the AA^1 hierarchy of second-order logic formulae is proper, we prove in Chapter 4 that the same holds for every AA^i hierarchy of higher-order logic formulae of order $i+1$. More precisely, we show that for every $i, r, m \geq 1$, there are Boolean queries not expressible in $AA^i(r, m)$ but expressible in $AA^i(r + c(r), m + 6)$, where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$.

From the perspective of database query languages this means that, for every $i \geq 2$, if we simultaneously increase the arity of the quantified relation variables by one and the number of alternating blocks of quantifiers by six in the fragment of higher-order relational calculus of order i, AA^{i-1}, then we can express more queries.

We get our result by roughly adapting the strategy in [MP96] to each higher-order logic of order $i \geq 2$. We could outline this strategy by saying that it consists in considering the set $AUTOSAT(F)$ of formulae of a given logic F which, encoded as finite structures, satisfy themselves. As the well known diagonalization argument applies here, when F is a level of an AA^i hierarchy of arity and alternation, it follows that $AUTOSAT(F)$ is not definable in F, but is definable in a higher level of the same hierarchy.

Also in Chapter 4, we explore an alternative strategy to prove the properness of arity and alternation hierarchies in higher-order logics, by means of a simpler complexity-theoretical type of argument involving a variant of the time-hierarchy theorem for alternating Turing machines with bounded alternation. We note that this approach would provide proofs of essentially the same type of results. However, such proofs would be existential while in the work of Makowsky and Pnueli as well as in our work constructive proofs are given, since computable queries which separate the classes are exhibited. Furthermore, to the best of our knowledge there is *no* written source for the variants of the time-hierarchy theorem required by this strategy.

The work carried on in Chapter 4 adapting the strategy in [MP96] to prove the properness of the AA^i hierarchies, allowed us to see and to prove that for $i \geq 2$, the HAA^i hierarchy of higher-order formulae of order $i+1$ is also proper. We present this result in Chapter 5 where we prove that for every $i \geq 2$ and every $r, m \geq 1$, there are Boolean queries not expressible in $HAA^i(r, m)$ but expressible in $HAA^i(r + c(r), m + 2)$ where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$. Interestingly, it is not known whether the corresponding version of this hierarchy for second-order logic, i.e., the HAA^1 hierarchy, is proper. Note that the HAA^1 hierarchy is the hierarchy denoted as SAA in [MP96]. It was established there that, for every $r, m \geq 1$, $AUTOSAT(HAA^1(r, m))$ is complete for PSPACE under polynomial-time reductions. But the properness of the HAA^1 hierarchy of second-order logic formulae is *not* investigated in that paper.

We conclude Chapter 5 examining a version for finite models of Tarski's famous theorem on the undefinability of truth which was introduced by M. Mostowski in [Mos01, Mos03]. We use this theorem together with the corresponding concept of finite model truth definitions studied in those papers, and the characterization given in [Kol04b, Kol05] of the logics for which the prenex classes of higher-order logics Σ^i_m define finite model truth, to give a sufficient condition for the HAA^1 hierarchy to be proper over finite structures. The condition which needs to be met for that to be the case is that, for every vocabulary σ there is a fixed k such that the data complexity for the first-order formulae of vocabulary σ is in NTIME(n^k). The question of whether this is actually the case is an important open question in descriptive complexity (see [Imm99]). Furthermore, as shown in [Kol04a], this is also a sufficient condition for the Σ^1_1 arity hierarchy to be strict over finite structures. This arity hierarchy is known to be strict if we allow vocabularies of arbitrary arity [Ajt83], but its strictness over vocabularies of a fixed arity is still open.

In Chapter 6 we again study in the context of higher-order logics the set $AUTO$-$SAT(F)$ of the sentences of a given logic F which encoded as finite structures satisfy themselves, but this time from a different perspective. We show that when F is one of the prenex fragments Σ^1_m of second-order logic, it follows that $AUTOSAT(F)$ becomes a complete problem for the corresponding prenex fragment Σ^2_m of third-order logic. We prove that $AUTOSAT(\Sigma^1_m)$ is hard for Σ^2_m by means of a polynomial-time reduction from the Σ^1_m theory of the Boolean model to $AUTOSAT(\Sigma^1_m)$. The Σ^1_m theory of the Boolean model is the set of Σ^1_m-formulae which are satisfied by a two element structure of the Boolean vocabulary, i.e., a vocabulary with two constant symbols that are interpreted by the two elements, respectively, and which has no relations or function symbols. As shown in [HT05, HT06a] the Σ^1_m theory of the Boolean model is complete for Σ^2_m under polynomial-time reductions. Then we prove that the problem is indeed in Σ^2_m, by building a *nondeterministic* Turing machine with an oracle in the fragment Σ^p_{m-1} of the polynomial-time hierarchy which decides $AUTOSAT(\Sigma^1_m)$ in time $\mathcal{O}(2^{n^c})$. Since by the characterization of the prenex fragments of higher-order logics in [HT03, HT06b], Σ^2_m captures the complexity class $\bigcup_{c \in \mathbb{N}} \mathrm{NTIME}(2^{n^c})^{\Sigma^p_{m-1}}$, it follows that $AUTOSAT(\Sigma^1_m)$ is in Σ^2_m.

Finally, in Chapter 7 we take a closer look to the restricted second-order logic SO^ω introduced by Dawar in [Daw98]. In particular, we further investigate its connection with the concept of relational complexity studied by Abiteboul, Vardi and Vianu in [AVV97]. We believe that the results proved in this chapter can be generalized to higher-order logics beyond second-order, and thus provide some additional information on the expressive power of restricted fragments of higher-order logics.

Among other interesting results, Dawar proved in [Daw98] that the existential fragment of SO^ω is equivalent to the nondeterministic inflationary fixed-point logic NFP. Since by a result in [AVV97], NFP captures the *relational* complexity class NP_r (see definition in Section 2.5), it then follows that the existential fragment of SO^ω captures NP_r. In Chapter 7, we give a direct proof, in the style of the proof of Fagin's theorem, of this fact.

Also in Chapter 7, we define formally the concept of relational machine with *relational oracle*. Up to our knowledge, this is the first attempt to formally define the *relational* polynomial-time hierarchy in terms of relational machines with *relational oracles*, i.e., oracles consisting on classes of relational structures closed under isomorphisms instead of sets of strings. Using these relational machines with relational oracles, we show the exact correspondence between the prenex fragments of SO^ω and the levels of the *relational* polynomial-time hierarchy. This allows us to establish a direct connection between the *relational* polynomial hierarchy and SO^ω without using the Abiteboul and Vianu normal form for relational machines (see [AV95]), which in turn would have required us to define the oracles for the relational machines as sets of strings, and consequently to use relational machines with oracles decided by standard Turing machines.

Chapter 2

Preliminaries

The goal of this chapter is to provide the theoretical background for our dissertation, and at the same time, to introduce notation and terminology.

2.1 Background from Relational Databases

Since Codd defined the relational model in 1970 [Cod70], first-order logic, or equivalently relational calculus, has been one of the main query languages for relational databases. Codd considered the domain of the relational databases to be infinite and thus he actually used a subset of first-order formulae which express only domain-independent queries, which among other desirable properties always evaluate to finite relations.

Following a more modern approach which has become the norm for the rigorous study of expressibility of query languages over relational databases, we consider the domain of a database instance to be *finite*. In this way we can benefit from the theoretical framework, formal tools and results from finite model theory, and we do not need to worry about domain-independent queries.

Given that attribute names associated with different relation columns are not relevant for our work, we take what is known as an unnamed perspective to relational databases [AHV94]. Thus, we define a *relational database schema*, or shortly a *schema*, as a set of relation symbols, each associated with a natural number, its *arity*. We do not allow constraints in the schema, and we do not allow constant symbols either.

Let $\sigma = \{R_1, \ldots, R_s\}$ be a schema, and for $1 \leq i \leq s$, let r_i be the associated arity of the relation symbol R_i. A *database instance* of signature σ, or shortly a *database*, is a structure $\mathbf{I} = \langle I, R_1^{\mathbf{I}}, \dots, R_s^{\mathbf{I}} \rangle$, where I is a finite set called the *domain* which contains exactly all individual elements of the database, and for $1 \leq i \leq s$, $R_i^{\mathbf{I}}$ is a relation of arity r_i on I, i.e., $R_i^{\mathbf{I}} \subseteq I^{r_i}$. For the sake of clarity, we sometimes write $dom(\mathbf{I})$ instead of I. We denote as $\mathcal{B}_\sigma$ the class of all (finite) databases of schema σ.

A *k-tuple* over a database $\mathbf{I}$, with $k > 0$, is a tuple of length k formed with elements from I, i.e., $(a_1, \dots, a_k) \in I^k$. Often we denote a k-tuple of $\mathbf{I}$ as $\bar{a}$, since usually its length is clear from the context. Sometimes we abuse the notation and write $(\bar{a}_1, \dots, \bar{a}_s)$, where $s > 1$ and $\bar{a}_1, \dots, \bar{a}_k$ are tuples of length $k_1, \dots, k_s > 0$, respectively, to denote the $(k_1 + \cdots + k_s)$-tuple $(a_{11}, \dots, a_{1k_1}, \dots, a_{s1}, \dots, a_{sk_s}) \in I^{k_1 + \cdots + k_s}$.

If $\mathbf{I}$ and $\mathbf{J}$ are databases of some schema σ, we denote as $\mathbf{I} \simeq \mathbf{J}$ the existence of an *isomorphism* between $\mathbf{I}$ and $\mathbf{J}$, i.e., a bijection $f : I \to J$ such that $(a_1, \dots, a_r) \in R^{\mathbf{I}}$ iff $(f(a_1), \dots, f(a_r)) \in R^{\mathbf{J}}$ for all r-ary relation symbol $R \in \sigma$ and all r-tuples $(a_1, \dots, a_r) \in I^r$. Intuitively, the databases $\mathbf{I}$ and $\mathbf{J}$ are isomorphic if $\mathbf{J}$ can be formed by replacing every element $a \in I$ by $f(a) \in J$ without changing the structure of the database. An *automorphism* of $\mathbf{I}$ is an isomorphism from $\mathbf{I}$ to itself.

2.1.1 Computable Queries

From a conceptual point of view it is desirable for a model of computation of queries to be *representation independent* [Ull88, Ull89]. This means, roughly, that queries to databases which represent the "same" reality should evaluate to the same result. In mathematical terms, we can partially capture this concept if we ask queries to isomorphic databases to evaluate to the same result. This is the central idea in the notion of computable query defined by Chandra and Harel [CH80] which we use in this work.

We consider only *typed* queries, that is, queries which have a fixed arity. In [CH80], queries which may evaluate to relations of different arities in different databases are also considered, and they are called *untyped* queries. For an in depth study on untyped queries see [Tur01a].

Definition 2.1. Fixing a schema σ and an integer $r > 0$, we define a *computable query* of arity r and schema σ, as a partial function $q : \mathcal{B}_\sigma \to \mathcal{R}_r$, where $\mathcal{R}_r$ denotes the class of all finite relations of arity r, and where q has the following properties:

i. For every database $\mathbf{I}$ of schema σ on which q is defined, $q(\mathbf{I}) \subseteq I^r$;

ii. q is a partial recursive function in some linear encoding of the database instances in $\mathcal{B}_\sigma$;

iii. q preserves isomorphisms, i.e., for every pair of databases of schema σ, $\mathbf{I}$ and $\mathbf{J}$, and for every isomorphism $f : I \to J$, either $q(\mathbf{J}) = \{(f(a_1), \ldots, f(a_r)) : (a_1, \ldots, a_r) \in q(\mathbf{I})\}$, or q is undefined on both $\mathbf{I}$ and $\mathbf{J}$.

Similarly, we define a *computable Boolean query* of schema σ, as a partial function $q : \mathcal{B}_\sigma \to \{0, 1\}$, with the following properties:

i. q is a partial recursive function in some linear encoding of the database instances in $\mathcal{B}_\sigma$;

ii. q preserves isomorphisms, i.e., for every pair of databases of schema σ, $\mathbf{I}$ and $\mathbf{J}$, which are isomorphic, either $q(\mathbf{J}) = q(\mathbf{I})$, or q is undefined on both $\mathbf{I}$ and $\mathbf{J}$.

Boolean queries may also be regarded as 0-ary queries.

Note that a query may be *partial*, that is, it may be defined on a proper sub-class of $\mathcal{B}_\sigma$. Otherwise, if the query is defined on the whole class of databases of schema σ, we say that it is a *total query*.

2.2 Background from Finite Model Theory

There are excellent books on finite model theory [EF99, Imm99, Lib04]. We just introduce briefly the concepts and notation needed for this work.

From the point of view of finite model theory, a *database schema* is simply a *relational vocabulary*, and a *database instance* of some schema σ is a finite relational structure of vocabulary σ, or *σ-structure*. Consequently, given a vocabulary σ, the class of all finite σ-structures is exactly what we denoted as $\mathcal{B}_\sigma$ in Section 2.1.

We use the notion of a logic in a general sense. A formal definition would only complicate the presentation and is unnecessary for our work. We refer the reader to [Ebb85] for a formal study of a general framework of abstract logics.

As it is usual in this framework, we regard a logic as a pair $(\mathcal{L}, \models_{\mathcal{L}})$, where $\mathcal{L}$ is a mapping which assigns to each vocabulary σ a set of words (the set of $\mathcal{L}$-sentences of vocabulary σ) and $\models_{\mathcal{L}}$ (the $\mathcal{L}$-satisfaction relation) is a relation between structures and $\mathcal{L}$-sentences. We are not interested here in proof theoretic concepts like axioms and inference rules. We only consider vocabularies, which are purely relational, i.e., there are no constant symbols nor function symbols. We assume always that the vocabulary includes a symbol for equality.

We use the classical Tarski's semantics, except that as our framework is finite model theory, *only finite* structures, or interpretations are considered. Thus our structures will always be *finite relational structures*, i.e., finite structures of a relational vocabulary.

Consequently, if $\mathcal{L}$ is a logic, the notions of *satisfaction* $\models_{\mathcal{L}}$ and *equivalence* between structures $\equiv_{\mathcal{L}}$, are related to only finite structures. If the logic is clear from the context, we use the notations $\models$ and $\equiv$, respectively.

If $\mathcal{L}$ is a logic and σ is a vocabulary, we denote as $\mathcal{L}[\sigma]$ the class of formulas from $\mathcal{L}$ of vocabulary σ.

For φ a sentence in $\mathcal{L}[\sigma]$ we denote by $Mod(\varphi)$ the class of finite σ-structures $\mathbf{I}$ such that $\mathbf{I} \models \varphi$. A class of finite σ-structures $\mathcal{C}$ closed under isomorphisms is *definable* by an $\mathcal{L}$-sentence if $\mathcal{C} = Mod(\varphi)$ for some sentence $\varphi \in \mathcal{L}[\sigma]$. If $\mathcal{L}$ and $\mathcal{L}'$ are logics, then $\mathcal{L} \subseteq \mathcal{L}'$ denotes that $\mathcal{L}'$ is at least as expressive as $\mathcal{L}$, i.e., all classes of models definable by $\mathcal{L}$-sentences are also definable by $\mathcal{L}'$-sentences. $\mathcal{L} = \mathcal{L}'$ holds if $\mathcal{L} \subseteq \mathcal{L}'$ and $\mathcal{L}' \subseteq \mathcal{L}$. $\mathcal{L} \subset \mathcal{L}'$ holds if $\mathcal{L} \subseteq \mathcal{L}'$ and $\mathcal{L} \neq \mathcal{L}'$.

By $\varphi(x_1, \ldots, x_r)$ we denote a formula of some logic whose free variables are *exactly* $\{x_1, \ldots, x_r\}$. Sometimes, we use the same notation also when the set of free variables of the formula is *strictly* included in $\{x_1, \ldots, x_r\}$, even when the formula is a sentence. We denote as *free*(φ) the set of free variables of the formula φ.

Recall that a *valuation* is a function which assigns to every variable in the logic, an element of a given structure. If $\varphi(x_1, \ldots, x_r)$ is a formula of a vocabulary σ, $\mathbf{I}$ is a σ-structure, and $\bar{a} = (a_1, \ldots, a_k)$ is an r-tuple over I, we use the notation $\mathbf{I} \models \varphi(x_1, \ldots, x_r)[\bar{a}]$ to denote that φ is satisfied by the structure $\mathbf{I}$, under every valuation v such that $v(x_i) = a_i$ for $1 \leq i \leq r$.

Then, we consider the set of all such tuples for which there is a corresponding valuation which satisfies φ, as follows:

$$\varphi^{\mathbf{I}} = \{(a_1, \ldots, a_r) \in I^k : \mathbf{I} \models \varphi(x_1, \ldots, x_r)[a_1, \ldots, a_r]\}$$

That is, $\varphi^{\mathbf{I}}$ is the relation defined by φ in the structure $\mathbf{I}$, and its arity is given by the number of free variables in φ.

Formally, we say that a formula $\varphi(x_1, \ldots, x_r)$ of vocabulary σ, *expresses* a query q of schema σ, if for every database $\mathbf{I}$ of schema σ where q is defined, it holds that $q(\mathbf{I}) = \varphi^{\mathbf{I}}$. Similarly, a sentence φ expresses a Boolean query q if for every database $\mathbf{I}$ of schema σ where q is defined, it holds that $q(\mathbf{I}) = 1$ iff $\mathbf{I} \models \varphi$.

Note that when using logics as query languages only *total* queries are considered. This is due to Tarski's semantics. One way in which *Partial* queries can be expressed is through pairs of formulas, where the first formula is a sentence which defines the domain of the query [Tur01a]. Anyway, in this work we deal only with total queries.

2.2.1 Some Logics Relevant in Finite Model Theory

We comment next on some logics which are referenced on this work. For that, we assume that the reader is familiar with the syntax and semantics of first-order logic (FO).

The weakness of first-order logic as a query language is well known. As early as in 1975, Fagin showed that a very simple query, namely the transitive closure query over finite relations, is not expressible in first-order logic [Fag75]. Aho and Ullman [AU79] rediscovered this result in 1979 and bought it to the attention of the database community.

Intuitively, one reason for the weakness of first-order logic as a query language is that there is no way of expressing any form of iteration or recursion. This suggests that by adding some kind of induction operation, we could augment the expressive power of first-order logic. This is the idea behind fixed point logics. Next, we comment briefly some of these logics to keep this work as self contained as possible.

Let σ be a vocabulary and R be a k-ary relation symbol which is not in σ. A formula $\varphi(x_1, \ldots, x_k, R) \in FO[\sigma \cup \{R\}]$ and a structure $\mathbf{I} \in \mathcal{B}_\sigma$ give rise to an operator $F^\varphi : \mathcal{P}(I^k) \longrightarrow \mathcal{P}(I^k)$ defined by:

$$F^\varphi(R^{\mathbf{I}}) = \{(a_1, \ldots, a_k) \in I^k : \langle \mathbf{I}, R^{\mathbf{I}} \rangle \models \varphi(x_1, \ldots, x_k)[a_1, \ldots, a_k]\},$$

where $\langle \mathbf{I}, R^{\mathbf{I}} \rangle$ denotes the expansion of $\mathbf{I}$ interpreting R as $R^{\mathbf{I}}$. A relation $R^{\mathbf{I}} \subseteq I^k$ is a *fixed point* of the operator F^φ if $F^\varphi(R^{\mathbf{I}}) = R^{\mathbf{I}}$.

Each such operator F^φ generates a sequence of stages F^φ_n defined inductively by: $F^\varphi_0 = \emptyset$ and $F^\varphi_{n+1} = F^\varphi(F^\varphi_n)$. When the sequence F^φ_n "converges", i.e., when there is $n_0 \geq 0$ such that $F^\varphi_{n_0} = F^\varphi_{n_0+1}$ and hence, $F^\varphi_{n_0} = F^\varphi_n$ for all $n \geq n_0$, then $F^\varphi_{n_0}$ is a fixed point of F^φ. We denote $F^\varphi_{n_0}$ as F^φ_∞.

F^{φ} is *inflationary* if, for every relation $R^{\mathbf{I}} \subseteq I^k$, it holds that $R^{\mathbf{I}} \subseteq F^{\varphi}(R^{\mathbf{I}})$. In particular, F^{φ} is inflationary if $\varphi(x_1, \ldots, x_k, R)$ is of the form $R(x_1, \ldots, x_k) \vee \psi(x_1, \ldots, x_k, R)$. We call such formulae, *inflationary formulae.* If F^{φ} is inflationary, then F^{φ} is inductive, i.e., the sequence F^{φ}_n is increasing, and hence it reaches a fixed point F^{φ}_{∞}. The *inflationary fixed point logic* (IFP) is defined to be the closure of first-order logic under the operation of taking inflationary fixed points of inflationary formulae. Note that the transitive closure of the edge relation E of a directed graph, which we know is not expressible in first-order logic, can now be expressed easily in IFP. The fixed point F^{φ}_{∞}, where $\varphi(x, y, R)$ is the inflationary formula

$$R(x,y) \vee (E(x,y) \vee \exists z(R(x,z) \wedge R(z,y))),$$

clearly defines the transitive closure of E.

The existence of a fixed point for the sequence F^{φ}_n can also be guaranteed when $\varphi(x_1, \ldots, x_k, R)$ is *positive* in R, i.e., when every occurrence of R in $\varphi(x_1, \ldots, x_k, R)$ is under the scope of an even number of negations. In that case the sequence F^{φ}_n is also increasing, and therefore has a limit, i.e., there is a $n_0 \leq |I|^n$ such that $F^{\varphi}_{n_0} = F^{\varphi}_m$ for every $m \geq n_0$. This limit is the *least fixed point* of the sequence F^{φ}_n. The closure of first-order logic under the operation of taking least fixed points of positive formulae, is usually called *least fixed point logic* (LFP). On finite structures, this logic has exactly the same expressive power than IFP [GS86].

Given an arbitrary first-order logic formula $\varphi(x_1, \ldots, x_k, R)$ and a structure $\mathbf{I}$, the corresponding sequence of stages F^{φ}_n is not necessarily increasing and may or may not converge to a fixed point. So, there are two possibilities. The first is that this sequence reaches a fixed point, i.e., for some $n_0 \in \mathbb{N}$ we have $F^{\varphi}_{n_0} = F^{\varphi}_{n_0+1}$, and thus $F^{\varphi}_{n_0} = F^{\varphi}_m$ for every $m \geq n_0$. If there is such n_0, it must be the case that $n_0 \leq 2^{|I|^k}$. The second possibility is that not such n_0 exists. The *partial fixed point* F^{φ}_{∞} is defined to be $F^{\varphi}_{n_0}$ if $F^{\varphi}_{n_0} = F^{\varphi}_{n_0+1}$ for some $n_0 \leq 2^{|I|^k}$, otherwise $F^{\varphi}_{\infty} = \emptyset$. *Partial fixed point logic* PFP is the closure of first-order logic under the operation defining partial fixed points of arbitrary first-order formulae.

The fixed point logics we have seen so far are obtained by deterministically iterating first-order operators. We see next a fixed point logic introduced by Abiteboul, Vardi and Vianu [AVV97] which is obtained by nondeterministically iterating first-order operators. Given two first-order formulae $\varphi_0(x_1, \ldots, x_k, R)$ and $\varphi_1(x_1, \ldots, x_k, R)$ of a same vocabulary σ, we define a sequence of stages $F^{(\varphi_0, \varphi_1)}_b$

indexed by binary strings $b \in \{0,1\}^*$, as follows:

$$F_{\lambda}^{(\varphi_0,\varphi_1)} = \emptyset, \text{ for the empty string } \lambda$$

$$F_{b\cdot 0}^{(\varphi_0,\varphi_1)} = F_{b}^{(\varphi_0,\varphi_1)} \cup F^{\varphi_0}(F_{b}^{(\varphi_0,\varphi_1)})$$

$$F_{b\cdot 1}^{(\varphi_0,\varphi_1)} = F_{b}^{(\varphi_0,\varphi_1)} \cup F^{\varphi_1}(F_{b}^{(\varphi_0,\varphi_1)}).$$

The *nondeterministic fixed point* of the sequence is $\bigcup_{b\in\{0,1\}^*} F_b^{(\varphi_0,\varphi_1)}$. The *nondeterministic inflationary fixed point logic* (NFP) is the closure of first-order logic under the operation of taking nondeterministic inflationary fixed points, with the restriction that negation cannot be applied to the fixed point operator.

Another way of extending the expressive power of first-order logic is by allowing disjunctions and conjunctions of arbitrary (not just finite) sets of formulae. But the resulting logic, usually denoted as $\mathcal{L}_{\infty\omega}$, is of little use in the study of finite models, since every query (including noncomputable queries) over finite structures is expressible in it. This changes when we concentrate on a bounded variable fragment of $\mathcal{L}_{\infty\omega}$ as all fixed point logics mentioned above can be viewed as fragments of it. More precisely, they can be viewed as fragments of the logic $\mathcal{L}^{\omega}_{\infty\omega} = \bigcup_{k\in\mathbb{N}} \mathcal{L}^{k}_{\infty\omega}$, where $\mathcal{L}^{k}_{\infty\omega}$ denotes the class of formulae of $\mathcal{L}_{\infty\omega}$ that use at most k different variables. The following picture follows from the works of Kolaitis and Vardi [KV92a, KV92b] and Dawar [Daw93].

$$\text{IFP} = \text{LFP} \subseteq \text{NFP} \subseteq \text{PFP} \subset \mathcal{L}^{\omega}_{\infty\omega}$$

The last containment is proper, since there are nonrecursive queries which can be expressed in $\mathcal{L}^{\omega}_{\infty\omega}$ [KV92b] while every query definable in PFP is computable in PSPACE. In fact, as noted by Dawar [Daw98], it can be shown that $\mathcal{L}^{k}_{\infty\omega}$ is complete on ordered structures, where the maximum arity of a relation symbol in its vocabulary is $\leq k$.

2.2.2 Type of a Tuple

Sometimes, we need to consider the properties of a tuple in a relational structure or database, which are definable in a given logic. For this, we use the model theoretic concept of type. The results that we use in this work are mainly from [Daw93, DLW95]. Another excellent source for the subject is [Ott97]. Our notation comes mostly from there.

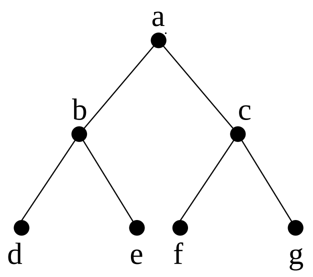

Figure 2.2.2

Definition 2.2. Let $\mathcal{L}$ be a logic, let $\mathbf{I}$ be a relational structure of vocabulary σ, and let $\bar{a} = (a_1, \ldots, a_k)$ be a k-tuple over $\mathbf{I}$. The *$\mathcal{L}$-type* of $\bar{a}$ in I, denoted $tp^{\mathcal{L}}_{\mathbf{I}}(\bar{a})$, is the set of formulas in $\mathcal{L}[\sigma]$ with free variables among $\{x_1, \ldots, x_k\}$ which are satisfied in $\mathbf{I}$ by any valuation which, for $1 \leq i \leq k$, assigns the i-th component of $\bar{a}$ to the variable x_i. In symbols,

$$tp^{\mathcal{L}}_{\mathbf{I}}(\bar{a}) = \{\varphi \in \mathcal{L}[\sigma] : \mathit{free}(\varphi) \subseteq \{x_1, \ldots, x_k\} \text{ and } \mathbf{I} \models \varphi[a_1, \ldots, a_k]\}$$

Note that, the $\mathcal{L}$-type of a given tuple $\bar{a}$ over a relational structure $\mathbf{I}$, includes not only the properties of all sub-tuples of $\bar{a}$, but also the set of all sentences in $\mathcal{L}$ which are true when evaluated on $\mathbf{I}$, i.e., the $\mathcal{L}$-theory of $\mathbf{I}$.

It is not difficult to see that the $\mathcal{L}$-type of two different k-tuples over the same relational structure may be different, even if the types of their corresponding components are the same. Take for instance the complete binary tree of Figure 2.2.2. If we consider the types of singletons, then we have (depending on the expressive power of $\mathcal{L}$) at most three different types. The type of a, the type of b and c, and the type of d, e, f and g. This is the case because b and c as well as d, e, f and g, are interchangeable by automorphisms of the tree and are thus equivalent from the point of view of any logic $\mathcal{L}$ with Tarski's semantics. Now, let us consider the types of the tuples (b, f) and (c, g). Clearly, they are not interchangeable by automorphisms of the tree. So, their type may be different (again, depending on the expressive power of $\mathcal{L}$) even when their components b and c as well as f and g have the same type.

According to Definition 2.2, a type is an *infinite* set of formulas which is *consistent*, i.e., there is a structure and a valuation which satisfy all the formulas in the set. Moreover, the set is *maximally consistent*, that is, if we add any formula to the set, we loose the consistency of the set. Therefore, we can think of the type of a tuple as a maximally consistent set of formulae, without having a particular relational structure in mind. If α is the $\mathcal{L}$-type of some tuple $\bar{a}$, i.e., α is a maximally consistent set of $\mathcal{L}$-formulae of some vocabulary σ, we say that a given σ-structure $\mathbf{I}$ *realizes* the type α if there is a tuple $\bar{a}$ over $\mathbf{I}$ such that $tp_{\mathbf{I}}^{\mathcal{L}}(\bar{a}) = \alpha$.

The notion of FO-type, usually encountered in classical (infinite) model theory, is not of much value in the context of finite model theory, since every tuple can be characterized up to isomorphism by its FO-type. However, if we consider logics which are weaker than first-order as to expressive power, that is not longer the case. In this context, a notion which has proven to be of great importance is that of FO^k-type, where for $k > 0$, FO^k denotes the fragment of first-order logic where only formulae with variables in $\{x_1, \ldots, x_k\}$ are allowed. Note that the class of queries expressible in FO^k is strictly included in the class of queries expressible in FO and that FO $= \bigcup_{k>0}$ FO^k.

Definition 2.3. For $k \geq r \geq 1$, we denote by $\equiv^k$ the *equivalence relation* induced in the set of r-tuples over a given structure $\mathbf{I}$, by equality of FO^k-types of r-tuples. That is, for every pair of r-tuples $\bar{a}$ and $\bar{b}$ over $\mathbf{I}$, $\bar{a} \equiv^k \bar{b}$ iff $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a}) = tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{b})$.

Kolaitis and Vardi [KV92b] showed that on finite structures, the equivalence relation $\equiv^k$ and the apparently stronger notion of equivalence of $\mathcal{L}^k_{\infty\omega}$-types, actually coincide. That is, they showed that if $\bar{a} \equiv^k \bar{b}$ over a given finite structure $\mathbf{I}$ of vocabulary σ, then for every $\varphi \in \mathcal{L}^k_{\infty\omega}[\sigma]$, it holds that $\mathbf{I} \models \varphi[\bar{a}]$ iff $\mathbf{I} \models \varphi[\bar{b}]$, and vice versa.

Definition 2.4. Let $k \geq 1$, let σ be a relational vocabulary, and let $\mathbf{I}$ be a σ-structure. We say that a relation $R^{\mathbf{I}}$ of arity $r \leq k$ is *closed* under the equivalence relation $\equiv^k$ iff, for every pair of r-tuples $\bar{a}$ and $\bar{b}$ over $\mathbf{I}$, if $\bar{a} \in R^{\mathbf{I}}$ and $\bar{a} \equiv^k \bar{b}$, then $\bar{b} \in R^{\mathbf{I}}$.

Kolaitis and Vardi [KV92b] also showed that a query q of arity $r \leq k$ is expressible in $\mathcal{L}^k_{\infty\omega}$ iff it is closed under $\equiv^k$. The same is true for the fixed point logics introduced earlier, since the classes of queries expressible in those logics are strictly included in the class of queries expressible in $\mathcal{L}^\omega_{\infty\omega}$.

We must mention that there is an elegant Ehrenfeucht-Fraïssé type of characterization of the relation $\equiv^k$ in terms of pebble games. It was first introduced by Barwise [Bar77] and then reinvented and successfully used by Immerman [Imm82] and Poizat [Poi82]. This characterization is an important source of intuition on the relation $\equiv^k$, but since it is also very well known and explained in any of the books on finite model theory referenced at the beginning of this section, we omit its description here.

Although types are infinite sets of formulas, due to a result of A. Dawar [Daw93], a *single* FO^k-formula is equivalent to the FO^k-type of a tuple over a given database. The equivalence holds for all databases of the same schema.

Lemma 2.5 (Corollary 2.18 in [Daw93]). *For every schema σ, for every database $\mathbf{I}$ of schema σ, for every $k \geq 1$, for every $1 \leq r \leq k$, and for every r-tuple $\bar{a}$ over $\mathbf{I}$, there is an FO^k-formula $\alpha \in tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a})$ such that for any database $\mathbf{J}$ of schema σ and for every r-tuple $\bar{b}$ over $\mathbf{J}$, $\mathbf{J} \models \alpha[\bar{b}]$ iff $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a}) = tp_{\mathbf{J}}^{\mathrm{FO}^k}(\bar{b})$.*

If an FO^k formula α satisfies the condition of Lemma 2.5, we say that α *isolates* the $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a})$.

It is well known that the relation $\equiv^k$ is uniformly definable in LFP, or equivalently in IFP.

Theorem 2.6 ([KV92a, DLW95]). *For every relational vocabulary σ and every $k \geq 1$, there is a formula φ_σ^k of* IFP*, with $2k$ free variables, such that on any finite σ-structure $\boldsymbol{I}$, given two k-tuples $\bar{a}$ and $\bar{b}$ on $\boldsymbol{I}$, $\boldsymbol{I} \models \varphi_\sigma^k[\bar{a}, \bar{b}]$ iff $\bar{a} \equiv^k \bar{b}$.*

Moreover, a total ordering of the FO^k-types of a given vocabulary is also uniformly definable in LFP, or equivalently in IFP.

Theorem 2.7 ([AV95, DLW95]). *For every relational vocabulary σ and every $k \geq 1$, there is a formula λ_σ^k of* IFP*, with $2k$ free variables, such that on any finite σ-structure $\boldsymbol{I}$, λ_σ^k defines a reflexive and transitive relation $\leq^k$ on k-tuples such that for every two k-tuples $\bar{a}$ and $\bar{b}$ on $\boldsymbol{I}$, either $\bar{a} \leq^k \bar{b}$ or $\bar{b} \leq^k \bar{a}$ and both $\bar{a} \leq^k \bar{b}$ and $\bar{b} \leq^k \bar{a}$ hold iff $\bar{a} \equiv^k \bar{b}$.*

That is, for every relational vocabulary σ and every $k \geq 1$, the corresponding IFP-formula λ_σ^k defines, on any finite σ-structure $\mathbf{I}$, a preorder such that the corresponding equivalence relation is $\equiv^k$. Thus, λ_σ^k can be seen as defining a total order on the equivalence classes of $\equiv^k$.

2.3 Background from Complexity Theory

We assume that the reader is familiar with Turing machines and the basic notions of computability theory. We start from there and, mainly to fix the notation, survey briefly the notions of complexity theory used in this work. Both [Pap94] and [BDG95] are excellent source books for the topic.

Let t and s be functions on the natural numbers such that $t(n) \geq n+1$ and $s(n) \geq 1$. As usual, DTIME($t(n)$) and NTIME($t(n)$) denote the classes of languages decidable by deterministic and nondeterministic Turing machines, respectively, whose running time is bounded above by t. DSPACE($s(n)$) and NSPACE($s(n)$) denote the classes of languages decidable by deterministic and nondeterministic Turing machines, respectively, whose work space is bounded above by $s(n)$.

We assume that the reader is familiar with the following complexity classes:

- the class P of the languages decidable in polynomial-time, $\mathrm{P} = \bigcup_{c \in \mathbb{N}} \mathrm{DTIME}(n^c)$;
- the class NP of languages decidable in nondeterministic polynomial time, $\mathrm{NP} = \bigcup_{c \in \mathbb{N}} \mathrm{NTIME}(n^c)$;
- The class PSPACE of languages decidable in polynomial space, $\mathrm{PSPACE} = \bigcup_{c \in \mathbb{N}} \mathrm{DSPACE}(n^c)$.

If $\mathcal{C}$ is a complexity class, we denote as co$\mathcal{C}$ the class of languages whose complements are in $\mathcal{C}$.

Recall that P is closed under complementation and also $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}$, but it is not known whether this containment is proper and whether NP equals coNP. Regarding space complexity, by Savitch's theorem, nondeterministic polynomial space collapses to deterministic polynomial space, i.e., $\mathrm{PSPACE} = \bigcup_{c \in \mathbb{N}} \mathrm{NSPACE}(n^c)$.

The concept of *reducibility* is the standard way in complexity theory of comparing the "difficulty" of deciding two different languages. It also allows us to formalize the concept of "the most difficult" languages of a class which are known as complete problems.

Given two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ of some alphabet A, we say that $\mathcal{L}_1$ is *polynomial-time (many-one) reducible* to $\mathcal{L}_2$ iff there is a function $f : A^* \longrightarrow A^*$, computable in polynomial-time, and such that $x \in \mathcal{L}_1$ iff $f(x) \in \mathcal{L}_2$ holds for all $x \in A^*$.

A language $\mathcal{L}$ is *hard* for a complexity class $\mathcal{C}$ iff for every language $\mathcal{L}'$ in $\mathcal{C}$, it holds that $\mathcal{L}'$ is polynomial-time reducible to $\mathcal{L}$. A language $\mathcal{L}$ is *complete* for a complexity class $\mathcal{C}$ iff it is hard for $\mathcal{C}$ and $\mathcal{L} \in \mathcal{C}$.

The theory of complete problems was initiated with the famous work of Cook [Coo71], who demonstrated that the set *SAT* of satisfiable propositional formulae is complete for NP.

2.3.1 Oracle and Alternating Turing Machines

Turing gave his original definition of oracles for the case of undecidable languages. An *oracle* for an *undecidable* language $\mathcal{L}$, is a mythical device that is able to answer in unit time whether a given word belongs to $\mathcal{L}$ or not. In complexity theory, *oracles* still are able to answer in unit time whether a given word belongs to a language $\mathcal{L}$ or not, but $\mathcal{L}$ must be a *decidable* language. Informally, we can think of an oracle as a subroutine which can magically solve a difficult problem in unit time. In this setting, they are used to compare the complexity of deciding difficult languages. Suppose we have a language $\mathcal{L}_1$ for which there is no known algorithm which decides it in P. If there is an algorithm that can decide $\mathcal{L}_1$ in P, provided that it uses as a subroutine an oracle for another language $\mathcal{L}_2$, then this indicates that the language $\mathcal{L}_2$ gives us some additional information which allows us to overcome the difficulty of deciding $\mathcal{L}_1$ in P.

Definition 2.8. An *oracle Turing machine* is a Turing machine with a distinguished tape, called oracle tape, and three distinguished states $q_?$, the query state, and q_{YES}, q_{NO}, the answer states.

The computation of an oracle Turing machine requires that an oracle language be fixed previously to the computation. Let $\mathcal{L}$ be an arbitrary language. The computation of a Turing machine M with oracle $\mathcal{L}$ proceeds like in an ordinary Turing machine, except for transitions from the query state. From the query state M transfers into the state q_{YES} if the string currently in the oracle tape belongs to $\mathcal{L}$; otherwise, M transfers into the state q_{NO}.

The time complexity of deterministic as well as nondeterministic Turing machines with oracles, is defined in exactly the same way as for ordinary Turing machines. Every query state is counted as an ordinary step. If $\mathcal{C}$ is any deterministic or non-deterministic complexity class and $\mathcal{L}$ is a language, then $\mathcal{C}^{\mathcal{L}}$ is the class of languages decided by *oracle machines* of the same sort and time bound as in $\mathcal{C}$, only that the machines have now an oracle $\mathcal{L}$. If $\mathcal{C}'$ is also a complexity class defined using bounds on resources, then $\mathcal{C}^{\mathcal{C}'}$ denotes $\bigcup_{\mathcal{L} \in \mathcal{C}'} \mathcal{C}^{\mathcal{L}}$.

An important complexity class which is usually defined in terms of oracle Turing machines, is the *polynomial-time hierarchy* (PH). The levels of this hierarchy are defined as follows:

$$\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathrm{P}$$

and for $m > 0$,

$$\Delta^p_{m+1} = \mathrm{P}^{\Sigma^p_m} \qquad \Sigma^p_{m+1} = \mathrm{NP}^{\Sigma^p_m} \qquad \Pi^p_{m+1} = \mathrm{coNP}^{\Sigma^p_m}$$

The complexity class PH is the union of all the complexity classes in the polynomial-time hierarchy, i.e., $\mathrm{PH} = \bigcup_{m\in\mathbb{N}} \Sigma^p_m$.

The polynomial-time hierarchy can also be defined in terms of alternating Turing machines. An *alternating Turing machine* is a variation of the nondeterministic machine which apart from the usual "existential states", admits a new kind of state called universal, which influence the notion of acceptance.

Definition 2.9. An *alternating Turing machine* is a Turing machine with two different kinds of non-final states: the existential states and the universal states. The notion of acceptance is defined by induction on the computation tree. The alternating Turing machine in a given configuration c *accepts* iff

- c is in a final accepting state, or
- c is in an existential state and at least one of its sons in the computation tree accepts, or
- c is in an universal state, it has at least one son in the computation tree, and all its sons in the computation tree accept.

Note that an alternating machine all of whose non-final states are existential, is essentially a nondeterministic machine.

In this work we use alternating machines which can switch a bounded number of times between existential and universal states. They are called Σ_m and Π_m machines. A Σ_m *machine* is an alternating machine which starts in an existential state and switches between existential and universal states at most $m-1$ times on a single computation path. A Π_m *machine* can also switch at most $m-1$ times between existential and universal states, but it starts in an universal state.

We denote as $\Sigma_m\mathrm{TIME}(f(n))$ the class of languages that are decidable by a Σ_m machine in time bounded by f. Similarly, we use $\Pi_m\mathrm{TIME}(f(n))$ to denote the class of languages decidable by a Π_m machine in time time bounded by f.

Using alternating machines, for $m \geq 1$, the level Σ^p_m of the polynomial-time hierarchy corresponds to the class $\bigcup_{c\in\mathbb{N}} \Sigma_m\mathrm{TIME}(n^c)$. The level Π^p_m can be defined as usual, i.e., as the complement of Σ^p_m, or as $\bigcup_{c\in\mathbb{N}} \Pi_m\mathrm{TIME}(n^c)$. However, there is no natural definition of the intermediate levels Δ^p_m of the polynomial-time hierarchy using alternating machines.

2.4 Background from Descriptive Complexity

The typical way of characterizing the expressive power of a given logic over finite structures, is by means of classes of queries defined in terms of the time or space complexity of their evaluation. This approach gave rise to an important field of finite model theory which is known as *descriptive complexity* ([Imm99, EF99, Lib04]). In this setting, the notion of complexity is that of the *complexity of describing* a collection of structures of a relational signature which is closed under isomorphisms, and which is recursive. For instance, we can look at the class of Boolean queries which are definable in first-order logic as a complexity class. Similarly, we can define descriptive complexity classes based in other logics.

What makes this approach to complexity particularly interesting is that it has a close correspondence with the classical classes of computational complexity, which in turn have emerged as natural levels of computational power, certified by robustness criteria and the existence of natural complete problems. As a consequence, matching logics constitute naturally distinguished levels of expressiveness.

Complexity classes in classical complexity theory are usually defined as classes of languages which are decidable by Turing machines. Therefore, if we want to talk about the complexity of logics on finite structures, we need to encode finite structures as words over a fixed alphabet.

We encode any finite relational structure $\mathbf{I}$ of a given vocabulary σ as a word over the alphabet $\{0,1\}$. The encoding $enc(\mathbf{I})$ of $\mathbf{I}$ is a word of length $O(|I|^k)$, where k is the maximal arity of a relation symbol in σ, or $k = 1$ if σ is the empty vocabulary. We define $enc(\mathbf{I})$ as the concatenation of $1^{|I|}0$ (which encodes the size of $\mathbf{I}$) with the encoding of each relation in $\mathbf{I}$ in some fixed order. To encode the relations we need to assume an ordering on the domain of $\mathbf{I}$. We cannot encode a structure as a string without an ordering. A relation $R^{\mathbf{I}}$ of arity r is encoded as a bit string of length $|I|^r$ where "1" in a given position indicates that the corresponding tuple in the lexicographical order of tuples in I^r is in $R^{\mathbf{I}}$.

In complexity theory n is usually reserved for the length of the input. However, for technical reasons, in descriptive complexity n is usually used to denote the size of the input structure, not the length of its encoding. So, in calculating the complexity of a class of structures, we consider the size of a given input structure $\mathbf{I}$ to be $|I|$, not the length of $enc(\mathbf{I})$.

Since we are interested on relational databases, i.e., we assume large finite structures and much smaller sentences (Boolean queries) which are evaluated on these large finite structures, we use the notion of data complexity. There are however other notions of complexity of a logic, namely combined complexity and expression complexity, but these are more appropriate for others areas such as verification and model checking (see [Lib04] for details).

Let $\mathcal{C}$ be a Turing machine complexity class and $\mathcal{L}$ be a logic. The *data complexity* of $\mathcal{L}$ is in $\mathcal{C}$ if for every vocabulary σ and every sentence $\varphi \in \mathcal{L}[\sigma]$, the corresponding language $\{enc(\mathbf{I}) : \mathbf{I} \in \mathcal{B}_\sigma \text{ and } \mathbf{I} \models \varphi\}$, is in $\mathcal{C}$. We say that a given Boolean query q of schema σ is in $\mathcal{C}$ if it can be tested with complexity $\mathcal{C}$, i.e., if the language $\{enc(\mathbf{I}) : \mathbf{I} \in \mathcal{B}_\sigma \text{ and } q(\mathbf{I}) = 1\}$ belongs to $\mathcal{C}$. We say that $\mathcal{L}$ *captures* $\mathcal{C}$ if the following holds:

i. The data complexity of $\mathcal{L}$ is in $\mathcal{C}$.

ii. For every Boolean query q in the complexity class $\mathcal{C}$, there is a sentence φ_q of $\mathcal{L}$ such that $\mathbf{I} \models \varphi_q$ iff $q(\mathbf{I}) = 1$.

The foundational result relating computational complexity to the expressive power of logics over finite structures, is due to Fagin [Fag74] who showed that the class NP contains exactly those properties (Boolean queries) that are expressible in existential second-order logic.

Theorem 2.10 ([Fag74])**.** Σ^1_1 *captures* NP.

This result was latter generalized by Stockmeyer [Sto76] who showed the correspondence between the prenex fragments of second-order logic and the polynomial-time hierarchy.

Theorem 2.11 ([Sto76])**.** *For* $m \geq 1$, Σ^1_m *captures* Σ^p_m.

Since then, the research in the area has demonstrated that virtually all known complexity classes can be mirrored in logic. However, some of the results equating logical expressibility to computational complexity require a built-in linear order. That is, the exact correspondence between expressibility in a logic and decidability within given resource bounds is restricted to those structures that have a linear order as one of their relations. For instance, LFP captures P on the class of ordered structures [Imm86, Var82], but no such logical characterization of the class P is known for arbitrary finite structures. Similarly, it is known that PFP captures PSPACE on the class of ordered structures [AV91a, Var82].

2.5 Background from Relational Complexity

The fact that the models of computation of queries do not assume an ordered domain, leads to a mismatch between the hardness of database queries and their Turing complexity. The typical example is the *even* query on a set, which has very low Turing complexity, but it is by all accounts a hard query. This query cannot be expressed in logics as powerful as $\mathcal{L}^{\omega}_{\infty\omega}$, which strictly includes all usual extensions of first-order logic with a fixed point operator. Recall that, it is only when we assume a build in *order* that IFP captures P and PFP caputres PSPACE.

Based on this observation, Abiteboul, Vardi and Vianu [AVV97] proposed a different notion to measure the complexity of computing queries. They called this notion, *relational complexity*. In relational complexity theory, computations over relational structures or databases are modelled by relational machines instead of Turing machines. The relational machine was originally called loosely coupled generic machine [AV91b, AV95]. Later on, it was renamed as relational machine [AVV95].

A *relational machine* is a Turing machine augmented with a finite set of fixed-arity relations forming a *relational store* (*rs*). Designated relations contain initially the input database, and one specific relation holds the output at the end of the computation. A relational machine uses a finite set of first-order formulae to interact with the *rs*. Transitions have the form:

If the internal state of the machine is q,
  the tape head is reading symbol x and
  the first-order sentence φ evaluates to true in the *rs*,
then change state to q',
    write symbol x' in the tape,
    move the tape head one cell to the left/right and
    replace the r-ary relation R by the relation defined by the first-order formula $\psi(x_1, \ldots, x_r)$ in the structure contained in the *rs*.

We give a formal definition of this machine in Chapter 7.

Each relational machine has an associated *arity*, which is the maximum number of variables which appear in any formula in its finite control.

Unlike Turing machines, relational machines have limited access to their input. Note that a k-ary relational machine can only access to its input through a fixed set of FO^k queries. In particular, relational machines cannot compute the size of their input structures in the general case. This is so because the discerning power of the relational machines is limited. More precisely, a relational machine of arity k cannot distinguish between tuples of elements of the input structure whose respective FO^k-types coincide.

Proposition 2.12 ([AVV97])**.** *Let $1 \leq r \leq k$. For every pair of r-tuples $\bar{a}$ and $\bar{b}$ over a relational structure $\boldsymbol{I}$, $\bar{a} \equiv^k \bar{b}$ iff no k-ary relational machine can distinguish among $\bar{a}$ and $\bar{b}$ over $\boldsymbol{I}$.*

Therefore, computations of k-ary relational machines are determined by the equivalence classes of the relation $\equiv^k$. In fact, as shown by Abiteboul and Vianu in [AV95], relational machines are complete on ordered input structures (where all distinct tuples have different FO^k-type), but they collapse to first-order logic on unordered sets (where the number of equivalence classes in $\equiv^k$ is bounded by a constant independently of the size of the input structure).

Since k-ary relational machines cannot distinguish between tuples which are $\equiv^k$-equivalent, they cannot compute the size of their input structures. But, they can compute the number of $\equiv^k$-classes.

Proposition 2.13 ([AVV97])**.** *Let the k-*size *of a structure $\boldsymbol{I}$, denoted* $\mathrm{size}_k(\boldsymbol{I})$*, be the number of $\equiv^k$-classes of k-tuples over $\boldsymbol{I}$. For each $k \geq 1$ and relational vocabulary σ, there is a deterministic relational machine M_σ of arity $2k$ that outputs on its Turing machine tape, for an input structure $\boldsymbol{I}$ of vocabulary σ, a string of length $size_k(\boldsymbol{I})$ in time polynomial in $size_k(\boldsymbol{I})$.*

Based on these facts, Abiteboul and Vianu proposed to used the k-size as a basis for measuring the complexity of relational machines. This is also the approach that we follow in Chapter 7.

We think of a relational machine M as an *acceptor* of a *relational language*, i.e., a class of structures of a relational vocabulary closed under isomorphisms. The relational language accepted by M, denoted $\mathcal{L}(M)$, is simply the set of input structures accepted by M. If M is deterministic, then the computation time of M on an input structure $\mathbf{I}$ is the number of transitions that M makes before accepting or rejecting $\mathbf{I}$, while the computation space is the number of tape cells scanned. If M is nondeterministic, then we only consider accepting computations. In that case, the computation time of M on an input structure $\mathbf{I}$ is the number of transitions in the shortest accepting computation of M on $\mathbf{I}$, while the computation space is the minimum number of tape cells scanned in any accepting computation of M on $\mathbf{I}$.

Definition 2.14. Let $\mathcal{L}(M)$ be the relational language accepted by a halting relational machine M of arity k. Let t and s be functions on the natural numbers such that $t(n) \geq n + 1$ and $s(n) \geq 1$. Then we say that:

- $\mathcal{L}(M) \in \mathrm{DTIME}_r(t(n))$ if M is deterministic and its computation time on any input structure $\mathbf{I}$ is bounded above by $t(\mathit{size}_k(\mathbf{I}))$;
- $\mathcal{L}(M) \in \mathrm{NTIME}_r(t(n))$ if M is nondeterministic and its computation time on any input structure $\mathbf{I}$ is bounded above by $t(\mathit{size}_k(\mathbf{I}))$;
- $\mathcal{L}(M) \in \mathrm{DSPACE}_r(s(n))$ if M is deterministic and its computation space on any input structure $\mathbf{I}$ is bounded above by $s(\mathit{size}_k(\mathbf{I}))$;
- $\mathcal{L}(M) \in \mathrm{NSPACE}_r(s(n))$ if M is nondeterministic and its computation space on any input structure $\mathbf{I}$ is bounded above by $s(\mathit{size}_k(\mathbf{I}))$.

Mirroring the classical complexity classes, we define the class P_r of the relational languages decidable by relational machines working in polynomial-time in the k-size of their input structures as $\mathrm{P}_r = \bigcup_{c \in \mathbb{N}} \mathrm{DTIME}_r(n^c)$; and the class NP_r of the relational languages decidable by nondeterministic relational machine working in polynomial-time in the k-size of their input structures as $\mathrm{NP}_r = \bigcup_{c \in \mathbb{N}} \mathrm{NTIME}_r(n^c)$. The class PSPACE_r of relational languages decidable by relational machines working in polynomial-space in the k-size of their input structures is $\bigcup_{c \in \mathbb{N}} \mathrm{DSPACE}_r(n^c)$.

A logic $\mathcal{L}$ *captures* a relational complexity class $\mathcal{C}_r$ iff every class of relational structures definable in $\mathcal{L}$ is in $\mathcal{C}_r$ and vice versa.

Abiteboul, Vardi and Vianu proved the following results relating fixed point logics over finite structures and relational complexity classes.

Theorem 2.15 ([AV95, AVV97])**.**

- IFP *captures* P_r*,*
- NFP *captures* NP_r*,*
- PFP *captures* PSPACE_r*.*

Interestingly, questions about containments among standard complexity classes translate to questions about containments among *relational* complexity classes.

Theorem 2.16 ([AVV97])**.** *Let* $\mathit{Class}(\mathit{Resource}, \mathit{Control}, \mathit{Bound})$ *and* $\mathit{Class}_r(\mathit{Resource}, \mathit{Control}, \mathit{Bound})$ *denote the classical complexity class and the relational complexity class, respectively, where Resource is either time or space, Control is either deterministic, nondeterministic, or alternating, and Bound is the bounding function or family of functions. Let* F_1 *and* F_2 *be polynomially closed sets of time/space constructible functions, and let* $\mathit{Resource}_1$*,* $\mathit{Resource}_2$*,* $\mathit{Control}_1$*,* $\mathit{Control}_2$ *be kinds of resources and controls, respectively. It holds that,*

$$\mathit{Class}(\mathit{Resource}_1, \mathit{Control}_1, F_1) \subseteq \mathit{Class}(\mathit{Resource}_2, \mathit{Control}_2, F_2)$$

if and only if

$$\mathit{Class}_r(\mathit{Resource}_1, \mathit{Control}_1, F_1) \subseteq \mathit{Class}_r(\mathit{Resource}_2, \mathit{Control}_2, F_2).$$

It follows that the known relationships between deterministic and nondeterministic complexity classes, also hold for relational complexity classes. For instance, the class of relational languages decidable by nondeterministic relational machines in polynomial space collapses to PSPACE_r. Also, open questions about standard complexity classes translate to questions about relational complexity classes. For example, $\mathrm{P} = \mathrm{NP}$ iff $\mathrm{P}_r = \mathrm{NP}_r$.

Note that, the well known Abiteboul-Vianu theorem, which reduces the problem of separating complexity classes P and PSPACE to separating the fixed point logics LFP and PFP over *unordered* structures, follows from Theorems 2.15 and 2.16.

Chapter 3

Higher-Order Logics

In this chapter we introduce the central subject of the thesis, namely higher-order logics over finite models.

Since it is not frequently found in the literature, we start with a detailed definition of the syntax and semantics of finite-order logic. Furthermore, the technicalities of the proofs of several of our results use heavily these definitions. We also introduce here the concept of *maximal-arity* (a generalization of the concept of arity to higher orders) used in the definition of the arity and alternation hierarchies studied in this work. At the end of the first section, we define the fragments of finite-order logic which are usually known as higher-order logics.

Then, we give a few examples of queries. The intention is to provide some intuition on how to express queries in higher-order logic. We also discuss briefly why we believe that the study of higher-order logics over finite models could be of interest from an application perspective.

The last two sections of the chapter are devoted to the review of known results regarding the expressive power of higher-order logics over finite structures. First we introduce the general results which characterize the expressive power of the prenex fragments of higher-order logics. Then we introduce some result on hierarchies in higher-order logics –a central topic in this work–. The fact that these results refer only to second-order logic, seems to be mainly the consequence of the fact that this is the fragment of finite-order logic which has received, up to know, most of the attention in the context of finite model theory.

3.1 Syntax and Semantics of Finite-Order Logic

Finite-order logic is an extension of first-order logic which allows us to quantify over higher-order relations. We define here its syntax and semantics following the account in [Lei94]. We emphasize the fact that the set of formulae of finite-order logic can be viewed as a set of strings over a *finite* alphabet, i.e., as a formal language. This plays an important role in the encoding of formulae as finite structures which we define in Section 4.2 in the next chapter.

Definition 3.1. We define the set of *types*, as the set **typ** of strings over the alphabet $\{\iota; (;); ,\}$ inductively generated by:

- $\iota \in \mathbf{typ}$;
- if $\tau_1, \ldots, \tau_r \in \mathbf{typ}$ $(r \geq 1)$, then $(\tau_1, \ldots, \tau_r) \in \mathbf{typ}$;
- nothing else belongs to **typ**.

If $\tau_1 = \cdots = \tau_r = \iota$, then $(\tau_1, \ldots, \tau_r)$ is denoted by ι^r. The set of types can be naturally stratified into *orders* which are inductively defined, as follows:

- $\mathit{order}(\iota) = 1$
- $\mathit{order}((\tau_1, \ldots, \tau_r)) = 1 + \mathit{max}(\{\mathit{order}(\tau_1), \ldots, \mathit{order}(\tau_r)\})$

For $\tau = (\tau_1, \ldots, \tau_r)$, r is the *arity* of the type τ. We associate a non-negative integer, the *maximal-arity* (*ma*), with each type, as follows:

- $\mathit{ma}(\iota) = 0$
- $\mathit{ma}((\tau_1, \ldots, \tau_r)) = \mathit{max}(\{r, \mathit{ma}(\tau_1), \ldots, \mathit{ma}(\tau_r)\})$

Clearly, if $\mathit{order}(\tau) = 2$, then the maximal-arity of τ coincides with its arity. We denote as $\mathit{typ}(i, r)$ the subset of types of order $\leq i$ and maximal-arity $\leq r$. Note that each subset $\mathit{typ}(i, r)$ is finite. For instance,

$$\mathit{typ}(3, 2) = \{\iota; (\iota); (\iota, \iota); ((\iota)); ((\iota, \iota)); (\iota, (\iota)); ((\iota), \iota); ((\iota), (\iota)); (\iota, (\iota, \iota));$$

$$((\iota, \iota), \iota); ((\iota), (\iota, \iota)); ((\iota, \iota), (\iota)); ((\iota, \iota), (\iota, \iota))\}.$$

The intended interpretation is that objects of type ι are individuals, i.e., elements of the universe of a given model, whereas objects of type $(\tau_1, \ldots, \tau_r)$ are r-ary relations, i.e., sets of r-tuples of objects of types $\tau_1, \ldots, \tau_r$, respectively.

Definition 3.2. Given a set U, the set U_τ of *objects* of type τ over U is defined by

$$\begin{aligned} U_\iota &= U \\ U_{(\tau_1,\ldots,\tau_r)} &= \mathcal{P}(\prod_{i=1}^{r} U_{\tau_i}) = \mathcal{P}(U_{\tau_1} \times \cdots \times U_{\tau_r}) \end{aligned}$$

Over a relational vocabulary σ, each formula of finite-order logic is a string of symbols taken from the *alphabet*

$$A = \{\neg; \vee; \wedge; \exists; \forall; (;); =; x; X; |; \iota; ,\} \cup \sigma$$

The words that belong to the language $\{x|^n : n > 0\}$ are called *individual variables*, while the words that belong to the language $\{X\tau|^n : \tau \in \mathbf{typ} \setminus \{\iota\} \text{ and } n > 0\}$ are called *higher-order variables*. We call the higher-order variables of the form $X\tau|^n$, for $i = \mathit{order}(\tau)$ and $r = \mathit{ma}(\tau)$, *i-th order variables* of *maximal-arity* r. To simplify the notation we denote strings of the form $|^n$, $n > 0$, as subscripts, e.g., we write x_3 for $x|||$. In addition, we write the types of the higher-order variables as superscripts, e.g., we write $X_2^{(\iota)}$ for $X(\iota)||$. Sometimes, we omit the superscript when we denote second-order variables (i.e., variables of type ι^r, for some $r \geq 1$) if their arity is clear from the context. We use V^τ to denote any variable of type τ. So, if $\tau = \iota$ then V^τ stands for an individual variable, otherwise V^τ stands for a higher-order variable of type τ.

Definition 3.3. We define the set of *well-formed formulae* of finite-order logic over a relational vocabulary σ (here we do not allow constant symbols), as follows:

i. If v_1 and v_2 are individual variables, then $v_1 = v_2$ is a wff.

ii. If R is a relation symbol in σ of arity $r \geq 1$, and $v_1, \ldots, v_r$ are individual variables, then $R(v_1, \ldots, v_r)$ is a wff.

iii. If V^τ is a higher-order variable of type $\tau = (\tau_1, \ldots, \tau_r)$ and $V_1^{\tau_1}, \ldots, V_r^{\tau_r}$ are variables of types $\tau_1, \ldots, \tau_r$, respectively, then $V^\tau(V_1^{\tau_1}, \ldots, V_r^{\tau_r})$ is a wff.

iv. If φ is a wff, then $(\neg\varphi)$ is a wff.

v. If φ and ψ are wff's, then $(\varphi \vee \psi)$ and $(\varphi \wedge \psi)$ are wff's.

vi. If φ is a wff and v is an individual variable, then $\exists v(\varphi)$ and $\forall v(\varphi)$ are wff's.

vii. If φ is a wff and V^{τ} is a higher-order variable, then $\exists V^{\tau}(\varphi)$ and $\forall V^{\tau}(\varphi)$ are wff's.

viii. Nothing else is a wff.

The *atomic formulae* are the ones introduced by clauses (i) to (iii). The free occurrence of a variable (either an individual variable or a higher-order variable) in a formula of finite-order logic is defined in the obvious way. Thus, the set *free*(φ) of *free variables* of a formula φ is the set of both individual and higher-order variables which do not occur in φ under the scope of a quantifier which binds them.

Note that, according to our definition of wff of finite-order logic, atomic formulae of the form $V^{(\ldots,\iota^{r},\ldots)}(\ldots,R,\ldots)$, where R is a relation symbol of arity r in the given vocabulary σ, are not allowed. We choose to impose this restriction only for the sake of clarity, in particular for the presentation of Section 4.5. This is clearly not a fundamental restriction in finite-order logic, and furthermore our results can also be proven without this restriction as shown in [FT05].

The *semantics* of formulae of finite-order logic is similar to the semantics of formulae of second-order logic, except that a valuation over a structure with universe U maps higher-order variables of type τ to objects in U_{τ}.

Definition 3.4. Let σ be a relational vocabulary. A *valuation val* on a σ-structure $\mathbf{I}$ with domain I, is a function which assigns to each individual variable an element in I and to each higher-order variable V^{τ}, for some type $\tau \neq \iota$, an object in I_{τ}. Let val_0, val_1 be two valuations on a σ-structure $\mathbf{I}$, we say that val_0 and val_1 are V^{τ}*-equivalent* if they coincide in every variable of whichever type, with the possible exception of variable V^{τ}. We also use the notion of equivalence w.r.t. sets of variables.

Let $\mathbf{I}$ be a σ-structure, and let *val* be a valuation on $\mathbf{I}$. The notion of *satisfaction* in finite-order logic extends the notion of satisfaction in first-order logic with the following rules:

i. $\mathbf{I}, val \models V^{\tau}(V_1^{\tau_1},\ldots,V_r^{\tau_r})$ where $\tau = (\tau_1,\ldots,\tau_r)$ iff $(val(V_1^{\tau_1}),\ldots,val(V_r^{\tau_r})) \in val(V^{\tau})$.

ii. $\mathbf{I}, val \models \exists V^{\tau}(\varphi)$ where V^{τ} is a higher-order variable and φ is a well-formed formula, iff there is a valuation val', which is V^{τ}-equivalent to val, such that $\mathbf{I}, val' \models \varphi$.

iii. $\mathbf{I}, val \models \forall V^\tau(\varphi)$ where V^τ is a higher-order variable and φ is a well-formed formula, iff for every valuation val', which is V^τ-equivalent to val, it holds that $\mathbf{I}, val' \models \varphi$.

The restriction of finite-order logic to formulae whose variables are all of order $\leq i$, for some $i \geq 1$, is called *i-th order logic* and is denoted by HO^i. Note that, for $i = 1$ this is first-order logic (FO), and for $i = 2$ this is second-order logic (SO). The logics of order $i \geq 2$ are usually known as *higher-order logics* (HO).

As in many other extensions of first-order logic, in second-order logic we can naturally associate a non-negative integer, the *arity*, with each formula. Usually, the arity of a formula of second-order logic is defined as the biggest arity of a second-order variable occurring in that formula. Taking a similar approach, we define the *maximal-arity* of a HO^i formula, $i \geq 2$, as the biggest maximal-arity of any higher-order variable occurring in that formula. For $r \geq 1$, the restriction of HO^i to formulae of maximal-arity $\leq r$ forms the fragment $\mathrm{HO}^{i,r}$ of HO^i. Clearly, for second-order logic the maximal-arity of a formula coincides with its arity. Note that, if a variable V^τ occurs in some $\mathrm{HO}^{i,r}$ formulae, then $\tau \in typ(i, r)$.

An easy induction using renaming of variables and equivalences such as $\neg\exists V^\tau(\varphi) \equiv \forall V^\tau(\neg\varphi)$ and $(\phi \vee \forall V^\tau(\psi)) \equiv \forall V^\tau(\phi \vee \psi)$ if V^τ is not free in ϕ, shows that each HO^i formula is logically equivalent to an HO^i formula in *prenex normal form*, i.e., to a formula of the form $Q_1V_1 \ldots Q_nV_n(\varphi)$, where $Q_1, \ldots, Q_n \in \{\forall, \exists\}$, and where $V_1, \ldots, V_n$ are variables of order $\leq i$ and φ is a quantifier-free HO^i formula. Moreover, for every $i \geq 2$, each HO^i formula is logically equivalent to one in prenex normal form in which the quantifiers of order i precede all the remaining quantifiers in the prefix (among others, see [HT03, HT06b] for a detailed proof of this fact). Such normal form is known as *generalized Skolem normal form*, or GSNF. The formulae of finite-order logic which are in GSNF comprise well known hierarchies whose levels are denoted Σ^i_m and Π^i_m. The class Σ^i_m consists of those HO^{i+1} formulae in GSNF in which the quantifiers of order $i + 1$ are arranged into at most m alternating blocks, starting with an existential block. Π^i_m is defined dually. Clearly, every HO^{i+1} formula is equivalent to a Σ^i_m formula for some m, and also to a Π^i_m formula.

3.2 Examples of Queries in Higher-Order Logics

It is not frequent to find in the literature examples of queries expressed in higher-order logics beyond second-order. So, we think it is appropriate to provide here some intuition on how to describe properties of finite structures using higher-order quantification. We do that by means of two simple examples of properties of graphs which allow for an elegant characterization in third-order logic. Further examples can be found in [FT04, FPT05].

In the examples below, we work on the vocabulary $\sigma = \{E\}$ of graphs. An *undirected graph* is a σ-structure $\mathbf{G} = \langle G, E^{\mathbf{G}} \rangle$ satisfying $\varphi_1 \equiv \forall xy(E(x,y) \rightarrow E(y,x))$ and $\varphi_2 \equiv \forall x(\neg E(x,x))$. If we do not require $\mathbf{G}$ to satisfy φ_1, then we speak of a *directed graph* (or *digraph*). As usual, we call *vertices* the elements of G and *edges* the elements of $E^{\mathbf{G}}$.

For the sake of clarity, we omit in our examples the types of the variables (which anyway are clear from the context) and use uppercase calligraphic letters to denote third-order variables.

Examples of second-order formulae expressing different properties of graphs, are frequently encountered in the literature. A classical example of that is graph 3-colorability, which can be expressed elegantly in Σ^1_1 using just unary second-order variables (see for instance [Imm99]). Anyway, we think it is better to start with a simple example of a second-order query, rather than to go straight to the examples of third-order queries.

Example 3.5. An undirected graph $\mathbf{G}$ is *regular* if all its vertices have the same degree. It is well known that the class of regular graphs is not definable in first-order logic [EF99, Imm99]. In second-order logic, this class can be defined as follows:

$$\exists A\Big(\forall x\big(\exists B(\text{“}B \text{ is the set of vertices which are adjacent to } x\text{”} \wedge \\ \text{“the sets } A \text{ and } B \text{ have the same cardinality”}\ )\big)\Big)$$

It is very easy to express that “B is the set of vertices which are adjacent to x”.

$$\forall z\big(B(z) \leftrightarrow E(x,z)\big)$$

Finally, we can express that “the sets A and B have the same cardinality” with a formula stating that there is a bijection F from A to B.

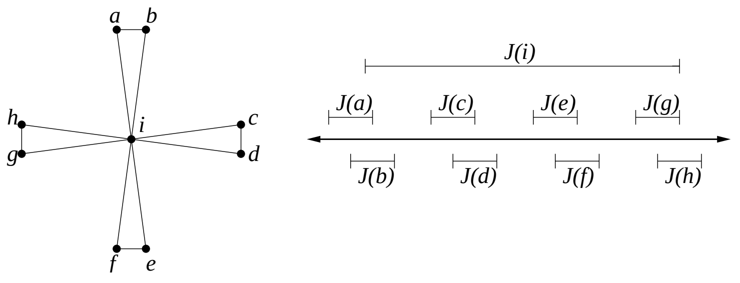

Figure 3.6

$$\begin{aligned}
\exists F \forall xyz \Big(&\big(F(x,y) \rightarrow A(x) \wedge B(y)\big) \wedge && \text{“}F \text{ is a subset of } A \times B\text{”} \\
&\big(F(x,y) \wedge F(x,z) \rightarrow y = z\big) \wedge && \text{“}F \text{ is a function”} \\
&\big(A(x) \rightarrow \exists y(F(x,y))\big) \wedge && \text{“}F \text{ is total”} \\
&\big(F(x,z) \wedge F(y,z) \rightarrow x = y\big) \wedge && \text{“}F \text{ is injective”} \\
&\big(B(y) \rightarrow \exists x(F(x,y))\big)\Big) && \text{“}F \text{ is surjective”}
\end{aligned}$$

Now we can move to our first example in third-order logic.

Example 3.6. An undirected graph **G** is an *interval graph* iff it is the intersection graph of a family of intervals of the real line. This means that to each vertex v of **G** corresponds an interval $J(v)$ of the real line such that $(u,v) \in E^{\mathbf{G}}$ iff $J(u) \cap J(v) \neq \emptyset$. Figure 3.6 displays the graph usually known as windmill together with its interval representation.

By using a characterization of Gilmore and Hoffman [GH64], we can define in third-order logic the class of interval graphs, as follows:

$$\begin{aligned}
&\exists \mathcal{C} \exists \mathcal{O} \big(\text{“}\mathcal{C} \text{ is the family of all vertex subsets of } \mathbf{G} \text{ that induce a maximal clique”} \wedge \\
&\quad \text{“}\mathcal{O} \text{ is a linear order of } \mathcal{C}\text{”} \wedge \\
&\quad \forall x(\text{“the subsets in } \mathcal{C} \text{ which contain } x \text{ occur consecutively in the linear order } \mathcal{O}\text{”})\big)
\end{aligned}$$

We express that “$\mathcal{C}$ is the family of all vertex subsets of **G** that induce a maximal clique” with the subformula:

$$\begin{aligned}
\forall X \Big(\mathcal{C}(X) \leftrightarrow \forall x \forall y \big(X(x) \wedge X(y) \wedge x \neq y \rightarrow E(x,y)\big) \wedge \quad &\text{“}X \text{ induces a clique on } \mathbf{G}\text{”} \\
\neg \exists Y \big(\forall x(X(x) \rightarrow Y(x)) \wedge \exists x(Y(x) \wedge \neg X(x)) \wedge & \\
\forall x \forall y (Y(x) \wedge Y(y) \wedge x \neq y \rightarrow E(x,y))\big)\Big) & \\
&\text{“no superset of } X \text{ induces a clique on } \mathbf{G}\text{”}
\end{aligned}$$

We express that "$\mathcal{O}$ is a linear order of $\mathcal{C}$" with the subformula:

$$\begin{aligned}
&\forall X \forall Y \forall Z \big((\mathcal{O}(X,Y) \rightarrow \mathcal{C}(X) \wedge \mathcal{C}(Y)) \wedge && \text{"}\mathcal{O}\text{ is a subset of }\mathcal{C} \times \mathcal{C}\text{"} \\
&\qquad (\mathcal{C}(X) \rightarrow \mathcal{O}(X,X)) \wedge && \text{"}\mathcal{O}\text{ is reflexive"} \\
&\qquad (\mathcal{O}(X,Y) \wedge \mathcal{O}(Y,X) \rightarrow X = Y) \wedge && \text{"}\mathcal{O}\text{ is antisymmetric"} \\
&\qquad (\mathcal{O}(X,Y) \wedge \mathcal{O}(Y,Z) \rightarrow \mathcal{O}(X,Z)) \wedge && \text{"}\mathcal{O}\text{ is transitive"} \\
&\qquad (\mathcal{C}(X) \wedge \mathcal{C}(Y) \rightarrow \mathcal{O}(X,Y) \vee \mathcal{O}(Y,X)) \big) && \text{"dichotomy holds in }\mathcal{O}\text{"}
\end{aligned}$$

Finally, we express that "the subsets in $\mathcal{C}$ which contain x occur consecutively in the linear order $\mathcal{O}$" with the subformula:

$$\begin{aligned}
&\exists F \exists L \Big(\mathcal{C}(F) \wedge F(x) \wedge \neg \exists X \big(\mathcal{O}(X,F) \wedge X \neq F \wedge X(x) \big) \wedge \\
&\qquad\qquad \text{"}F\text{ is the first subset in the order }\mathcal{O}\text{ which includes the vertex }x\text{"} \\
&\quad \mathcal{C}(L) \wedge L(x) \wedge \neg \exists X \big(\mathcal{O}(L,X) \wedge X \neq L \wedge X(x) \big) \wedge \\
&\qquad\qquad \text{"}L\text{ is the last subset in the order }\mathcal{O}\text{ which includes the vertex }x\text{"} \\
&\quad \forall X \big(\mathcal{O}(F,X) \wedge \mathcal{O}(X,L) \rightarrow X(x) \big) \Big) \\
&\quad \text{"the vertex }x\text{ belongs to all subsets which are between }F\text{ and }L\text{ in the order }\mathcal{O}\text{"}
\end{aligned}$$

Actually, we do not really need the expressive power of third-order logic to characterize the class of interval graphs. In fact, interval graphs can be recognized in linear time [BL76], and thus there is a formula in Σ^1_1 which can express this property. Recall that by Fagin's theorem, Σ^1_1 captures NP. However, it is very unlikely that there is a formula in second-order logic that expresses the property in a way which is as intuitive and simple as in the example. Similar observations can be applied to our next example.

As an aside consideration and from an application perspective, this indicates that it could make sense to add some kind of higher-order quantification to database query languages like SQL. Even though usually the queries in P are enough, as to the queries needed in the industry, we think that the use of higher-order quantification makes the expression of some of them much easier.

Example 3.7. An *n-cube* $\mathbf{Q}_n$ is an undirected graph whose vertices are binary n-tuples. Two vertices of $\mathbf{Q}_n$ are adjacent iff they differ in exactly one bit. A 1-cube $\mathbf{Q}_1$, a 2-cube $\mathbf{Q}_2$ and a 3-cube $\mathbf{Q}_3$ are displayed in Figure 3.7.

We can build an $(n+1)$-cube $\mathbf{Q}_{n+1}$ starting with two isomorphic copies of an n-cube $\mathbf{Q}_n$ and adding edges between corresponding vertices. Using this fact, we can define in third-order logic the so called class of hypercube graphs, as follows:

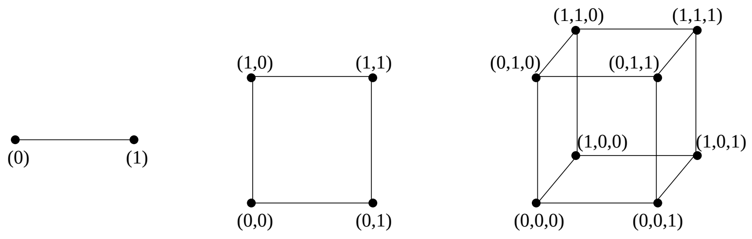

Figure 3.7

$\exists \mathcal{C} \exists \mathcal{O} \big($"$\mathcal{C}$ is a class of undirected graphs" $\wedge$ "$\mathcal{O}$ is a total order on $\mathcal{C}$" $\wedge$

$\quad \forall G_1 \forall G_2 \big(\mathcal{C}(G_1) \wedge \mathcal{C}(G_2) \wedge$

$\qquad$ "G_1 is the immediate predecessor of G_2 in the order $\mathcal{O}$" $\rightarrow$

$\qquad$ "G_2 can be built from two isomorphic copies of G_1 by adding edges between corresponding vertices" $\big) \wedge$

$\quad$ "the first graph in the order $\mathcal{O}$ is a Q_1"

$\quad$ "the last graph in the order $\mathcal{O}$ is the input graph" $\big)$

We can express that "G_2 can be built from two isomorphic copies of G_1 by adding edges between corresponding vertices", as follows:

$\exists F_1 \exists F_2 \big($"$F_1$ and F_2 are injective and total functions from $dom(G_1)$ to $dom(G_2)$" $\wedge$

"The ranges of F_1 and F_2 form a partition of $dom(G_2)$" $\wedge$

"F_1 and F_2 are isomorphisms from G_1 to the sub-graphs of G_2 induced by the ranges of F_1 and F_2, respectively" $\wedge$

$\forall x(x \in dom(G_1) \rightarrow$ "there is an edge in G_2 which connects $F_1(x)$ and $F_2(x)$") $\wedge$

$\neg \exists xy(x, y \in dom(G_1) \wedge x \neq y \wedge$

$\quad$ "there is an edge in G_2 which connects $F_1(x)$ and $F_2(y)$")$\big)$

Note that, if there is an edge (a, b) in G_2 such that a belongs to the range of F_1 and b belongs to the range of F_2, or vice versa, then either $F_1^{-1}(a) = F_2^{-1}(b)$ or $F_1^{-1}(b) = F_2^{-1}(a)$.

It is clear how to express in third-order logic the remaining details. So, we end this example here.

3.3 The Expressive Power of Higher-Order Logics

The known results on the expressive power of higher-order logics turned out to be extensions of the Fagin-Stockmeyer correspondence between the prenex fragments of second-order and the polynomial hierarchy. In order to state them, we first define by induction on i the i-fold exponential function $\exp_i$ on the natural numbers as follows: $\exp_0(n) = n$, and for $i \geq 1$, $\exp_i(n) = 2^{\exp_{i-1}(n)}$.

The expressive power of higher-order logics on finite structures has been studied, among a few others, by Kuper and Vardi [KV88], by Leivant [Lei89] and by Hull and Su [HS91]. However, the *exact* characterization of each prenex fragment of higher-order logics over finite structures is more recent and it is due to Hella and Turull Torres [HT03, HT06b].

Theorem 3.8 ([HT03, HT06b]). *For $i, m \geq 1$, it holds that:*

- Σ^i_m *captures* $\left(\bigcup_{c\in\mathbb{N}} \mathrm{NTIME}(\exp_{i-1}(n^c))\right)^{\Sigma^p_{m-1}}$
- Π^i_m *captures* $\left(\bigcup_{c\in\mathbb{N}} \mathrm{coNTIME}(\exp_{i-1}(n^c))\right)^{\Sigma^p_{m-1}}$

where Σ^p_{m-1} is the level $m-1$ of the polynomial-time hierarchy.

The following corollary is a direct consequence of the fact that, for every $i \geq 2$, $\mathrm{HO}^i = \bigcup_{m\geq 0} \Sigma^{i-1}_m$.

Corollary 3.9 ([HT03, HT06b]). *For every $i \geq 2$,*

- HO^i *captures* $\bigcup_{m\geq 0}\left(\bigcup_{c\in\mathbb{N}} \mathrm{NTIME}(\exp_{i-2}(n^c))\right)^{\Sigma^p_m}$

Independently, Kolodziejczyk [Kol04b, Kol05] characterized the prenex fragments of higher-order logic in terms of alternating Turing machines, but taking also into account the arity of the higher-order variables. He uses for that a notion of arity which he called *basic-arity* (*ba*). If τ is a type of order 2, then $ba(\tau)$ is simply the arity of τ. If $order(\tau) > 2$ and $\tau = (\tau_1, \ldots, \tau_r)$, then $ba(\tau) = max(\{ba(\tau_i) : order(\tau_i) = order(\tau) - 1, 1 \leq i \leq r\})$. For $i \geq 2$ and $m \geq 1$, they defined $[\Sigma^i_m]^{\leq r}$ as the class of Σ^i_m formulae in which the type of the variables of order i and $i+1$ have basic-arity bounded by r, and proved the following.

Theorem 3.10 ([Kol04b, Kol05]). *For $i \geq 2$ and $m \geq 1$,*

- $[\Sigma^i_m]^{\leq r}$ *captures* $\bigcup_{c\in\mathbb{N}} \Sigma_m\mathrm{TIME}(\exp_{i-1}(cn^r))$

A characterization of the expressive power of the Σ^i_m and Π^i_m fragments in terms of alternating Turing machines instead of oracle machines, follows easily from the previous theorem. According to [Bör84], the case $m = 1$ was proved by Christen in his Ph.D. thesis [Chr74].

Corollary 3.11. *For $i, m \geq 1$,*

- Σ^i_m *captures* $\bigcup_{c \in \mathbb{N}} \Sigma_m \mathrm{TIME}(\exp_{i-1}(n^c))$
- Π^i_m *captures* $\bigcup_{c \in \mathbb{N}} \Pi_m \mathrm{TIME}(\exp_{i-1}(n^c))$

Previously known results from Hull and Su [HS91] and from Kuper and Vardi [KV88] regarding the expressive power of higher-order logics, can be obtained as corollaries of Theorem 3.8 and Corollary 3.9.

The class of Kalmar elementary functions can be characterized as the class of functions on natural numbers which are computable using resources bounded by $\exp_i(n^c)$ for some $i \geq 0$ and some constant $c \in \mathbb{N}$ (see [Ros84]). Observe that, by the standard relationships that hold among the different resources (see Theorem 2.26 in [BDG95]), it does not matter whether we take resources to mean DTIME, NTIME, DSPACE or NSPACE.

Corollary 3.12 ([Ben62, HS91])**.** *Finite-order logic captures the class of languages decidable in* $\mathrm{DTIME}(f(n))$*, where $f(n)$ is a Kalmar elementary function.*

Consequently, finite-order logic (or equivalently $\bigcup_{i \geq 2} HO^i$) is not complete, in the sense that it cannot express all computable Boolean queries. In [HT03, HT06b] a logic which is *complete*, called *variable order logic* (VO), was defined. This logic permits the use of untyped relation variables, i.e., variables of variable order, by allowing quantification over orders. They showed that the resulting logic can express all computable queries, but it can also express non computable queries. A characterization of the class of computable queries in terms of an undecidable fragment of VO was also given there.

As pointed out by Hella and Turull-Torres, the following result is a straightforward consequence of their characterization of HO^i and the hierarchy theorem for nondeterministic time (see [Coo72, SFM78, Zák83]). The original proof in [HS91] is by contrast quite involved.

Corollary 3.13 ([HS91])**.** *For every $i \geq 2$, $\mathrm{HO}^i \subset \mathrm{HO}^{i+1}$.*

A similar result also follows for the existential and universal fragments of higher-order logics. We denote as $\exists\mathrm{HO}^i$ the class of HO^i formulae in which universal quantification of order i is not allowed and where negation is allowed only in atoms. To denote the corresponding universal fragment, we use $\forall\mathrm{HO}^i$.

Corollary 3.14 ([KV88]). *For every $i \geq 2$,*

- $\exists\mathrm{HO}^i \subset \exists\mathrm{HO}^{i+1}$
- $\forall\mathrm{HO}^i \subset \forall\mathrm{HO}^{i+1}$

3.4 Known Hierarchies in Second-Order Logic

In second-order logic, a considerable amount of effort was devoted to the study of hierarchies defined in terms of *alternations* of quantifiers. In this line of work, important results have been obtained for *monadic second-order logic* (MSO), i.e., second-order logic restricted to unary second-order variables.

Some of the earlier results on this regard concern a class of special structures which are usually known a *word models* [EF99]. Let A be a finite alphabet and let $\pi(A)$ be the vocabulary $\{<\} \cup \{R_a : a \in A\}$, where $<$ is a binary relation symbol and the different R_a are unary relation symbols. We can identify any word $w = a_1 \ldots a_n$ in A^+ with a $\pi(A)$-structure (word model) $\mathbf{I}_w$, where the cardinality of $dom(\mathbf{I}_w)$ equals the length of w, $<^{\mathbf{I}_w}$ is a linear order in $dom(\mathbf{I}_w)$, and, for each $R_a \in \pi(A)$, $R_a^{\mathbf{I}_w}$ contains the positions in w carrying an a, i.e., $R_a^{\mathbf{I}_w} = \{b \in I_w :$ for some j $(1 \leq j \leq n)$, b is the j-th element in the order $<^{\mathbf{I}_w}$ and $a_j = a\}$.

By the fundamental result of Büchi and Elgot the regular languages are precisely those which can be defined in monadic second-order on word models, cf. [Büc60, Elg61].

Theorem 3.15 ([Büc60]). *Let A be a finite alphabet. A language $L \subseteq A^+$ is definable in monadic second-order logic iff it is regular.*

Furthermore, since every regular language is actually definable by a monadic Σ^1_1 sentence (see for instance Proposition 6.2.1 in [EF99]), it follows that on word models, monadic second-order logic collapses to its existential fragment.

Corollary 3.16. *Over word models,* MSO $=$ *monadic* Σ^1_1.

Thomas [Tho82] gave a refinement of this result. He observed that over word models monadic-second order logic even collapses to monadic Σ^1_1 with only a single existential quantification of a unary second-order variable. This result does not extend to all finite structures. On the contrary, Otto [Ott95] showed that the number of existential quantifiers in monadic Σ^1_1 induces a strict hierarchy as to expressive power. Let k-monΣ^1_1 be the set of monadic Σ^1_1 formulae which allow up to k existential quantifications of unary second-order variables.

Theorem 3.17 ([Ott95])**.** *For every $k \geq 1$, there are Boolean queries not expressible in k-monΣ^1_1 but expressible in $(k+1)$-monΣ^1_1.*

Other classes of finite structures over which interesting results concerning monadic second-order hierarchies have been obtained, are the classes of graphs and grids. We already defined the class of directed and undirected graphs in Section 3.2. A *grid* is a graph isomorphic to

$$\langle\{0,\dots,n\}\times\{0,\dots,m\},\{((i,j),(k,l)) : i,k\leq n; j,l\leq m; |i-k|+|j-l|=1\}\rangle$$

for some n, m. That is, for $0 \leq i < n$ and $0 \leq j < n$, each node (i,j) is connected to the nodes $(i+1,j)$ and $(i,j+1)$.

In 1994, Fagin [Fag94] raised the question of whether by increasing m the monadic Σ^1_m-formulae could express more properties of finite graphs. This question is the "monadic analogue" of the problem whether the polynomial hierarchy is strict. Some years before that, Ajtai and Fagin [AF90] gave a partial answer to this question proving that reachability for directed graphs is a monadic property which is not expressible in monadic Σ^1_1. Schwentick [Sch94] extended this result to directed graphs with a built-in order. Fagin's question was answered in [MT97] where it was shown that over directed graphs, the monadic Σ^1_m-formulae induce a strict hierarchy of graph properties. A slightly stronger result was proven for grids in [Sch97], where it was shown that over grids, monadic$-\Sigma^1_m \subset$ monadic$-\Sigma^1_{m+1} \cap$ monadic$-\Pi^1_{m+1}$. These results on monadic quantifier hierarchies over grids and graphs were further studied and strengthened in [MST02], where among other interesting results, the following was shown.

Theorem 3.18 ([MST02])**.** *Let* $\mathrm{B}(\text{monadic}-\Sigma^1_m)$ *denote the smallest superset of monadic Σ^1_m which is closed under conjunction and negation of formulae in monadic Σ^1_m. Let $m \geq 1$. If $\mathcal{C}$ is the class of directed graphs, or undirected graphs, then it holds that over $\mathcal{C}$,* $\mathrm{B}(\text{monadic}-\Sigma^1_m) \subset \text{monadic}-\Sigma^1_{m+1} \cap \text{monadic}-\Pi^1_{m+1}$.

We should note here that this result does not hold for *colored grids*, i.e., grids extended with unary relations which may be thought of as representing colors on the vertices. Over colored grids, the collapse of existential monadic second-order prefixes to a single existential quantifier was proved in [Mat98]. Also, several classes of graphs of bounded tree-width in which the monadic quantifier hierarchy collapses, have been characterized [MM03].

In many extensions of first-order logic, the maximum *arity* of the relation variables occurring in a formula was shown to be of a great relevance. The fragments allowing only formulae of a bounded arity in its relation variables form a natural hierarchy inside such logics, and a natural question to be asked is whether this hierarchy is strict. An affirmative answer to this question for various extensions of first-order logic by fixed-point operators and transitive closure operators has been given by Grohe in [Gro93, Gro96]. In [Hel89, Hel92], Hella studied the notion of arity on first-order logic extended with Lindström quantifiers. In [GH96], a double arity hierarchy theorem for transitive closure logic was proven. However, in second-order logic less progress has been made regarding arity hierarchies.

The Σ^1_1 arity hierarchy, i.e., the hierarchy $\cup_{r \in \mathbb{N}} (\Sigma^1_1)^{\leq r}$ where $(\Sigma^1_1)^{\leq r}$ denotes the set of Σ^1_1 formulae restricted to second-order variables of arity at most r, is known to be strict over vocabularies of arbitrary (unbounded) arity [Ajt83]. But it is still open whether the arity hierarchy in Σ^1_1 is strict over vocabularies of a fixed arity. A sufficient condition for the strictness of this hierarchy was given in [Kol04a].

Proposition 3.19 ([Kol04a])**.** *If for every vocabulary σ there is a fixed k such that data complexity for* $\mathrm{FO}[\sigma]$ *is in* $\mathrm{NTIME}(n^k)$*, then* $(\Sigma^1_1)^{\leq r} \subset (\Sigma^1_1)^{\leq r+1}$ *for every* $r \geq 1$.

In the study of the full hierarchy Σ^1_m, Makowsky and Pnueli followed a different approach in [MP96]. They investigated the expressive power of second-order logic over finite structures when limitations in the arity of the second-order variables and in the number of alternations of both first-order and second-order quantifiers, are simultaneously imposed. Under these conditions, they proved the existence of a proper hierarchy of arity and alternation in second-order logic. We explain this result with more detail in the next chapter where we prove analogous results for each higher-order logic of order ≥ 2.

Finally, we should mention that an important hierarchy theorem for second-order generalized quantifiers was recently proved in [Kon06].

Chapter 4

An Arity-Alternation Hierarchy in Higher-Order Logics

In this chapter, aiming to gain a better understanding on the kind of syntactic restrictions which are relevant as to the expressive power of different fragments of higher-order logics over finite structures, we study the effect of simultaneously bounding the maximal-arity of the higher-order variables and the alternation of quantifiers in formulae of higher-order logics. Let $AA^i(r,m)$ be the class of $(i+1)$-th order logic formulae where all quantifiers of *whichever* order are grouped together at the beginning of the formula, forming up to m alternating blocks of consecutive existential and universal quantifiers, and such that the maximal-arity of the higher-order variables is bounded by r. Note that, the order of the quantifiers in the prefix may be mixed.

We show that, for every $i \geq 1$, the resulting $AA^i(r,m)$ hierarchy of formulae of $(i+1)$-th order logic is proper. We get our result by roughly adapting the strategy of Makowsky and Pnueli [MP96] to each higher-order logic of order $i \geq 2$. The strategy consists in considering the set $AUTOSAT(F)$ of formulae of a given logic F which, encoded as finite structures, satisfy themselves. As the well known diagonalization argument applies when F is a level of an AA^i hierarchy of arity and alternation, it follows that $AUTOSAT(F)$ is not definable in F, but is definable in a higher level of the same hierarchy.

The chapter is organized as follows. In the first section we formally define the AA^i hierarchies. Then, in Section 4.2 we fix an encoding for the formulae of finite-order logic as relational structures, and we define the sets $WFF(F)$, $AUTOSAT(F)$ and

$DIAG(F)$ of structures encoding well-formed formulae in F, self-satisfying formulae in F, and well-formed formulae in the complement of $AUTOSAT(F)$ with respect to $WFF(F)$, respectively, for a given logic F. In Section 4.3 we study the descriptive complexity of $WFF(F)$ for different fragments of higher-order logics. After that, we move to Section 4.4 where we use a diagonalization argument to give lower bounds for the definability of $DIAG(F)$ and $AUTOSAT(F)$ for different fragments of higher-order logics. In Section 4.5 we give an upper bound for the definability of the classes $AUTOSAT(F)$, with F being the different levels of the AA^i hierarchies. Finally, in Section 4.6 we present our main result regarding the properness of the AA^i hierarchies as well as some considerations regarding an alternative strategy to prove this result by means of a complexity-theoretical type of argument.

4.1 The AA^i Hierarchies

We define next for each order $i \geq 2$ a hierarchy in HO^i defined in terms of both, alternation of quantification and maximal-arity of the higher-order variables.

Definition 4.1. **(AA^i hierarchies).** For $i, r, m \geq 1$,

i. $AA\Sigma^i(r, m)$ is the restriction of $\mathrm{HO}^{i+1,r}$ to prenex formulae with at most m alternating blocks of quantifiers, starting with an existential block. That is, $AA\Sigma^i(r, m)$ is the class of formulae $\varphi \in \mathrm{HO}^{i+1,r}$ of the form

$$\exists\, \overline{V_1}\, \forall\, \overline{V_2}\, \ldots\, Q_m\, \overline{V_m}\, (\psi)$$

where ψ is a quantifier-free $\mathrm{HO}^{i+1,r}$ formula, Q_m is either $\exists$ if m is odd or $\forall$ if m is even, and for $1 \leq j \leq m$, each variable in the vector $\overline{V_j}$ is a variable of order $\leq i+1$ and maximal-arity $\leq r$.

ii. $AA\Pi^i(r, m)$ is defined in the same way as $AA\Sigma^i(r, m)$, but now we require that the first block consists of universal quantifiers.

iii. $AA^i(r, m) = AA\Sigma^i(r, m) \cup AA\Pi^i(r, m)$.

Note that, the formulae in the AA^i hierarchies are in prenex normal form, but not necessarily in GSNF. Thus, the quantifiers of the highest order do not necessarily precede all the remaining quantifiers in the prefix, as it is the case in the Σ^i_m hierarchies. Furthermore, in the AA^i hierarchies we count every alternation of quantifiers in the prefix, independently of their order, while in the Σ^i_m hierarchies the only alternation counted are those corresponding to the quantifiers applied to the variables of the highest order.

Makowsky and Pnueli showed that the AA^1 hierarchy imposes a proper hierarchy in second-order logic.

Theorem 4.2 (Th. A [MP96][1]). *For every $r, m \geq 1$ there are Boolean queries not expressible in $AA^1(r, m)$ but expressible in $AA^1(r + c(r), m + 6)$, where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$.*

They proved this result by introducing the set $AUTOSAT(F)$ of formulae in a given logic F which satisfy themselves, in a certain encoding of the formulae as structures. Following a similar strategy, we show that, for every $i \geq 2$, the analogous AA^i hierarchy imposes a proper hierarchy in HO^{i+1}.

4.2 Encoding Well-Formed Formulae into Relational Structures

Using *word models* (see Section 3.4 in the previous chapter or [EF99] for a definition), every formula of finite-order logic over a given relational vocabulary σ can be viewed as a finite relational structure of the following vocabulary.

$$\pi(\sigma) = \{<, R_\neg, R_\vee, R_\wedge, R_\exists, R_\forall, R_(, R_), R_=, R_x, R_X, R_|, R_\iota, R_,\} \cup \{R_a : a \in \sigma\}$$

Example 4.3. If $\sigma = \{E\}$ is the vocabulary of graphs and φ is the sentence

$$\exists X_1^{(\iota)}(\exists x_2(X_1^{(\iota)}(x_2) \vee E(x_2, x_2))),$$

which using our notation for the variables corresponds to

$$\exists X(\iota)|(\exists x||(X(\iota)|(x||) \vee E(x||, x||))),$$

then the following $\pi(\sigma)$-structure $\mathbf{I}$ encodes φ.

$$\mathbf{I} = \langle I, <^{\mathbf{I}}, R^{\mathbf{I}}_\neg, R^{\mathbf{I}}_\vee, R^{\mathbf{I}}_\wedge, R^{\mathbf{I}}_\exists, R^{\mathbf{I}}_\forall, R^{\mathbf{I}}_(, R^{\mathbf{I}}_), R^{\mathbf{I}}_=, R^{\mathbf{I}}_x, R^{\mathbf{I}}_X, R^{\mathbf{I}}_|, R^{\mathbf{I}}_\iota, R^{\mathbf{I}}_,, R^{\mathbf{I}}_E\rangle$$

[1]Theorem A in [MP96] actually states $m + 4$ instead of $m + 6$, but as we explain latter in Remark 4.29, $m + 4$ seems to be a bit stronger than what is warranted by the proof.

where $<^{\mathbf{I}}$ is a linear order on I, $|I| = length(\varphi)$, and for each $R_a \in \pi(\sigma)$, $R_a^{\mathbf{I}}$ contains the positions in φ carrying an a,

$$R_a^{\mathbf{I}} = \{b \in I : \text{for some } j \ (1 \leq j \leq |I|),\ a \text{ is the } j\text{-th symbol in } \varphi, \text{ and}$$

$$b \text{ is the } j\text{-th element in the order } <^{\mathbf{I}}\}$$

Moreover, instead of having a different vocabulary $\pi(\sigma)$ depending on the vocabulary σ of the formulae which we want to encode as relational structures, we can have a vocabulary ρ rich enough to describe formulae of finite-order logic for any arbitrary vocabulary σ. That is, we can fix a vocabulary ρ such that every formula of finite-order logic of whichever vocabulary σ can be viewed as a finite ρ-structure. This can be done as follows. Let ρ be the vocabulary

$$\{<, R_\neg, R_\vee, R_\wedge, R_\exists, R_\forall, R_(, R_), R_x, R_X, R_|, R_\iota, R_,, R_P\}$$

We first identify every formula φ of finite-order logic over an arbitrary vocabulary σ, with a formula φ' over the vocabulary $\sigma' = \{P|^i : 1 \leq i \leq |\sigma| + 1\}$, where, for a predefined bijective function f from $\sigma \cup \{=\}$ to σ', φ' is the formula obtained by replacing in φ each occurrence of a relation symbol $R \in \sigma \cup \{=\}$ by the word $f(R) \in \sigma'$. We then identify every formula φ with the ρ-structure $\mathbf{I}_{\varphi'}$ corresponding to the word model for φ'.

Note that, following the previous schema, even the formulae of finite-order logic over ρ can be viewed as finite ρ-structures. This motivates the following important definition.

Definition 4.4. Let F be a set of formulae of finite-order logic of vocabulary ρ, let $\rho' = \{P|^i : 1 \leq i \leq 15\}$ and let f be the following bijective function from $\rho \cup \{=\}$ to ρ'.

$$\{< \mapsto P_1, = \mapsto P_2, R_\neg \mapsto P_3, R_\vee \mapsto P_4, \ldots, R_\iota \mapsto P_{13}, R_, \mapsto P_{14}, R_P \mapsto P_{15}\}$$

where, for $1 \leq i \leq 15$, P_i denotes a string of the form $P|^i$. For every formula φ of finite-order logic over ρ, let φ' be the formula obtained by replacing in φ each occurrence of a relation symbol $R \in \rho \cup \{=\}$ by the word $f(R) \in \rho'$. We identify every formula φ with the ρ-structure $\mathbf{I}_{\varphi'}$, where the cardinality of I is the length of φ', $<^{\mathbf{I}_{\varphi'}}$ is a linear order on I, and for each $R_a \in \rho$, $R_a^{\mathbf{I}_{\varphi'}}$ corresponds to the positions in φ' carrying an a, i.e., we identify φ with the word model for φ'.

- We denote by $\mathit{WFF}(F)$ the set of finite ρ-structures $\mathbf{I}_{\varphi'}$ such that $\varphi \in F$.
- We denote by $\mathit{AUTOSAT}(F)$ the set of finite ρ-structures $\mathbf{I}_{\varphi'}$ such that φ is in F and there is a valuation val for which $\mathbf{I}_{\varphi'}, \mathit{val} \models \varphi$.
- We define $\mathit{DIAG}(F) = \mathit{WFF}(F) \setminus \mathit{AUTOSAT}(F)$.

Let us see some concrete examples of finite ρ-structures which belong to $\mathit{WFF}(F)$, $\mathit{AUTOSAT}(F)$ and $\mathit{DIAG}(F)$.

Example 4.5. Let $\mathbf{I}_{\varphi'}$ be the ρ-structure corresponding to the word model for

$$\varphi' \equiv \exists x|(\exists x||(P|(x|, x||)))$$

which encodes the formula $\varphi \equiv \exists x_1(\exists x_2(<(x_1, x_2)))$. It follows that $\mathbf{I}_{\varphi'}$ belongs to $\mathit{WFF}(\mathrm{FO}[\rho])$, as φ is a wff in $\mathrm{FO}[\rho]$. Since $\mathbf{I}_{\varphi'}$ is the word model for φ', there are at least two elements $x_1, x_2 \in \mathit{dom}(\mathbf{I}_{\varphi'})$ such that $x_1 < x_2$. Therefore, $\mathbf{I}_{\varphi'} \models \varphi$ and consequently $\mathbf{I}_{\varphi'}$ belongs to $\mathit{AUTOSAT}(\mathrm{FO}[\rho])$. On the other hand, the ρ-structure $\mathbf{I}_{\psi'}$ corresponding to the word model for $\psi' \equiv (\neg\varphi')$, also belongs to $\mathit{WFF}(\mathrm{FO}[\rho])$, as the formula $\psi \equiv (\neg\varphi)$ encoded by $\mathbf{I}_{\psi'}$ is a wff in $\mathrm{FO}[\rho]$, but clearly $\mathbf{I}_{\psi'}$ is not in $\mathit{AUTOSAT}(\mathrm{FO}[\rho])$. This means that $\mathbf{I}_{\psi'}$ is in $\mathit{DIAG}(\mathrm{FO}[\rho])$. Finally, the $\mathbf{I}_{\alpha'}$ structure corresponding to the word model for $\alpha' \equiv xP|||$ is not in $\mathit{WFF}(\mathrm{FO}[\rho])$, and therefore neither is in $\mathit{AUTOSAT}(\mathrm{FO}[\rho])$ nor in $\mathit{DIAG}(\mathrm{FO}[\rho])$, as the formula $\alpha \equiv xR_{\neg}$ encoded by $\mathbf{I}_{\alpha'}$ is not a well-formed formulae in $\mathrm{FO}[\rho]$.

Note that, for $i, r, m \geq 1$ and $F = AA^i(r, m)$, $\mathit{WFF}(F)$, $\mathit{AUTOSAT}(F)$ and $\mathit{DIAG}(F)$ are not empty.

4.3 Recognizing Well-Formed Formulae

We consider now the complexity of recognizing well-formed formulae of finite-order logic.

Recall that, by the fundamental result of Büchi and Elgot (see Theorem 3.15), the regular languages are precisely those which can be defined in monadic second-order. Since every regular language is actually definable by a monadic Σ^1_1 sentence formed by a block of existential quantifiers with unary second-order variables, followed by a block of first-order universal quantifiers, followed in turn by a first-order formula which is quantifier free, we have the following fact.

Fact 4.6. *Every regular language is definable in the level $A A\Sigma^1(1,2)$ of the hierarchy of arity and alternation of second-order.*

Regarding context-free languages, Makowsky and Pnueli proved the following.

Proposition 4.7 (Prop. 14 [MP96])**.** *Every context-free language is definable in the level $A A\Sigma^1(3,3)$ of the hierarchy of arity and alternation of second-order.*

Therefore, for each context-free grammar G with set of terminal symbols T there is a $A A\Sigma^1(3,3)$ formula φ_G of vocabulary $\pi(T) = \{<\} \cup \{R_a : a \in T\}$ such that

$$Mod(\varphi_G) = \{\mathbf{I} \in \mathcal{B}_{\pi(T)} : \mathbf{I} \text{ is a word model for some } v \in \mathcal{L}(G)\}$$

It is not difficult to see that, for $i, r, m \geq 1$ and a relational vocabulary σ, the set of wff's of $AA^i(r,m)$ over σ is a context-free language. The following example illustrates this fact by showing a context-free grammar for one of these sets of formulae.

Example 4.8. Let $G = (N, T, P, \mathrm{S})$ be a context-free grammar where $N = \{\mathrm{S}, \mathrm{F}, \mathrm{B}_\Sigma, \mathrm{B}_\Pi, \mathrm{V}, \mathrm{A}, \mathrm{I}\}$ is the set of nonterminal symbols, $T = \{\neg; \vee; \wedge; \exists; \forall; (;); =; x; X; |; \iota; R; ,\}$ is the set of terminal symbols, S is the start symbol, and P is the following set of productions:

$\mathrm{S} \to \mathrm{F}$ "quantifier-free formula F"

$\mathrm{S} \to \mathrm{B}_\Sigma\mathrm{F} \cup \mathrm{B}_\Sigma\mathrm{B}_\Pi\mathrm{F} \cup \mathrm{B}_\Sigma\mathrm{B}_\Pi\mathrm{B}_\Sigma\mathrm{F}$ "zero, one or two alternations of quantifiers, starting with an existential block"

$\mathrm{S} \to \mathrm{B}_\Pi\mathrm{F} \cup \mathrm{B}_\Pi\mathrm{B}_\Sigma\mathrm{F} \cup \mathrm{B}_\Pi\mathrm{B}_\Sigma\mathrm{B}_\Pi\mathrm{F}$ "zero, one or two alternations of quantifiers, starting with an universal block"

$\mathrm{B}_\Sigma \to \exists\mathrm{V} \cup \exists\mathrm{V}\mathrm{B}_\Sigma$ "block of existential quantifiers"

$\mathrm{B}_\Pi \to \forall\mathrm{V} \cup \forall\mathrm{V}\mathrm{B}_\Pi$ "block of universal quantifiers"

$\mathrm{F} \to \mathrm{A} \cup (\mathrm{F}\vee\mathrm{F}) \cup (\mathrm{F}\wedge\mathrm{F}) \cup (\neg\mathrm{F})$ "atomic formula or compound formula"

$\mathrm{V} \to x\mathrm{I} \cup X(\iota)\mathrm{I} \cup X((\iota))\mathrm{I}$ "a variable is a first-order variable x or a higher-order variable $X\tau$, where $\tau \in typ(3,1) \setminus \{\iota\}$, plus an index"

$\mathrm{A} \to x\mathrm{I} = x\mathrm{I}$ "atomic formula of the form $v_1 = v_2$"

$\mathrm{A} \to \mathrm{R}(x\mathrm{I})$ "atomic formula of the form $R(v)$"

A $\to$ $X(\iota)$I(xI) $\cup$ $X((\iota))$I($X(\iota)$I) "atomic formula of the form $V^{\tau}(V_1^{\tau_1})$ where $\tau = (\tau_1) \in typ(3,1) \setminus \{\iota\}$"

I $\to$ | $\cup$ |I "index"

The language $\mathcal{L}(G)$ generated by G is precisely the set of wff's of $AA^2(1,3)$ over $\sigma = \{R\}$, where R is a unary relation symbol.

The following proposition is a consequence of the previous observations.

Proposition 4.9. *For $i, r, m \geq 1$, the notion of wff of $AA^i(r,m)$ is not definable in $AA\Sigma^1(1,1)$ but is definable in $AA\Sigma^1(3,3)$.*

Proof. (Sketch). Given that, for $i, r, m \geq 1$ and a relational vocabulary σ, the set of wff's of $AA^i(r,m)$ over σ is a context-free language, and that, by Proposition 4.7, every context free language is definable in $AA\Sigma^1(3,3)$, we can conclude that the notion of wff of $AA^i(r,m)$ is definable in $AA\Sigma^1(3,3)$.

To see that the notion of wff of $AA^i(r,m)$ is not definable in $AA\Sigma^1(1,1)$, let us assume that, for some relational vocabulary σ and some $i, r, m \geq 1$, the set of wff's of $AA^i(r,m)$ over σ is definable in $AA\Sigma^1(1,1)$. Since monadic Σ^1_1 includes $AA\Sigma^1(1,1)$, it follows that, by Büchi's characterization of regular languages, the set of wff's over σ for such fragment $AA^i(r,m)$ is regular. But it is straightforward to show, by using the pumping lemma for regular languages (see [HU79]), that for every $i, r, m \geq 1$, $AA^i(r,m)[\sigma]$ is non-regular. This contradicts the assumption that the set of wff's of some fragment $AA^i(r,m)$ is definable in $AA\Sigma^1(1,1)$, completing the proof. $\square$

For every $i, r, m \geq 1$, we now want to establish the complexity of recognizing the class $WFF(AA^i(r,m)[\rho])$. Note that, Proposition 4.9 refers to the class $\{\mathbf{I}_\varphi \in \mathcal{B}_{\pi(\rho)} : \varphi \in AA^i(r,m)[\rho]\}$, whereas $WFF(AA^i(r,m)[\rho])$ is the class $\{\mathbf{I}_{\varphi'} \in \mathcal{B}_\rho : \varphi \in AA^i(r,m)[\rho]\}$. However, since the set $\{\varphi' : \varphi \in AA^i(r,m)[\rho]\}$ is still context-free, Proposition 4.9 also follows for the class $WFF(AA^i(r,m)[\rho])$.

Fact 4.10. *For $i, r, m \geq 1$, the set $WFF(AA^i(r,m)[\rho])$, i.e., the set of finite ρ-structures $\mathbf{I}_{\varphi'}$ such that $\varphi \in AA^i(r,m)[\rho]$, is not definable in $AA\Sigma^1(1,1)[\rho]$, but is definable in $AA\Sigma^1(3,3)[\rho]$.*

4.4 Lower Bound

In this section, we define, for each level of the AA^i hierarchies, Boolean queries which are not expressible in that level.

Proposition 4.11. *For $i, r, m \geq 1$ and $F = AA^i(r,m)[\rho]$, $DIAG(F)$ is* not *definable in F.*

Proof. Towards a contradiction, let us assume that $DIAG(F)$ is definable in F. Then there is a sentence $\psi_D \in F$ such that $Mod(\psi_D) = DIAG(F)$. From the definition of $DIAG(F)$ and from our assumption, it follows that for an arbitrary ρ-structure $\mathbf{I}_{\varphi'}$ which encodes a sentence $\varphi \in F$, $\mathbf{I}_{\varphi'} \not\models \varphi$ iff $\mathbf{I}_{\varphi'} \models \psi_D$. On the other hand, there is a finite ρ-structure $\mathbf{I}_{\psi'_D}$ which encodes the sentence ψ_D. But then $\mathbf{I}_{\psi'_D} \not\models \psi_D$ iff $\mathbf{I}_{\psi'_D} \models \psi_D$, which is a contradiction. $\square$

Proposition 4.12. *For $i \geq 1$, $r \geq 3$, $m \geq 4$ and $F = AA^i(r,m)[\rho]$, $AUTOSAT(F)$ is* not *definable in F.*

Proof. Let us assume that $AUTOSAT(F)$ is definable in F, i.e., that there is a sentence $\psi_A \in F$ such that $Mod(\psi_A) = AUTOSAT(F)$. We know by Fact 4.10 that there is a sentence ψ_W in $AA\Sigma^1(3,3)[\rho]$, and therefore in F, such that $Mod(\psi_W) = WFF(F)$. But then, there is a sentence φ which is in prenex normal form and which is logically equivalent to $\psi_W \wedge \neg\psi_A$. Furthermore, as ψ_W has at most 3 alternating blocks of quantifiers and ψ_A has at most $m \geq 4$ alternating blocks of quantifiers, there is a sentence ψ_D with at most m alternating blocks of quantifiers which is equivalent to φ. It follows that $\psi_D \in F$ and $Mod(\psi_D) = DIAG(F)$ which contradicts Proposition 4.11 and hence the assumption that $AUTOSAT(F)$ is definable in F is not true. $\square$

Remark 4.13. It seems natural to conjecture that the previous proposition is also true for every level of the AA^i hierarchies below $AA^i(3,4)$. Unfortunately, at least for the AA^1 hierarchy of second-order logic, it does not seem possible to prove that by using the diagonalization argument which we employ here and which was introduced in the work of Makowsky and Pnueli. The major problem is that by Fact 4.10 we cannot define $WFF(F)$ in $AA\Sigma^1(1,1)$. Furthermore, as we know that $WFF(F)$ is definable in $AA^1(3,3)$, but we do not know whether $WFF(F)$ is definable in $AA^1(2,3)$, we need $r \geq 3$. Finally, as the set of formulae in each level of the $AA^1(r,m)$ hierarchy is closed under negation, but not under conjunction, we need $m \geq 4$. Note that the fact that each level of the AA^i hierarchies is closed under negation seems to be not enough since, for every $i, r, m \geq 1$, $WFF(AA^i(r, m)) \subset \mathcal{B}_\rho$. That is, there are structures of signature ρ which *do not* encode a formula from $AA^i(r, m)$.

We summarize the classes of formulae for which we know that $AUTOSAT(F)$ is *not* definable in F, in the following proposition. We say that a class of formulae F is *closed under conjunction with a formula γ of vocabulary ρ* iff, for every $\varphi \in F[\rho]$ there is a formula $\psi \in F[\rho]$ which is equivalent to $\varphi \wedge \gamma$.

Proposition 4.14. *Let F be a class of formulae of finite-order logic whose syntax is context free. If F is closed under negation and conjunction with second-order formulae of the form $\exists \overline{X} \forall \bar{y} \exists \bar{z}(\psi)$ where $\overline{X}$ is a vector of second-order variables of arity 3, $\bar{y}$ and $\bar{z}$ are vectors of first-order variables and ψ is a quantifier-free formula of vocabulary ρ, then $AUTOSAT(F[\rho])$ is* not *definable in $F[\rho]$.*

Note that, according to the proof of Proposition 14 in [MP96] (Proposition 4.7 in the present work), every context-free language is definable by a second-order formula of the form $\exists \overline{X} \forall \bar{y} \exists \bar{z}(\psi)$ where $\overline{X}$ is a vector of second-order variables of arity 3, $\bar{y}$ and $\bar{z}$ are vectors of first-order variables and ψ is a quantifier-free formula.

4.5 Upper Bound

We give in this section an upper bound for the definability for every $i, r, m \geq 1$ of $AUTOSAT(AA^i(r, m)[\rho])$. We follow the same strategy as in [MP96], adapting the definitions and proving the corresponding lemmas for every order $i \geq 2$.

Given that for each $\mathrm{HO}^{i,r}$ fragment, and thus for each layer of the AA^i hierarchies, both the order and the maximal-arity of the variables are bounded, we assume, for the sake of simplicity, a vocabulary ρ with a different relation symbol for each different type of higher-order variable, i.e.,

$$\rho = \{<, R_\neg, R_\vee, R_\wedge, R_\exists, R_\forall, R_(, R_), R_x, R_|, R_,, R_P\} \cup \{R_{X^\tau} : \tau \in \mathit{typ}(i, r) \setminus \{\iota\}\}$$

Note that strictly speaking, ρ should be denoted as $\rho_{i,r}$, since it depends upon i and r. However, to simplify the notation, we use simply ρ.

We identify every formula $\varphi \in \mathrm{HO}^{i,r}[\rho]$ with a formula φ' over the vocabulary $\rho' = \{P|^j : 1 \leq j \leq 12 + |\mathit{typ}(i, r)|\}$, where φ' is the formula obtained by replacing in φ each occurrence of a relation symbol $R \in \rho \cup \{=\}$ by the word $f(R) \in \rho'$, for the following bijective function f from ρ to ρ',

$$\{< \mapsto P_1, = \mapsto P_2, R_\neg \mapsto P_3, \ldots, R_P \mapsto P_{13}\} \cup \{R_{X^{\tau_j}} \mapsto P_{13+j} : \tau_j \text{ is}$$

$$\text{the } j\text{-th type in the lexicographical order of } \mathit{typ}(i,r) \setminus \{\iota\}\}.$$

As before, we encode each formula φ using the ρ-structure $\mathbf{I}_{\varphi'}$ which corresponds to the word model for φ'.

From now on, we call a word model of vocabulary ρ simply a word, and if $R_a(x)$ for some unary relation symbol $R_a \in \rho$, we say that the symbol in position x of the word is a.

Definition 4.15 (Def. 21 [MP96][2])**.** For each $\tau \in \mathit{typ}(i,r)$,

i. $\mathit{VAR}^\tau(x)$ is a predicate indicating that the symbol in position x is a symbol of a variable of type τ;

ii. $\mathit{INDEX}^\tau(x, y_1, y_2)$ is a predicate indicating that the symbol in position x is a symbol of a variable of type τ, and the symbols in positions y_1 to y_2 encode its index;

iii. $\mathit{SAME}^\tau(x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 and x_2 are symbols of variables of type τ, and they have the same index (that is, they refer to the *same* variable).

Lemma 4.16 (Lemma 22 [MP96][2])**.** *The predicate $\mathit{VAR}^\tau(x)$ can be expressed by a quantifier-free first-order formula; $\mathit{INDEX}^\tau(x, y_1, y_2)$ can be expressed by a prenex first-order formula with an existential block of quantifiers followed by a universal block; and $\mathit{SAME}^\tau(x_1,\, x_2)$ can be expressed by a formula in $AA\Sigma^1(2,3)$.*

Proof. $\mathit{VAR}^\tau(x)$ is simply $R_{X^\tau}(x)$ where R_{X^τ} is the relation symbol in ρ used to denote variables of type τ.

For $\mathit{INDEX}^\tau(x, y_1, y_2)$, we write

$\mathit{VAR}^\tau(x) \wedge \forall z\big((y_1 < z \wedge z < y_2) \rightarrow R_|(z)\big)$

"x is a variable of type τ and every symbol z between y_1 and y_2 is "|"" $\wedge$

[2]This definition/lemma is an adaptation of the cited definition/lemma from the work of Makowsky and Pnueli to the case of higher-order logics.

$R_|(y_1) \wedge R_|(y_2)$
"the symbols in positions y_1 and y_2 are both "|"" $\wedge$

$\forall z \neg(x < z \wedge z < y_1)$
"y_1 is the successor of x" $\wedge$

$\exists z \big(y_2 < z \wedge \forall z_2 \neg(y_2 < z_2 \wedge z_2 < z) \wedge \neg R_|(z)\big)$
"z is the successor of y_2 and the symbol in position z is not "|""

For $SAME^{\tau}(x_1, x_2)$, we write

$$\exists y_1 y_2 y_3 y_4 F \Big(INDEX^{\tau}(x_1, y_1, y_2) \wedge INDEX^{\tau}(x_2, y_3, y_4) \wedge \\ \text{"}F \text{ is a bijective function from the range } y_1 - y_2 \text{ to } y_3 - y_4\text{"} \Big)$$

It is not difficult to see that there is formula in $AA\Sigma^1(2,3)$ which is equivalent to the previous one. □

We now want to encode valuations for the different variables in a formula. Given a ρ-structure $\mathbf{I}_{\varphi'}$ which encodes a formula $\varphi \in AA^i(r,m)[\rho]$, and given a valuation v on $\mathbf{I}_{\varphi'}$, for each type $\tau \in typ(i+1,r)$ the values assigned by v to all variables X^{τ} of type $\tau = (\tau_1, \dots, \tau_k)$ which appear in φ are encoded by an object (relation) $R^{\tau'} \in I_{\tau'}$ of type $\tau' = (\iota, \tau_1, \dots, \tau_k)$. First-order variables, i.e., variables of type ι, are a special case since the values assigned by v to the first-order variables which appear in φ are encoded by a binary relation $R \in I_{\iota^2}$ of type ι^2, with the additional restriction that each element corresponding to a first-order variable is related to exactly one element. We use $\overline{V}$ to denote a vector which contains a different variable $V^{(\iota,\tau_1,\dots,\tau_k)}$ for each type $(\tau_1, \dots, \tau_k) \in typ(i+1,r) \setminus \{\iota\}$, plus a second-order variable F of arity 2. Such vector $\overline{V}$ has exactly $|typ(i+1,r)|$ different variables which we use to encode valuations for the fragment $AA^i(r,m)$.

Definition 4.17 (Def. 23 [MP96]^2)**.**

i. $VAL^{\tau}(V^{\tau'})$ are predicates indicating that $V^{\tau'}$ encodes a valuation for all variables of type τ in the formula.

ii. $VAL(\overline{V})$ is a predicate indicating that the vector $\overline{V}$ encodes a valuation for all variables occurring in the formula.

iii. $ASSIGNS^{\tau}(V^{\tau'}, Y^{\tau}, x)$ are predicates indicating that Y^{τ} is the object assigned by the valuation encoded by $V^{\tau'}$ to the variable of type τ in position x.

Lemma 4.18 (Lemma 24 [MP96][2]). *$VAL^\iota(F)$ can be expressed by a formula in $AA\Pi^1(2,3)$. For $\tau = (\tau_1,\dots,\tau_k)$, $\tau \neq \iota$, $VAL^\tau(V^{\tau'})$ can be expressed by a formula in $AA\Pi^{j-1}(ma(\tau'),3)$, where $j = order(\tau)$. For a given order i and maximal-arity r, $VAL(\overline{V})$ can be expressed by a formula in $AA\Pi^{i-1}(r+1,3)$.*

$ASSIGNS^\iota(F,y,x)$ can be expressed by a quantifier-free second-order formula of arity 2. For $\tau = (\tau_1,\dots,\tau_k)$, $\tau \neq \iota$, $ASSIGNS^\tau(V^{\tau'},Y^\tau,x)$ can be expressed by a formula in $AA\Pi^{j-1}(ma(\tau'),1)$, where $j = order(\tau)$.

Proof. For $VAL^\iota(F)$, we say that F encodes a valuation for all first-order variables in the formula.

$\forall x_1 x_2 y\big(\neg SAME^\iota(x_1,x_2) \vee (F(x_1,y) \leftrightarrow F(x_2,y))\big)$
"If a first-order variable occurs more than once, then F assigns the same set of values to each occurrence" $\wedge$

$\forall x_1 \exists x_2 \big(VAR^\iota(x_1) \rightarrow F(x_1,x_2)\big)$
"F assigns at least one value to each first-order variable" $\wedge$

$\forall x_1 x_2 x_3 \big(F(x_1,x_2) \wedge F(x_1,x_3) \rightarrow x_2 = x_3\big)$
"F assigns at most one value to each first-order variable"

For $VAL^\tau(V^{\tau'})$ where $\tau \neq \iota$, we say that $V^{\tau'}$, $\tau' = (\iota,\tau_1,\dots,\tau_k)$, encodes a valuation for all higher-order variables of type $\tau = (\tau_1,\dots,\tau_k)$ in the formula.

$$\forall x_1 x_2 X_1^{\tau_1}\dots X_k^{\tau_k}\Big(\neg SAME^\tau(x_1,x_2) \vee \big(V^{\tau'}(x_1,X_1^{\tau_1},\dots,X_k^{\tau_k}) \leftrightarrow V^{\tau'}(x_2,X_1^{\tau_1},\dots,X_k^{\tau_k})\big)\Big)$$

"$V^{\tau'}$ assigns the same object in I_τ to each occurrence of a same higher-order variable of type τ in the formula"

For a given order i and maximal-arity r, $VAL(\overline{V})$ is written as the conjunction of the $|typ(i,r)|$ formulae for VAL^τ, i.e,

$$\bigwedge_{(\tau_1,\dots,\tau_k)\in typ(i,r)\setminus\{\iota\}} \Big(VAL^{(\tau_1,\dots,\tau_k)}(V^{(\iota,\tau_1,\dots,\tau_k)})\Big) \wedge VAL^\iota(F)$$

For $ASSIGNS^\iota(F,y,x)$, we say that F assigns y to the first-order variable in position x if $F(x,y)$.

For $ASSIGNS^\tau(V^{\tau'},Y^\tau,x)$, where $\tau = (\tau_1,\dots,\tau_k)$ and $\tau' = (\iota,\tau_1,\dots,\tau_k)$, the corresponding formula is as follows.

$\forall Z_1^{\tau_1} \ldots Z_k^{\tau_k} \left(V^{(\iota,\tau_1,\ldots,\tau_k)}(x, Z_1^{\tau_1} \ldots Z_k^{\tau_k}) \leftrightarrow Y^{\tau}(Z_1^{\tau_1} \ldots Z_k^{\tau_k}) \right)$

"$V^{(\iota,\tau_1,\ldots,\tau_k)}$ assigns Y^{τ} to x" □

Definition 4.19 (Def. 25 [MP96]^2)**.**

i. *WFF*(x_1, x_2) is a predicate indicating that the symbols in positions x_1 to x_2 form a well-formed sub-formula.

ii. For $3 \leq j \leq 12 + |typ(i,r)|$, $ATOM_{P_j}(x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 to x_2 form an atomic sub-formula of the form $P|^j(x)$. Note that, $P|^j$ is a word in ρ' which corresponds to a unary relation symbol in the vocabulary ρ.

iii. For $1 \leq j \leq 2$, $ATOM_{P_j}(x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 to x_2 form an atomic sub-formula of the form $P|^j(x, y)$. Recall that $P|$ and $P||$ are the words in ρ' which correspond to the binary relation symbols $<$ and $=$, respectively.

iv. For every $\tau \in typ(i,r) \setminus \{\iota\}$, $ATOM_{X^{\tau}}(x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 to x_2 form an atomic sub-formula of the form $V^{\tau}(V_1^{\tau_1}, \ldots, V_k^{\tau_k})$, where $\tau = (\tau_1, \ldots, \tau_k)$.

v. For every $\tau \in typ(i,r) \setminus \{\iota\}$, $POS_j^{\tau}(x_1, x_2)$ is a predicate indicating that the symbol in position x_1 is a variable of type $\tau = (\tau_1, \ldots, \tau_k)$, $k \geq j$, and the symbol in position x_2 is the variable V^{τ_j} in the j-th position of it.

vi. $NOT(x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 to x_2 form a sub-formula of the form $(\neg\varphi)$.

vii. $OR(x_1, x_2, x_3)$ is a predicate indicating that the symbols in positions x_1 to x_3 form a sub-formula of the form $(\varphi_1 \vee \varphi_2)$ with x_2 the position of the $\vee$ symbol.

viii. $AND(x_1, x_2, x_3)$ is a predicate indicating that the symbols in positions x_1 to x_3 form a sub-formula of the form $(\varphi_1 \wedge \varphi_2)$ with x_2 the position of the $\wedge$ symbol.

Lemma 4.20 (Lemma 26 [MP96]^2)**.** *The predicates in Definition 4.19 can be expressed by formulae in $AA\Sigma^1(3,3)$.*

Proof. It follows from Proposition 4.7 and the fact that each predicate in Definition 4.19 defines in each model a language which is context-free. □

Definition 4.21 (Def. 27 [MP96]^2).

i. $\mathit{ATOMSAT}(\overline{V}, x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 to x_2 form an atomic sub-formula, which is satisfied by the valuation encoded by $\overline{V}$ for a structure which is the whole formula.

ii. $\mathit{QFREESAT}(\overline{V}, x_1, x_2)$ is a predicate indicating that the symbols in positions x_1 to x_2 form a quantifier-free well-formed sub-formula, which is satisfied by the valuation encoded by $\overline{V}$ for a structure which is the whole formula.

Lemma 4.22 (Lemma 28 [MP96]^2). *Let $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$. For a given order i and maximal-arity r, $\mathit{ATOMSAT}(\overline{V}, x_1, x_2)$ can be expressed by a formula in $AA\Pi^{i-1}(r + c(r), 4)$ whereas $\mathit{QFREESAT}(\overline{V}, x_1, x_2)$ can be expressed by a formula in $AA\Sigma^{i-1}(r + c(r), 5)$.*

Proof. For $\mathit{ATOMSAT}(\overline{V}, x_1, x_2)$ we have different cases depending on the kind of atom we are considering. If the atom is a formula of the form $P|^j(x)$, where $f(R_a) = P|^j$ for R_a a unary relation symbol in ρ, we write

$$\begin{aligned}&\mathit{ATOM}_{P_j}(x_1, x_2) \wedge \mathit{VAL}(\overline{V})\wedge\\ &\exists y p\big(R_x(p) \wedge x_1 < p \wedge p < x_2 \wedge R_a(y) \wedge \mathit{ASSIGNS}^{\iota}(F, y, p)\big)\end{aligned}$$

If the atom is a formula of the form $P|(x, y)$, we write

$$\begin{aligned}&\mathit{ATOM}_{P_1}(x_1, x_2) \wedge \mathit{VAL}(\overline{V})\wedge\\ &\exists y_1 y_2 p_1 p_2\big(R_x(p_1) \wedge R_x(p_2) \wedge x_1 < p_1 \wedge p_1 < p_2 \wedge p_2 < x_2\wedge\\ &\qquad y_1 < y_2 \wedge \mathit{ASSIGNS}^{\iota}(F, y_1, p_1) \wedge \mathit{ASSIGNS}^{\iota}(F, y_2, p_2)\big)\end{aligned}$$

If the atom is a formula of the form $P||(x, y)$, we just replace $y_1 < y_2$ by $y_1 = y_2$ and $\mathit{ATOM}_{P_1}(x_1, x_2)$ by $\mathit{ATOM}_{P_2}(x_1, x_2)$ in the previous formula.

And, if the atom is a formula of the form $V^{\tau}(V_1^{\tau_1}, \dots, V_k^{\tau_k})$, $\tau = (\tau_1, \dots, \tau_k)$, we write

$$\begin{aligned}&\mathit{ATOM}_{X^{\tau}}(x_1, x_2) \wedge \mathit{VAL}(\overline{V})\wedge\\ &\exists Y_1^{\tau_1} \dots Y_k^{\tau_k} p_1 \dots p_k\Big(V^{(\iota,\tau_1,\dots,\tau_k)}(x_1, Y_1^{\tau_1}, \dots, Y_k^{\tau_k})\wedge\\ &\qquad \mathit{POS}_1^{\tau}(x_1, p_1) \wedge \dots \wedge \mathit{POS}_k^{\tau}(x_1, p_k)\wedge\\ &\qquad \mathit{ASSIGNS}^{\tau_1}(V^{\tau_1'}, Y_1^{\tau_1}, p_1) \wedge \dots \wedge \mathit{ASSIGNS}^{\tau_k}(V^{\tau_k'}, Y_k^{\tau_k}, p_k)\Big)\end{aligned}$$

For $\mathit{QFREESAT}(\overline{V}, x_1, x_2)$ we use a second-order variable X and say that, a tuple

(a,b) is in X iff the symbols in positions a to b form a quantifier free sub-formula which is satisfied by the valuation encoded by $\overline{V}$ for a structure which is the whole formula.

$$\begin{aligned}
&\exists X\Big(X(x_1,x_2) \wedge \forall y_1 y_2\Big(X(y_1,y_2) \leftrightarrow \Big(\mathit{ATOMSAT}(\overline{V},y_1,y_2) \vee \\
&\qquad \big(\mathit{NOT}(y_1,y_2) \wedge \neg X(y_1+2,y_2-1)\big) \vee \\
&\qquad \big(\exists y\big(\mathit{OR}(y_1,y,y_2) \wedge (X(y_1+1,y-1) \vee X(y+1,y_2-1))\big)\big) \vee \\
&\qquad \big(\exists y\big(\mathit{AND}(y_1,y,y_2) \wedge (X(y_1+1,y-1) \wedge X(y+1,y_2-1))\big)\big)\Big)\Big)\Big)
\end{aligned}$$

□

Definition 4.23 (Def. 29 [MP96]²).

i. $\exists \mathit{BLOCK}(x_1,x_2,x_3)$ is a predicate indicating that the symbols in positions x_1 to x_2 form a block of existentially quantified variables and the symbols in positions x_2+1 to x_3 form a well-formed sub-formula.

ii. $\forall \mathit{BLOCK}(x_1,x_2,x_3)$ is a predicate indicating that the symbols in positions x_1 to x_2 form a block of universally quantified variables and the symbols in positions x_2+1 to x_3 form a well-formed sub-formula.

iii. $\mathit{VEQUIV}(\overline{V}_1,\overline{V}_2,x_1,x_2)$ is a predicate indicating that $\overline{V}_1$ and $\overline{V}_2$ encode valuations for all variables occurring in the formula, and that $\overline{V}_1$ is S-equivalent to $\overline{V}_2$, with S being the set of variables occurring in the positions x_1 to x_2 of the formula.

Lemma 4.24 (Lemma 30 [MP96]²). *$\exists \mathit{BLOCK}(x_1,x_2,x_3)$ and $\forall \mathit{BLOCK}(x_1,x_2,x_3)$ can be expressed by formulae in $AA\Sigma^1(3,3)$. For a given order i and maximal-arity r, $\mathit{VEQUIV}(\overline{V}_1,\overline{V}_2,x_1,x_2)$ can be expressed by a formula in $AA\Pi^{i-1}(r+1,3)$.*

Proof. $\exists \mathit{BLOCK}(x_1,x_2,x_3)$ and $\forall \mathit{BLOCK}(x_1,x_2,x_3)$ are context free.

For $\mathit{VEQUIV}(\overline{V}_1,\overline{V}_2,x_1,x_2)$, we write

$$\begin{aligned}
&\mathit{VAL}(\overline{V}_1) \wedge \mathit{VAL}(\overline{V}_2) \wedge \\
&\textstyle\bigwedge_{\tau \in typ(i+1,r)} \Big(\forall y\Big(\mathit{VAR}^{\tau}(y) \wedge \neg\exists z\big(x_1 < z \wedge z < x_2 \wedge \mathit{SAME}^{\tau}(y,z)\big) \\
&\qquad\qquad \text{“if the variable in position } y \text{ is a variable of type } \tau \\
&\qquad\qquad \text{which does not appear in the range } x_1 - x_2\text{”} \rightarrow \\
&\qquad\qquad \forall Z^{\tau}\big(\mathit{ASSIGNS}^{\tau}(V_1^{\tau'},Z^{\tau},y) \leftrightarrow \mathit{ASSIGNS}^{\tau}(V_2^{\tau'},Z^{\tau},y)\big)
\end{aligned}$$

"the valuations encoded by $\overline{V}_1$ and $\overline{V}_2$ assign to
the variable in position y, the same object from I_τ" $\Big)\Big)$

Note that $\neg\exists z(x_1 < z \wedge z < x_2 \wedge SAME^\tau(y,z))$ is equivalent to $\forall z(\neg(x_1 < z) \vee \neg(z < x_2) \vee \neg SAME^\tau(y,z))$ and that, by Lemma 4.16, $SAME^\tau(y,z))$ can be expressed by a formula in $AA\Sigma^1(2,3)$ and thus $\neg SAME^\tau(y,z))$ can be expressed by a formula in $AA\Pi^1(2,3)$. Therefore, it is not difficult to see that two alternations of quantifiers starting with a universal block are enough to express $VEQUIV(\overline{V}_1,\overline{V}_2,x_1,x_2)$. $\square$

Definition 4.25 (Def. 31 [MP96][2]).

i. $\exists BLOCKSAT_m(\overline{V},x_1,x_2)$ is a predicate indicating that $\overline{V}$ encodes a valuation and the symbols in positions x_1 to x_2 form a formula with no more than m alternating blocks of quantifiers, starting with an existential quantifier, which is satisfied by $\overline{V}$.

ii. $\forall BLOCKSAT_m(\overline{V},x_1,x_2)$ is a predicate indicating that $\overline{V}$ encodes a valuation and the symbols in positions x_1 to x_2 form a formula with no more than m alternating blocks of quantifiers, starting with a universal quantifier, which is satisfied by $\overline{V}$.

Lemma 4.26 (Lemma 32 [MP96][2]). *Let $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$. For a given order i, maximal-arity r and $m \geq 1$ alternating blocks of quantifiers, $\exists BLOCKSAT_m(\overline{V},x_1,x_2)$ can be expressed by formulae in $AA\Sigma^{i-1}(r + c(r), m + 4)$ if m is odd, and by formulae in $AA\Sigma^{i-1}(r + c(r), m + 5)$ if m is even. Conversely, $\forall BLOCKSAT_m(\overline{V},x_1,x_2)$ can be expressed by formulae in $AA\Pi^{i-1}(r + c(r), m + 4)$ if m is even, and by formulae in $AA\Pi^{i-1}(r + c(r), m + 5)$ if m is odd.*

Proof. We use induction on the number of blocks of quantifiers m. For $m = 0$, $\exists BLOCKSAT_0(\overline{V},x_1,x_2)$ and $\forall BLOCKSAT_0(\overline{V},x_1,x_2)$ are simply

$$QFREESAT(\overline{V},x_1,x_2)$$

For $m > 0$, we say that $\exists BLOCKSAT_m(\overline{V},x_1,x_2)$, if

$$\exists \overline{V}_1 x\big(\exists BLOCK(x_1,x,x_2) \wedge VEQUIV(\overline{V}_1,\overline{V},x_1,x) \wedge$$
$$\forall BLOCKSAT_{m-1}(\overline{V}_1,x+1,x_2)\big)$$

And we say that $\forall BLOCKSAT_m(\overline{V},x_1,x_2)$, if

$$\begin{aligned}
&\forall \overline{V}_1 x\big(\forall BLOCK(x_1, x, x_2) \wedge VEQUIV(\overline{V}_1, \overline{V}, x_1, x) \rightarrow \\
&\qquad \exists BLOCKSAT_{m-1}(\overline{V}_1, x+1, x_2)\big)\wedge \\
&\exists x\big(\forall BLOCK(x_1, x, x_2)\big)
\end{aligned}$$

Note that, the formula expressing $\exists BLOCKSAT_m(\overline{V}, x_1, x_2)$ has an alternating block of quantifiers more than the formula expressing $\forall BLOCKSAT_{m-1}(\overline{V}, x_1, x_2)$. Conversely, the formula expressing $\forall BLOCKSAT_m(\overline{V}, x_1, x_2)$ has an alternating block of quantifiers more than the formula expressing $\exists BLOCKSAT_{m-1}(\overline{V}, x_1, x_2)$. □

Finally, the upper bound for the definability of $AUTOSAT(AA^i(r, m)[\rho])$ is as follows.

Proposition 4.27. *For every $i, r, m \geq 1$, $AUTOSAT(AA^i(r, m)[\rho])$ is definable in $AA\Sigma^i(r + c(r), m + 6)[\rho]$ where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$.*

Proof. We have to show that, for every $i, r, m \geq 1$, there is a sentence $\psi_A \in AA^i(r+c(r), m+6)[\rho]$ where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$, such that $Mod(\psi_A) = AUTOSAT(AA^i(r, m)[\rho])$. The required formulae are simply disjunctions of the $BLOCKSAT$ formulae above.

$$\begin{aligned}
&\exists \overline{V} x_1 x_2 \Bigg(\forall z(\neg(z < x_1)) \wedge \forall z(\neg(x_2 < z)) \\
&\qquad\text{“}x_1 \text{ and } x_2 \text{ are the first and the last element in} \\
&\qquad\text{the ordering } <\text{, respectively”} \wedge \\
&\qquad\Big(QFREESAT(\overline{V}, x_1, x_2)\vee \\
&\qquad\exists BLOCKSAT_1(\overline{V}, x_1, x_2) \vee \ldots \vee \exists BLOCKSAT_m(\overline{V}, x_1, x_2)\vee \\
&\qquad\forall BLOCKSAT_1(\overline{V}, x_1, x_2) \vee \ldots \vee \forall BLOCKSAT_m(\overline{V}, x_1, x_2)\Big)\Bigg)
\end{aligned}$$

□

4.6 Main Result

Our main result stating the properness of the AA^i hierarchies follows.

Theorem 4.28. *For every $i, r, m \geq 1$, there are Boolean queries not expressible in $AA^i(r, m)$ but expressible in $AA^i(r + c(r), m + 6)$, where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$.*

Proof. The result follows from the fact that, for $i, r, m \geq 1$, $\mathit{DIAG}(AA^i(r,m)[\rho])$ is not definable in $AA^i(r,m)[\rho]$ but is definable in $AA^i(r + c(r), m+6)[\rho]$. Proposition 4.11 shows that $\mathit{DIAG}(AA^i(r,m)[\rho])$ is not definable in $AA^i(r,m)[\rho]$. We prove that $\mathit{DIAG}(AA^i(r,m)[\rho])$ is definable in $AA^i(r+c(r), m+6)[\rho]$ as follows. Let $i, r, m \geq 1$, by Fact 4.10 and Proposition 4.27, there is a sentence $\psi_W \in AA\Sigma^1(3,3)[\rho]$ such that $\mathit{Mod}(\psi_W) = \mathit{WFF}(AA^i(r,m)[\rho])$ and a sentence $\psi_A \in AA\Sigma^i(r+c(r), m+6)[\rho]$ such that $\mathit{Mod}(\psi_A) = \mathit{AUTOSAT}(AA^i(r,m)[\rho])$, respectively. Therefore, it is not difficult to see that there is a sentence $\psi_D \in AA\Pi^i(r+c(r), m+6)[\rho]$ which is logically equivalent to $\psi_W \wedge \neg\psi_A$. Since $\mathit{Mod}(\psi_D) = \mathit{DIAG}(AA^i(r,m)[\rho])$, it follows that, for every $i, r, m \geq 1$, $\mathit{DIAG}(AA^i(r,m)[\rho])$ is definable in $AA^i(r + c(r), m+6)$. $\square$

Remark 4.29. Note that, we increment here the number of alternating blocks of quantifiers by 6 instead of by 4 as in the original theorem of Makowsky and Pnueli for second-order logic. We believe that this small divergence is originated in Lemma 26 in [MP96] and propagated downwards to subsequent lemmas in that section. In [MP96], Lemma 26 states that the predicates in Definition 25 can be expressed by formulae in $AA^1(3,1)$ instead of by formulae in $AA\Sigma^1(3,3)$ as we would have expected, since each predicate in Definition 25 in [MP96] defines in each model a language which is context-free.

4.6.1 On an Existential Proof

The following discussion is motivated by observations of Kolodziejczyk (personal communications, October 2005 and November 2007).

According to Kolodziejczyk, the properness of the AA^i hierarchies could also be proved by a complexity-theoretical type of argument involving variants of the time-hierarchy theorem for Σ_m−TIME for any fixed m. We sketch this argument for AA^1 below, but let us note two important points first.

- This approach would provide an existential proof of the AA^i hierarchies while in the work of Makowsky and Pnueli for second-order as well as in our work for all natural orders, a constructive proof is given, since computable queries which separate the classes are exhibited.

- In the opinion of Kolodziejczyk and to the best of our knowledge, there is no written source for the variants of the time-hierarchy theorem for Σ_m−TIME for every fixed m, which are required for this complexity-theoretical argument. Theorem 3.3 in [BDG90] establishes a time-hierarchy theorem for the class ATIME($f(n)$) of languages accepted by $f(n)$ time bounded alternating Turing machines. However, the proof of that theorem uses the fact that ATIME is closed under complementation. Thus, the same strategy cannot be used to prove variants of the time-hierarchy theorem for Σ_m-TIME for every fixed m, since it is not known whether the Σ_m-TIME classes are closed under complementation, and furthermore this seems unlikely. Then, as Kolodziejczyk pointed out to us, it seems that the right thing to look at are the proofs of time-hierarchy theorems for NTIME (see among others [Coo72, SFM78, Zák83]), as they do not require closure under complementation. In his opinion, those proofs could then be adapted to prove similar time-hierarchy theorems for Σ_m-TIME for every fixed m. But we do not know how hard the new proofs can be or whether they are really possible at all.

Now, let us *assume that we can prove* variants of the time-hierarchy theorem for Σ_m-TIME for any fixed m. Then, an alternative existential proof (sketch) for the properness of the AA^1 hierarchy could be as follows: It can be shown that, for $r, m \geq 1$, the data complexity for $AA\Sigma^1(r,m)$ is in Σ_m-TIME(n^r) and the data complexity for $AA\Pi^1(r,m)$ is in Π_m-TIME(n^r). Together, the data complexity for $AA^1(r,m)$ is in $\Sigma_{m+1}-$TIME(n^r). By the time-hierarchy theorem for $\Sigma_{m+1}-$TIME, there is a language L in the level Σ^p_{m+1} of the polynomial-time hierarchy which is not definable in $AA^1(r,m)$. But by Fagin-Stockmeyer theorem (recall Section 2.4), L is definable in Σ^1_{m+1}, and thus in $AA(r', m')$ for some $r' > r$ and $m' > m$. Since by the normal form in [EGG96], every Σ^1_m formula, $m \geq 2$, reduces to a Σ^1_m formula whose first-order part is a Boolean combination of existential formulae, then m' is at most $m+2$. Essentially the same type of argument should also work for each AA^i hierarchy of order $i \geq 2$ if we use Corollary 3.11 instead of Fagin-Stockmeyer theorem, and take into account the tower of exponents $\exp_{i-1}(n^r)$.

Moreover, and *also assuming that we can prove* variants of the time-hierarchy theorem for Σ_m-TIME for any fixed m, the classes $[\Sigma^i_m]^{\leq r}$ of Σ^i_m formulae in which the type of the variables of order i and $i+1$ have *basic-arity* bounded by r, would form for each $i \geq 2$ and $m \geq 1$, a strict hierarchy within Σ^i_m. That is, for $i \geq 2$ and $m, r \geq 1$, it would hold that $[\Sigma^i_m]^{\leq r} \subset [\Sigma^i_m]^{\leq r+1}$. As pointed out in [Kol04b, Kol05], for each $i \geq 2$ and $m \geq 1$, the strictness of the $[\Sigma^i_m]^{\leq r}$ hierarchy would follow from Theorem 3.10 and the time-hierarchy theorem for Σ_m−TIME. Recall that, the *basic-arity* (ba) of a type τ of order ≥ 2 is defined as follows: If τ is a type of order 2, then $ba(\tau)$ is simply the arity of τ. If $order(\tau) > 2$ and $\tau = (\tau_1, \ldots, \tau_r)$, then $ba(\tau) = max(\{ba(\tau_i) : order(\tau_i) = order(\tau) - 1, 1 \leq i \leq r\})$.

Regarding the relationship between the $[\Sigma^i_m]^{\leq r}$ hierarchies and the AA^i hierarchies, we note that:

- By our result (Theorem 4.28), for every $i \geq 1$, the classes $AA^i(r, m)$ form a proper hierarchy of formulae within HO^{i+1}, while for every $i \geq 2$, the classes $[\Sigma^i_m]^{\leq r}$ form a strict hierarchy of formulae within Σ^i_m, provided that we can prove variants of the time-hierarchy theorem for Σ_m−TIME for any fixed m.

- Each $AA^i(r, m)$ class of formulae is closed under complementation. It is not known whether the $[\Sigma^i_m]^{\leq r}$ classes are closed under complementation. Therefore, the same argument that we used to prove a lower bound for the definability of $AUTOSAT(AA^i(r, m)[\rho])$ (see Proposition 4.12), cannot be used to prove a lower bound for the definability of $AUTOSAT([\Sigma^i_m]^{\leq r}[\rho])$.

- Unlike the AA^1 hierarchy of second-order logic formulae which we know by Theorem 4.2 is proper, it is not known whether for some $m, k, c \geq 1$, $[\Sigma^1_m]^{\leq r} \subset [\Sigma^1_m]^{\leq r+c}$ for every $r \geq k$. That is, it is not known whether for some $m \geq 1$, the classes $[\Sigma^1_m]^{\leq r}$ form a proper hierarchy of Σ^1_m formulae. Even assuming that we can prove variants of the time-hierarchy theorem for Σ_m−TIME for any fixed m, it does *not* follows from Theorem 3.10 that for $m \geq 1$, the classes $[\Sigma^1_m]^{\leq r}$ from a proper hierarchy of Σ^1_m formulae, since Theorem 3.10 holds for $i \geq 2$. Recall that, a sufficient condition for the $[\Sigma^1_1]^{\leq r}$ hierarchy to be strict was given in [Kol04a] (see Proposition 3.19 in the previous chapter).

- In the definition of the $[\Sigma^i_m]^{\leq r}$ fragments, only the variables of order $i+1$ and i are restricted to basic-arity $\leq r$. However, as explained in [Kol05], a definition of $[\Sigma^i_m]^{\leq r}$ in which all higher-order variables of order $\leq i + 1$ are required to have basic-arity at most r, would give a class with the same expressive power with respect to sentences. That is, for $i \geq 2$, $m, r \geq 1$, and every sentence $\varphi \in [\Sigma^i_m]^{\leq r}$ (possibly containing variables of order $< i$ with basic-arity $> r$), there is a corresponding sentence $\varphi' \in [\Sigma^i_m]^{\leq r}$ which does not contain higher-order variables of basic-arity $\geq r$ and which is equivalent to φ. To illustrate how this can be done, let us assume that φ is a sentence in $[\Sigma^3_m]^{\leq r}$. By definition of the fragments $[\Sigma^3_m]^{\leq r}$, all variables of order 3 and 4 which appear in φ have basic-arity $\leq r$. Suppose there is a second-order variable V^{ι^k} in φ which has basic-arity $k > r$. We can replace V^{ι^k} by a third-order variable of type $\tau = \underbrace{((\iota), \ldots, (\iota))}_{k}$ and thus of basic-arity 1. This is the case because every second-order relation R of arity k can be encoded as a third-order relation $\mathcal{R}$ of type τ consisting exclusively of tuples of singletons. A tuple of singletons $(\{x_1\}, \ldots, \{x_k\})$ is in $\mathcal{R}$ iff $(x_1, \ldots, x_k)$ is in R. This encoding requires to add some quantification of order 2 and 3 to φ, but no extra quantification of order 4 is needed, therefore the resulting formula φ' is still in $[\Sigma^3_m]^{\leq r}$. A similar encoding can be used for higher orders.

- Apart from the fact that for every type τ of order ≥ 2, it holds that $ba(\tau) \leq ma(\tau)$, there is a more relevant difference between the concepts of basic-arity and maximal-arity. While the set of types of a given order i and maximal-arity r is finite (see Definition 3.1 in the previous chapter), the corresponding set of types of order i and basic-arity r is infinite if $i \geq 3$. For instance, the third-order types $((\iota))$, $((\iota), \iota)$, $((\iota), \iota, \iota)$, $((\iota), \iota, \iota, \iota)$, $\ldots$, have all basic-arity 1, but have maximal-arity 1, 2, 3, 4, $\ldots$, respectively. Thus, if we replace in the definition of the AA^i hierarchies the concept of maximal-arity by the concept of basic-arity, then the proof of the upper bound for $AUTOSAT(AA^i(r,m)[\rho])$ (Proposition 4.27) does not longer hold. Recall that in our proof, for every fragment $AA^i(r,m)$, we fix a vector $\overline{V}$ which has exactly $|typ(i+1,r)|$ different variables. We use $\overline{V}$ to encode valuations for the variables which may appear in any given $AA^i(r,m)$ formula (see the discussion before Definition 4.17 and see also the formula expressing $VAL(\overline{V})$ in the proof of Lemma 4.18). It is not possible to use the same encoding if we use the concept of basic-arity instead of the concept of maximal-arity in the definition of the $AA^i(r,m)$ fragments, since we would have an infinite number of types of order $\leq i+1$ and basic-arity $\leq r$.

- In the $[\Sigma^i_m]^{\leq r}$ hierarchies the first block of quantifiers in the formulae is existential, the quantifiers of order $i+1$ precede all other quantifiers and only the alternations of quantifiers of order $i+1$ are considered, while in the AA^i hierarchies the first block of quantifiers in the formulae can be existential or universal, the quantifiers of order $i + 1$ do not necessarily precede all other quantifiers and all alternations of quantifiers of whichever order are considered. That is, the formulae in the classes $AA^i(r, m)$ are in prenex normal form, but not necessarily in GSNF, while the formulae in the classes $[\Sigma^i_m]^{\leq r}$ are in GSNF and start with an existential block of quantifiers. In the next chapter we study a variation of the AA^i hierarchies, the HAA^i hierarchies, in which the alternations of quantifiers are considered as in the $[\Sigma^i_m]^{\leq r}$ hierarchies.

Chapter 5

Alternating Only the Quantifiers of the Highest Order

In this chapter we study a variation of the AA^i hierarchies, the HAA^i hierarchies, where the alternations are counted as in Σ^i_m and Π^i_m. The difference between the HAA^i and the AA^i hierarchies is that in the HAA^i hierarchies the quantifiers of order $i+1$ precede all other quantifiers in the formulae and only the alternations of quantifiers of order $i+1$ are considered, while in the AA^i hierarchies the quantifiers of order $i+1$ do not necessarily precede all other quantifiers and all alternations of quantifiers of whichever order are considered.

Let $HAA^i(r,m)$ be the class of $\Sigma^i_m \cup \Pi^i_m$ formulae in which the higher-order variables of all orders up to $i+1$ have maximal-arity at most r. We prove in the first section of this chapter that, for each order $i \geq 3$, the corresponding HAA^{i-1} hierarchy is proper. This result is again obtained by using a constructive proof base on the query $AUTOSAT$.

Note that the corresponding version of this hierarchy for second-order logic, HAA^1, was *not* studied in [MP96] regarding the properness of the hierarchy. The HAA^1 hierarchy was used there denoted as SAA, to prove that, for every $r, m \geq 1$, $AUTOSAT(SAA(r,m))$ is PSPACE-complete. Indeed, it is not known whether the $HAA^1(r,m)$ hierarchy is proper.

A method which has been frequently used in classical model theory to compare the expressive power of logics, is Tarski's method of truth definitions [Tar33]. In particular, this method has been successfully used for higher-order logics (see [Lei94]). In finite model theory, Tarski's method has first been explored by M. Mostowski in [Mos01, Mos03]. He introduced the notion of *finite model truth definitions* (*FM-truth*, from now on) and proved a finite version of Tarski's theorem on undefinability of truth. Roughly, he proved that no logic closed under some basic first-order constructions, most notably negation, can define FM-truth for itself over sufficiently large finite models which have a suitable amount of arithmetical structure so that gödelization can be carried out. Same as in the classical case, the concept of FM-truth can be used to compare the expressive power of different fragments of higher order logics, but now over finite models. We examine this in detail in Section 5.2 and make use of the method in Section 5.3. Observe that the same underlying idea of expressing the relationship of satisfaction for logics restricted to finite models by means of logics restricted to finite models, is used in both, $AUTOSAT$ and the concept of FM-truth.

Recently, in [Kol04b, Kol05] the notion of FM-truth definition was further discussed and compared with Vardi's concept of *combined complexity* [Var82], noting an important difference: the possibility of defining FM-truth for a logic $\mathcal{L}$ does not depend on the syntax of $\mathcal{L}$, as long as it is decidable. Furthermore, for each $i, m \geq 1$, a characterization of the logics for which the Σ^i_m class defines FM-truth, was given. In [Kol04a], FM-truth definitions were used to give a sufficient condition for the Σ^1_1 arity hierarchy to be strict over finite structures (see Proposition 3.19 in Chapter 3). We show in Section 5.3 that under the same assumption, the $HAA^1(r, m)$ hierarchy is proper.

5.1 The HAA^i Hierarchies

We define next a new kind of arity-alternation hierarchies in which the only alternations bounded are those corresponding to the quantifiers of the highest order.

Definition 5.1. (**HAA^i hierarchies**) Let $i, r, m \geq 1$,

i. $HAA\Sigma^i(r, m)$ is the class of Σ^i_m formulae in which the higher-order variables of all orders up to $i + 1$ have maximal-arity at most r.

ii. $HAA\Pi^i(r, m)$ is the class of Π^i_m formulae in which the higher-order variables of all orders up to $i + 1$ have maximal-arity at most r.

iii. $HAA^i(r, m) = HAA\Sigma^i(r, m) \cup HAA\Pi^i(r, m)$.

As we mentioned at the beginning of the chapter, the difference between the HAA^i and the AA^i hierarchies is that in the HAA^i hierarchies the quantifiers of order $i+1$ precede all other quantifiers in the formulae and only the alternations of quantifiers of order $i+1$ are considered, while in the AA^i hierarchies the quantifiers of order $i+1$ do not necessarily precede all other quantifiers and all alternations of quantifiers of whichever order are considered.

Note that because of (iii), for every $i \geq 1$, the set of formulae in each level of the $HAA^i(r, m)$ hierarchy is closed under negation.

Regarding the relation between the AA^i hierarchies and the HAA^i hierarchies, it can be easily observed from the proof of Lemma 1 in [HT03, HT06b] that, for every $i, r, m \geq 1$, $HAA^i(r, m) \supseteq AA^i(r, m)$. It is a trivial task to adapt that proof to show that, for every formula $\varphi \in AA^i(r, m)$, there is a formula ψ in GSNF which is equivalent to φ, has the same maximal-arity as φ and does not have more than m alternating blocks of quantifiers of order $i+1$.

The corresponding lower and upper bounds for the definability of $AUTOSAT$ for the different levels of the HAA^i hierarchies are as follows.

Proposition 5.2.

i. For $i = 1$, $r \geq 3$, $m \geq 2$ and $F = HAA^i(r, m)[\rho]$, $AUTOSAT(F)$ is not *definable in F.*

ii. For $i \geq 2$, $r \geq 3$, $m \geq 1$ and $F = HAA^i(r, m)[\rho]$, $AUTOSAT(F)$ is not *definable in F.*

Proof. (Sketch). For every $i, r, m \geq 1$, it is clear that $HAA^i(r, m)$ is a context free language. For instance, it is not difficult to modify the context free grammar for $AA^2(1, 3)$ in Example 4.8 to obtain a context free grammar for $HAA^2(1, 3)$. We just need to add the productions $\mathrm{V}_1 \rightarrow X((\iota))\mathrm{I}$, $\mathrm{V}_2 \rightarrow x\mathrm{I} \cup X(\iota)\mathrm{I}$, and $\mathrm{F} \rightarrow \exists\mathrm{V}_2(\mathrm{F}) \cup \forall\mathrm{V}_2(\mathrm{F})$, and replace the productions with nonterminals B_Σ and B_Π in the left hand side by the productions $\mathrm{B}_\Sigma \rightarrow \exists\mathrm{V}_1 \cup \exists\mathrm{V}_1\mathrm{B}_\Sigma$ and $\mathrm{B}_\Pi \rightarrow \forall\mathrm{V}_1 \cup \forall\mathrm{V}_1\mathrm{B}_\Pi$, respectively.

Since for $r \geq 3$ and $m \geq 2$, the set of formulae in $HAA^1(r, m)[\rho]$ is closed under negation and conjunction with second-order formulae of the form $\exists\overline{X}\forall\bar{y}\exists\bar{z}(\psi)$ where $\overline{X}$ is a vector of second-order variables of arity 3, $\bar{y}$ and $\bar{z}$ are vectors of first-order variables and ψ is a quantifier-free formula of vocabulary ρ, then by Proposition 4.14 it follows that for $r \geq 3$ and $m \geq 2$, $AUTOSAT(HAA^1(r, m)[\rho])$ is not definable in

$HAA^1(r,m)[\rho]$. This proves Part (i). Part (ii) also follows from Proposition 4.14 by noting that, for every $i \geq 2$, $r \geq 3$ and $m \geq 1$, the set of formulae in $HAA^i(r,m)[\rho]$ is closed under negation and conjunction with any formula in $AA^1(3,3)[\rho]$, including of course those of the form $\exists \overline{X} \forall \bar{y} \exists \bar{z}(\psi)$. $\square$

Proposition 5.3. *For $i \geq 2$ and $r, m \geq 1$, $AUTOSAT(HAA^i(r,m)[\rho])$ is definable in $HAA\Sigma^i(r+c(r), m+2)[\rho]$ where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$.*

Proof. (Sketch). We show how the proof of Proposition 4.27 can be adapted to obtain, for every $i \geq 2$, $r, m \geq 1$, a sentence $\psi_A \in HAA^i(r+c(r), m+2)[\rho]$ such that $Mod(\psi_A) = AUTOSAT(HAA^i(r,m)[\rho])$.

The predicates used in Section 4.5 which are related to the identification of variables (Definition 4.15), the encoding of valuations (Definition 4.17), the identification of well-formed sub-formulae (Definition 4.19) and the notion of satisfaction of quantifier free sub-formulae (Definition 4.21), remain the same.

Given the different way in which the quantifiers are arranged in the HAA^i hierarchies, it follows that the predicates that need to be adapted are only those which have to deal with quantified formulae.

First, we need to modify Definition 4.23. Predicates $\exists BLOCK(x_1, x_2, x_3)$ and $\forall BLOCK(x_1, x_2, x_3)$ apply now only to blocks of variables of order at most i, i.e., they hold only if the quantified variables in positions x_1 to x_2 are all of order $\leq i$. We also need to define $\exists BLOCK^i(x_1, x_2, x_3)$ and $\forall BLOCK^i(x_1, x_2, x_3)$ in order to identify blocks of quantifiers of order $i+1$. All these predicates can be expressed by formulae in $HAA\Sigma^1(3,1)$, since they all define in each model a language which is context-free, and in the proof of Proposition 4.7 given in [MP96], the corresponding formula has a block of existentially quantified second-order variables of arity 3, followed by a first-order part.

The most delicate part of the proof is the redefinition of the predicates $\exists BLOCK$-$SAT_m(\overline{V}, x_1, x_2)$ and $\forall BLOCKSAT_m(\overline{V}, x_1, x_2)$. The main difference with the previous case is that now these predicates hold only if the formula in positions x_1 to x_2 is a formula with no more than m alternating blocks of quantifiers of order $i+1$, but with an arbitrary number of alternating blocks of quantifiers of order $\leq i$, which is satisfied by the valuation encoded by $\overline{V}$. Recall that $\overline{V}$ is a vector which contains a different variable $V^{(\iota,\tau_1,\ldots,\tau_k)}$ for each type $(\tau_1, \ldots, \tau_k) \in typ(i+1, r) \setminus \{\iota\}$, plus a second-order variable F of arity 2. Recall also that $\overline{V}$ has exactly $|typ(i+1,r)|$ different variables. We use induction on the number m of alternating blocks of quantifiers of order $i+1$. For $m=0$, $\exists BLOCKSAT_0(\overline{V}, x_1, x_2)$ and $\forall BLOCKSAT_0(\overline{V}, x_1, x_2)$ say that $\overline{V} = \overline{V}_1 \cup \overline{V}_2$ encodes a valuation for all variables occurring in the formula, $\overline{V}_1$ encodes the sub-valuation for the variables of order $i+1$, $\overline{V}_2$ encodes the sub-valuation for the variables of order $\leq i$, and the symbols in positions x_1 to x_2 form a HO^{i,r} prenex formula which is satisfied by $\overline{V}$. Note that, we use here a variable $\mathcal{X}$ of order $i+1$ and arity $|typ(i,r)|+2$ and state that $\mathcal{X}(\overline{V}'_2, y_1, y_2)$ iff the symbols in positions y_1 to y_2 form a HO^{i,r} formula (possibly with free variables of order $i+1$) which is satisfied by $\overline{V}_1 \cup \overline{V}'_2$.

$$\begin{aligned}
&\exists \mathcal{X}\Big(\mathcal{X}(\overline{V}_2, x_1, x_2) \wedge \qquad (5.1)\\
&\quad \forall \overline{V}'_2 y_1 y_2 \Big(\mathcal{X}(\overline{V}'_2, y_1, y_2) \leftrightarrow \\
&\qquad \Big(QFREESAT(\overline{V}_1 \cup \overline{V}'_2, y_1, y_2) \vee \\
&\qquad\quad \exists z \overline{V}''_2 (\exists BLOCK(y_1, z, y_2) \wedge VEQUIV(\overline{V}''_2, \overline{V}'_2, y_1, z) \wedge \\
&\qquad\qquad \mathcal{X}(\overline{V}''_2, z+1, y_2)) \vee \\
&\qquad\quad \exists z \forall \overline{V}''_2 (\forall BLOCK(y_1, z, y_2) \wedge VEQUIV(\overline{V}''_2, \overline{V}'_2, y_1, z) \rightarrow \\
&\qquad\qquad \mathcal{X}(\overline{V}''_2, z+1, y_2))\Big)\Big)\Big)
\end{aligned}$$

For $m > 0$, $\exists BLOCKSAT_m(\overline{V}, x_1, x_2)$ and $\forall BLOCKSAT_m(\overline{V}, x_1, x_2)$ are written exactly as in the proof of Lemma 4.26, but replacing $\exists BLOCK(x_1, x_2, x_3)$ and $\forall BLOCK(x_1, x_2, x_3)$ with $\exists BLOCK^i(x_1, x_2, x_3)$ and $\forall BLOCK^i(x_1, x_2, x_3)$, respectively.

Since we only count here alternations of quantifiers of order $i+1$, the reader can easily check that, for $m \geq 1$, $\exists BLOCKSAT_m(\overline{V}, x_1, x_2)$ can be expressed by formulae in $HAA\Sigma^i(|typ(i,r)|+2, m)$ if m is odd, and by formulae in $HAA\Sigma^i(|typ(i,r)|+2, m+1)$ if m is even. Conversely, $\forall BLOCKSAT_m(\overline{V}, x_1, x_2)$ can be expressed by formulae in $HAA\Pi^i(|typ(i,r)|+2, m)$ if m is even, and by formulae in $HAA\Pi^i(|typ(i,r)|+2, m+1)$ if m is odd.

But we still can get a much tighter upper bound in the maximal-arity than $|typ(i,r)|+2$, since the arity of $\mathcal{X}$ can be significantly decreased if we encode the valuations for the variables of order $\leq i$ differently. For instance we can encode the valuations for all variables of order ≤ 3 and maximal-arity ≤ 2 using just one variable of order 3 and maximal-arity 3, namely V^τ where $\tau = (\iota, (\iota, \iota), (\iota, \iota))$, instead of using a different variable for each different type in $typ(3,2)$. Of course, in order to do this we need to redefine the predicates in Definition 4.17 accordingly. But that can be done without too much difficulty. Just to exemplify, let us suppose that we want to encode valuations for variables of type $\tau' = (\iota, (\iota, \iota))$ using V^τ. We could say that the set of tuples assigned by V^τ to the variable $V^{\tau'}$ in position p, is the set which contains precisely those tuples (z, Z) that have a corresponding tuple (p, X, Z) in V^τ such that (z, z) is the only tuple in X. Thus, the formula expressing the predicate $\mathit{ASSIGNS}^{\tau'}(V^\tau, Y^{\tau'}, x)$ could be written as follows.

$$\forall XZ\Big(V^\tau(x, X, Z) \leftrightarrow \\ \exists z\big(Y^{\tau'}(z, Z) \wedge X(z, z) \wedge \forall y_1 y_2 (X(y_1, y_2) \rightarrow z = y_1 \wedge z = y_2)\big)\Big)$$

Taking into account these considerations, it is not difficult to see that we can encode the valuations for all variables of order $\leq i$ and maximal-arity $\leq r$ which appear in a given formula, using only one variable of order i, maximal-arity $r + 1$ and type $(\iota, \underbrace{\tau_{i-1,r}, \ldots, \tau_{i-1,r}}_{r})$ where, for $j, k \geq 1$,

- $\tau_{j,k} = \iota$ if $j = 1$ and
- $\tau_{j,k} = \underbrace{(\tau_{j-1,k}, \ldots, \tau_{j-1,k})}_{k}$ if $j > 1$.

Thus we can replace the variable $\mathcal{X}$ of arity $|\mathit{typ}(i, r)| + 2$ in (5.1) by a variable of the same order but of arity 3 and type $((\iota, \underbrace{\tau_{i-1,r}, \ldots, \tau_{i-1,r}}_{r}), \iota, \iota)$.

Hence, for $m \geq 1$, $\exists \mathit{BLOCKSAT}_m(\overline{V}, x_1, x_2)$ and $\forall \mathit{BLOCKSAT}_m(\overline{V}, x_1, x_2)$ can be expressed by formulae in $\mathit{HAA}\Sigma^i(r + c(r), m + 1)$ and $\mathit{HAA}\Pi^i(r + c(r), m + 1)$, respectively. For $m = 0$, $\exists \mathit{BLOCKSAT}_m(\overline{V}, x_1, x_2)$ and $\forall \mathit{BLOCKSAT}_m(\overline{V}, x_1, x_2)$ can be expressed by formulae in $\mathit{HAA}\Sigma^i(r + c(r), 1)$.

Finally, the required formulae are simply disjunctions of the $2m$ $\mathit{BLOCKSAT}$ formulae above. $\square$

It seems that the previous result does not apply to the HAA^1 hierarchy of second-order. The main obstacle is that the valuation for first-order variables is encoded into a second-order variable. Thus the variable $\mathcal{X}$ in the proof must be of order at least 3.

We show next that, for $i \geq 2$, each HAA^i hierarchy is proper.

Theorem 5.4. *For every $i \geq 2$ and every $r, m \geq 1$, there are Boolean queries not expressible in $HAA^i(r,m)$ but expressible in $HAA^i(r + c(r), m + 2)$ where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$.*

Proof. Note that, if we replace $F = AA^i(r,m)[\rho]$ by $F = HAA^i(r,m)[\rho]$, Proposition 4.11 still holds. This proves that, for every $i, r, m \geq 1$, $DIAG(HAA^i(r,m)[\rho])$ is not definable $HAA^i(r,m)[\rho]$.

Let us now prove the upper bound. As seen in the proof of Proposition 5.2, for every $i, r, m \geq 1$, $HAA^i(r,m)[\rho]$ is a context free language. It follows from Proposition 4.9 that, for every $i, r, m \geq 1$, there is a sentence $\psi_W \in AA\Sigma^1(3,3)[\rho]$ such that $Mod(\psi_W) = WFF(HAA^i(r,m)[\rho])$. Since $HAA\Sigma^2(3,1) \supseteq AA\Sigma^1(3,3)$, we have that ψ_W is in $HAA\Sigma^2(3,1)[\rho]$ as well. By Proposition 5.3, for every $i \geq 2$ and every $r, m \geq 1$, there is a sentence $\psi_A \in HAA\Sigma^i(r + c(r), m + 2)[\rho]$ such that $Mod(\psi_A) = AUTOSAT(HAA^i(r,m)[\rho])$. Therefore, for every $i \geq 2$ and every $r, m \geq 1$, there is a sentence $\psi_D \in HAA\Pi^i(r+c(r), m+2)[\rho]$ which is logically equivalent to $\psi_W \wedge \neg\psi_A$. Since $Mod(\psi_D) = DIAG(HAA^i(r,m)[\rho])$, it follows that for every $i \geq 2$ and every $r, m \geq 1$, $DIAG(HAA^i(r,m)[\rho])$ is definable in $HAA^i(r+c(r), m+2)$. $\square$

5.2 The Concept of Finite Model Truth Definitions

Tarski's theorem on the undefinability of truth is often expressed by saying that arithmetical truth is not definable (by an arithmetical formula) in the standard model of arithmetic. In finite models, this is trivially true since we cannot define an infinite set in a finite structure. It is therefore impossible to define in a finite structure the set of sentences which are true in that structure, since there are infinitely many true (and false) sentences. However, M. Mostowski [Mos01, Mos03] showed that for many interesting questions we do not need infinitely many formulae. For instance, we could ask whether restricting our arithmetical universe to some initial segment $\{0, \ldots, n\}$, is it possible to define in a given logic arithmetical truth for the finite set of first-order sentences whose Gödel numbers are in such segment. If the definition exists, we could additionally ask whether it also holds for any initial segment larger than $\{0, \ldots, n\}$.

All structures considered from now on are assumed to be finite with built-in arithmetic, i.e., the domain of a σ-structure $\mathbf{A}$ is always an initial segment $A = \{0, \ldots, n\}$ of the natural numbers, and σ contains the arithmetical sub-vocabulary $\sigma_0 = \{+, \times, \leq, 0, MAX\}$, where $+, \times, \leq$ are relation symbols and MAX, 0 are constant symbols. The relation symbols and constants in σ_0 are interpreted in the standard way. For instance, $+(x, y, z)$ is satisfied in a σ-structure iff $x + y = z$ is true for $x, y, z \in \mathbb{N}$. Given a string w, we denote as $\ulcorner w \urcorner$ the natural number corresponding to w in some appropriate Gödelization.

Definition 5.5 (Def. 2 [Mos01]). Let $R \subseteq \mathbb{N}^r$ and let $\varphi(x_1, \ldots, x_r)$ be a first-order logic formula of vocabulary σ_0 with no free variables besides $x_1, \ldots, x_r$. We say that R is *FM-represented* by φ iff for each $k \in \mathbb{N}$ there is an $m \in \mathbb{N}$ such that for each finite σ_0-structure $\mathbf{A}$ with domain $I = \{0, \ldots, n-1\}$, for some $n > m$,

$$R(a_1, \ldots, a_r) \text{ iff } \mathbf{A} \models \varphi(x_1, \ldots, x_r)[a_1, \ldots, a_r], \text{ for all } a_1, \ldots a_r \leq k.$$

R is *FM-representable* iff there is a first-order logic formula of vocabulary σ_0 which FM-represents it.

The previous definition leads us to the following question: which finite relations on natural numbers can be represented in "sufficiently large" finite models?

Theorem 5.6 (Th. 5 [Mos01]). *(FM-representability theorem). Let $r \geq 1$ and $R \subseteq \mathbb{N}^r$. Then R is FM-representable iff R is recursive with some recursively enumerable oracle.*

This implies that we may freely talk of relations connected to the syntax of logics. For instance, we can FM-represent relations such as "the formula x is the result of preceding the formula y with an existential quantifier". As all these kind of relations are decidable, we simply use the formulae which FM-represent them. However we should have in mind that a formula $\varphi(\bar{x})$ which FM-represents some relation only "tells us" whether a given tuple $\bar{a}$ is in the relation if we look at the truth value of $\varphi(\bar{a})$ in a sufficiently large model.

Once syntax can be somehow represented in finite structures, a finite model analogue to Tarski's notion of truth definition can be defined.

Definition 5.7 (Def. 1 [Mos01]). Let $\mathcal{L}$ be a logic and σ be a vocabulary. We say that the σ-formula $Tr_{\mathcal{L},\sigma}(x)$ is an *FM-truth definition* for $\mathcal{L}$ over σ iff for each $\mathcal{L}$-sentence ψ of vocabulary σ there is an $m \in \mathbb{N}$ such that for each finite σ-structure $\mathbf{A}$ with $|A| > m$, $\mathbf{A} \models \psi \leftrightarrow Tr_{\mathcal{L},\sigma}(x)[\ulcorner \psi \urcorner]$.

Thus, if there is an FM-truth definition for a logic $\mathcal{L}$, then for each sentence ψ there is an m such that in all finite structures of cardinality greater than m the FM-truth definition "tells us" whether ψ is true or not.

5.2.1 Expressibility Results via FM-Truth Definitions

The FM-representability theorem can be used to prove a diagonal lemma which roughly states that, for any formula with one free variable, there is a sentence which asserts of itself the property defined by the formula if we look at sufficiently large finite models. In analogy to the classical case, the finite version of Tarski's famous theorem on the undefinability of truth follows from this lemma.

Theorem 5.8 (Th. 2 [Mos01]). *(Tarski's theorem, version for finite models). Let $\mathcal{L}$ be a logic which is closed under negation, and under conjunction with first-order formulae. That is, for every vocabulary σ, if $\varphi \in \mathcal{L}[\sigma]$ and $\alpha \in \mathrm{FO}[\sigma]$, then there is a formula $\psi \in \mathcal{L}[\sigma]$ which is equivalent to $\neg\varphi$, and there is a formula $\psi' \in \mathcal{L}[\sigma]$ which is equivalent to $\varphi \wedge \alpha$. It follows that it is not true that there is an $\mathcal{L}$-formula which is an FM-truth definition for $\mathcal{L}$.*

As with the original Tarski's theorem, Theorem 5.8 can be used to compare the expressive power of different logics, now over finite structures. Let $\mathcal{L}$ be a logic which is closed under negation, and under conjunction with first-order formulae. The strategy consists in showing that a given logic $\mathcal{L}'$ which is known to be at least as expressive as $\mathcal{L}$, additionally defines FM-truth for $\mathcal{L}$. If that is the case, then we have shown – using the "method of FM-truth definitions" – that $\mathcal{L}'$ is strictly more expressive than $\mathcal{L}$.

It should be noted here that positive results, i.e., results of the kind "$\mathcal{L}$ defines FM-truth for $\mathcal{L}'$", are more difficult than in the infinite case. For instance, the standard proof that HO^{i+1} has a truth definition over *infinite* models for HO^{i} can be adapted to give:

Proposition 5.9 (Th. 3 [Mos01]). *On finite structures, for every order $i \geq 1$, there is an FM-truth definition for HO^{i} in HO^{i+2}.*

That is, in order to define FM-truth for HO^{i} on finite models, the order needs to be increased by 2 instead of 1. One of the main obstacles to adapt the classical proof, which increases the order by 1, is that in finite models we have no pairing function (i.e., a function which uniquely encodes two natural numbers into a single natural number). It is an open problem whether there is an FM-truth definition for HO^i in HO^{i+1}. Indeed, in [Kol05] it is pointed out that this problem is equivalent to a presumably hard problem in complexity theory.

For every $i > 1$ and $r > 0$, let $RS^i(r)$ be the class of formulae of the form

$$Q_1 X_1^{\tau_1} \dots Q_m X_m^{\tau_m} \varphi,$$

where $m \geq 0$, $\varphi \in \mathrm{HO}^{i-1}$, for $j = 1 \dots m$, Q_j is one of the quantifiers $\forall, \exists$, $\mathit{order}(\tau_1) = \dots = \mathit{order}(\tau_m) = i$, and $\mathit{arity}(\tau_1) + \dots + \mathit{arity}(\tau_m) \leq r$. These classes were defined in [Mos01] where they were called "classes of restricted size"[1]. It is stated there that for $i > 1$ and $r > 0$, there is an FM-truth definition for $RS^i(r)$ in $RS^i(r + c)$ for some $c \in \mathbb{N}$. No proof was given, though. Furthermore, note that according to its definition each class $RS^i(r)$ of restricted size of order i contains the whole HO^{i-1} fragment. Therefore, if the result is correct, then in particular there is an FM-truth definition for HO^i in HO^{i+1}. But, as we mentioned earlier, this is an open problem and presumable a difficult one.

An alternative to avoid this problem could be to use a restricted notion of arity, similar to our notion of maximal-arity (see Definition 3.1), in the definition of the classes of restricted size. It would be also necessary to restrict the arity of the variables of order less than i. We believe that, the results on hierarchies of restricted size that could be obtained in that way, would be essentially equivalent to our results on the HAA^i hierarchies.

In [Kol04b, Kol05], Kolodziejczyk was able to characterize, using the notion of data complexity, the logics for which the Σ^i_m classes define FM-truth. Recall that, *data complexity* for a logic $\mathcal{L}$ is said to be in a complexity class $\mathcal{C}$ if every class $\mathcal{K}$ definable in $\mathcal{L}$ is in $\mathcal{C}$. The characterization is based on the alternating complexity classes $\Sigma_m-\mathrm{TIME}$ and $\Pi_m-\mathrm{TIME}$ (see Section 2.3.1 for details).

Theorem 5.10 (Th. 5.2 [Kol04b])**.** *Let $\mathcal{L}$ be a logic and σ a relational vocabulary. For $i, m \geq 1$, it holds that $\Sigma^i_m[\sigma]$ defines FM-truth for $\mathcal{L}[\sigma]$ iff there is a number k such that data complexity*[2] *for $\mathcal{L}[\sigma]$ is in $\Sigma_m-\mathrm{TIME}(\exp_{i-1}(n^k))$. Here n denotes the*

[1]Note that, what M. Mostowski defines in page 514 in [Mos01] as the *size* of a type is exactly what we call in Definition 3.1 the *arity* of a type.

[2]Kolodziejczyk uses in his work the term "model checking" instead of "data complexity". Since the definition given in [Kol04b] of model checking (see third paragraph in page 186) coincides with

size of the structure. For Π^i_m and Π_m−TIME($\exp_{i-1}(n^k)$), the analogous situation also holds.

It has been pointed out in [Kol04b] that, for $r, m \geq 1$, data complexity for $AA\Sigma^1(r,m)$ is in Σ_m−TIME(n^r) and for $AA\Pi^1(r,m)$ is in Π_m−TIME(n^r). Then, for each fragment $AA^1(r,m)$ of second-order logic, data complexity is in Σ_{m+1}−TIME(n^r). Therefore, for every vocabulary σ and $m, r \geq 1$, it holds that $\Sigma^1_{m+1}[\sigma]$ defines FM-truth for $AA^1(r,m)[\sigma]$.

The fact that, for $r, m \geq 1$, Σ^1_{m+1} defines FM-truth for $AA^1(r,m)$, is not enough to conclude that, for every $r, m \geq 1$, Σ^1_{m+1} is strictly more expressive than $AA^1(r,m)$ using the method of FM-truth definitions. We also need to show that $AA^1(r,m)$ does not define FM-truth for itself. Unfortunately, we *cannot* apply Theorem 5.8 here, since the formulae in $AA^1(r,m)$ are not closed under conjunction with first-order formulae.

5.3 A Sufficient Condition for the Properness of the HAA^1 Hierarchy

FM-truth definitions were used in [Kol04a] to prove that, if for every vocabulary σ there is a fixed k such that data complexity for FO[σ] is in NTIME(n^k), then the Σ^1_1 arity hierarchy is strict (see Proposition 3.19 in Chapter 3). Recall that this hierarchy is known to be strict if we allow vocabularies of arbitrary arity [Ajt83], but its strictness over vocabularies of a fixed arity is still open.

The question of whether for every vocabulary σ there is a fixed k such that data complexity for FO[σ] is in NTIME(n^k), is an important open question in descriptive complexity (see [Imm99], open problem 7.13).

By Theorem 5.10, if data complexity for FO over any fixed vocabulary is in NTIME(n^k), where k may depend on σ, then Σ^1_1 defines FM-truth for FO over any vocabulary σ. Not surprisingly, these FM-truth definitions can be transformed into Σ^1_1 FM-truth definitions for $HAA\Sigma^1(r,1)$, for any $r \geq 1$ and over any vocabulary σ. Indeed, this can be generalized to the whole HAA^1 hierarchy of second-order logic.

the definition of what is more commonly known as data complexity (see Definition 6.1 in [Lib04]), we use this last term.

Proposition 5.11. *If for every vocabulary σ there is a fixed k such that data complexity for* $\mathrm{FO}[\sigma]$ *is in* $\mathrm{NTIME}(n^k)$*, then for any vocabulary σ and arity r there is a k_r such that data complexity for $HAA\Sigma^1(r, m)$ over σ is in class $\Sigma_m{-}\mathrm{TIME}(n^{k_r})$, and data complexity for $HAA\Pi^1(r, m)$ over σ is in the class $\Pi_m{-}\mathrm{TIME}(n^{k_r})$.*

Proof. (Sketch). We show that given our assumption, for any vocabulary σ and arity r there is a k_r such that data complexity for $HAA\Sigma^1(r, m)$ over σ is in $\Sigma_m{-}\mathrm{TIME}(n^{k_r})$. The proof for $HAA\Pi^1(r, m)$ is completely analogous.

Let σ be a vocabulary and let φ be a $HAA\Sigma^1(r, m)[\sigma]$ sentence. Given a σ-structure $\mathbf{A}$, to check whether $\mathbf{A} \models \varphi$, an alternating Σ_m machine M_φ works as follows:

- Starting in an existential state, M_φ guesses relations corresponding to the variables in the initial block of existential quantifiers in φ. Recall that these quantifiers are of order 2 and arity at most r. Thus, using the well-known encoding of relations over $\{0, \dots, n-1\}$ as binary strings, this amounts to writing in the tape a fixed number of binary strings of length bounded by $O(r \cdot n^r \cdot \log n)$.

- M_φ then switches to a universal state and guesses relations corresponding to the variables in the first block of universal quantifiers, then switches back to an existential state, and so on.

- Once the witnesses for all the second-order quantifiers have been guessed, it only remains to check the truth of the first-order part ψ of φ. Note that, as we assume that data complexity for FO[σ] is in NTIME(n^k), then it must also be in co-NTIME(n^k), since FO is closed under negation. Thus, if M_φ is in an existential state, it checks the truth of ψ in NTIME(n^k). Otherwise, if M_φ is in an universal state, it checks the truth of ψ in co-NTIME(n^k).

□

Corollary 5.12. *If for every vocabulary σ there is a fixed k such that data complexity for* $\mathrm{FO}[\sigma]$ *is in* $\mathrm{NTIME}(n^k)$*, then for $r, m \geq 1$, there is a $c \in \mathbb{N}$ such that $HAA^1(r, m) \subset HAA^1(r + c, m + 1)$.*

Proof. Since every level of the $HAA^1(r, m)$ hierarchy is closed under negation, and conjunction with first-order logic formulae, it follows from Theorem 5.8 that, for any vocabulary σ and $r, m \geq 1$, it is *not* true that there is a formula in $HAA^1(r, m)$ which is an FM-truth definition for $HAA^1(r, m)$ over σ.

On the other hand, by the previous result and by Theorem 5.10, if for every vocabulary σ there is a fixed k such that data complexity for FO[σ] is in NTIME(n^k), then for every vocabulary σ and $r, m \geq 1$, Σ^1_m defines FM-truth for $HAA\Sigma^1(r, m)$ over σ and Π^1_m defines FM-truth for $HAA\Pi^1(r, m)$ over σ. Since $HAA^1(r, m) = HAA\Sigma^1(r, m) \cup HAA\Pi^1(r, m)$ and $\Pi^1_m \subseteq \Sigma^1_{m+1}$, it follows that under our assumption, for every vocabulary σ and $r, m \geq 1$, Σ^1_{m+1} defines FM-truth for $HAA^1(r, m)[\sigma]$; and thus there is a formula $Tr_{HAA^1(r,m),\sigma}(x)$ in $\Sigma^1_{m+1}[\sigma]$ which is an FM-truth definition for $HAA^1(r, m)$ over σ. Let s be the maximal-arity of the formula $Tr_{HAA^1(r,m),\sigma}(x)$, it follows that $Tr_{HAA^1(r,m),\sigma}(x)$ is also in $HAA^1(r + c, m + 1)[\sigma]$, for $c \in \mathbb{N}$ and $r + c \geq s$. Note that $\Sigma^1_{m+1} \subseteq \bigcup_{r \in \mathbb{N}} HAA^1(r, m + 1)$. Hence, if for every vocabulary σ there is a fixed k such that data complexity for FO[σ] is in NTIME(n^k), then for every vocabulary σ and $r, m \geq 1$, there is a $c \in \mathbb{N}$ such that $HAA^1(r + c, m + 1)$ defines FM-truth for $HAA^1(r, m)$ over σ. $\square$

It is important to note here that it is not known whether, for $r, m \geq 1$, *AUTO-SAT*($HAA^1(r, m)[\rho]$) is definable in $HAA^1(r + c_1, m + c_2)[\rho]$, for some $c_1, c_2 \in \mathbb{N}$. In Proposition 5.3 we show that *AUTOSAT*($HAA^i(r, m)[\rho]$) is definable in $HAA^i(r + c(r), m + 2)[\rho]$ where $c(r) = 1$ for $r > 1$ and $c(r) = 2$ for $r = 1$, but only for $i \geq 2$.

Chapter 6

On the Complexity of Sentences which Satisfy Themselves

The set $AUTOSAT(F)$ of the sentences of a given logic F which encoded as finite structures satisfy themselves, was also studied in [MP96] from the point of view of its complexity. Let Σ^0_m denote the set of first-order formulae in prenex normal form which start with an existential block of quantifiers and have up to m alternating blocks. They proved the following result.

Theorem 6.1 ([MP96]). *Under polynomial-time reductions, it holds that:*

i. $AUTOSAT(\mathrm{FO})$ *is complete for* PSPACE.

ii. *For every* $r, m \geq 1$, $AUTOSAT(HAA^1(r,m))$ *is complete for* PSPACE.

iii. *For every* $m \geq 1$, $AUTOSAT(\Sigma^0_m)$ *is complete for the class* Σ^p_m *of the polynomial-time hierarchy.*

The fact that for first-order logic $AUTOSAT$ is hard for PSPACE, was proved by reduction from the well known problem of *quantified satisfiability* (QSAT) to $AUTOSAT(\mathrm{FO})$. Note that QSAT, also known as quantified boolean formulae (QBF), is complete for PSPACE (see [Pap94, BDG95] among other sources). Since for every $r, m \geq 1$, it holds that $\mathrm{FO} \subset HAA^1(r,m)$, the same reduction from QSAT to $AUTOSAT(\mathrm{FO})$, also applies from QSAT to $AUTOSAT(HAA^1(r,m))$, for every $r, m \geq 1$. Regarding $AUTOSAT(\Sigma^0_m)$, they gave a polynomial-time reduction from QSAT_m (*quantified satisfiability with* m *alternations of quantifiers*). Since QSAT_m

is complete for the corresponding level Σ^p_m of the polynomial-time hierarchy, this proves that $AUTOSAT(\Sigma^0_m)$ is hard for Σ^p_m.

In this chapter, we extend this result establishing the complexity of $AUTOSAT$ for the prenex fragments Σ^1_m of second-order logic. Notice that the argument used in [MP96] to prove that $AUTOSAT(HAA^1(r,m))$ is in PSPACE, does not apply to $AUTOSAT(\Sigma^1_m)$. The main obstacle is that the arity of the variables which appear in the formulae in Σ^1_m is not bounded by a fixed r as in the case of $HAA^1(r,m)$.

We prove instead that for every m, $AUTOSAT(\Sigma^1_m)$ is in $\bigcup_{c\in\mathbb{N}} \text{NTIME}(2^{n^c})^{\Sigma^p_{m-1}}$. Since for every $m \geq 1$, we also give in this chapter a polynomial-time reduction from a complete problem in the fragment Σ^2_m of third-order logic to $AUTOSAT(\Sigma^1_m)$ and, by Theorem 3.8, Σ^2_m captures $\bigcup_{c\in\mathbb{N}} \text{NTIME}(2^{n^c})^{\Sigma^p_{m-1}}$, we get the following result.

Theorem 6.2. *For every $m \geq 1$, it holds that $AUTOSAT(\Sigma^1_m)$ is complete for Σ^2_m under polynomial-time reductions.*

We do not pursue this matter further in this chapter, but we would like to mention that we strongly believe the previous result also holds for every order $i > 1$. We do not see any obstacle to extending our strategy to prove that for every $i, m \geq 1$, it holds that $AUTOSAT(\Sigma^i_m)$ is complete for Σ^{i+1}_m under polynomial-time reductions. Among other things, in the last chapter of this dissertation we explain how we think that our strategy could be extended to prove this conjecture.

6.1 A Complete Problem for Higher-Order Logics

We define now the problem which we use in the next section to show that *AUTO-SAT*(Σ^1_m) is hard for the fragment Σ^2_m of third-order logic. This problem was introduced in [HT05, HT06a]. It is a generalization for higher-order logics of the well known QSAT$_m$ problem which as we mentioned at the beginning of this chapter, is known to be complete for the corresponding level Σ^p_m of the polynomial-time hierarchy. And hence, by the correspondence between the prenex fragments of second-order logic and the polynomial-time hierarchy (see Theorem 2.11), it is also complete for the Σ^1_m fragment of second-order logic.

Definition 6.3. Let $\mathbf{B}$ be the *Boolean model* $\langle\{0,1\}, 0^{\mathbf{B}}, 1^{\mathbf{B}}\rangle$ of the *Boolean vocabulary* $\{0,1\}$, i.e., a vocabulary with no relation (or function) symbols, and which has two constant symbols which are interpreted by the two elements, respectively.

For $i, m \geq 1$, we denote Σ^i_m-Th($\mathbf{B}$), the Σ^i_m theory of the Boolean model $\mathbf{B}$, i.e., Σ^i_m-Th($\mathbf{B}$) is the set of Σ^i_m-formulae of the Boolean vocabulary which are satisfied by $\mathbf{B}$.

Theorem 6.4 ([HT05, HT06a])**.** *For $i, m \geq 1$, it holds that Σ^i_m-Th($\boldsymbol{B}$) is complete for Σ^{i+1}_m under polynomial-time reductions.*

Note that in particular, for each $m \geq 1$, the Σ^1_m theory of the Boolean model $\mathbf{B}$ is complete for the fragment Σ^2_m of third-order logic.

Although we do not use it here, we should mention that as shown also in [HT05, HT06a], there are complete problems for all Σ^i_m fragments even under *quantifier-free first-order reductions*, and without assuming the existence of an order in the input structures in such reductions. Using those problems as the interpretation of (relativized) Lindström quantifiers, they showed that every fragment Σ^i_m of higher-order logics can be captured with a first-order logic Lindström quantifier. Moreover, they gave a normal form showing that each Σ^i_m sentence is equivalent to a single occurrence of the quantifier plus a tuple of quantifier-free first-order formulae.

6.2 $AUTOSAT(\Sigma^1_m)$ is hard for Σ^2_m

By a polynomial-time reduction from Σ^1_m-Th($\mathbf{B}$) to $AUTOSAT(\Sigma^1_m[\rho])$, we show in this section that $AUTOSAT(\Sigma^1_m)$ is hard for Σ^2_m. Here ρ is the vocabulary used in the definition of $AUTOSAT$ (see Definition 4.4).

First, we fix a translation from second-order formulae of Boolean vocabulary to second-order formulae of vocabulary ρ.

Definition 6.5. Let φ be a any second-order formula of the Boolean vocabulary, we denote $\hat{\varphi}$ the following second-order formula of vocabulary ρ:

$$\exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \varphi'(w_0, w_1))$$

where $\psi_{min}(x) \equiv \forall y(\neg y < x)$, $\psi_{max}(x) \equiv \forall y(\neg x < y)$, w_0 and w_1 are first order variables which do not appear in φ, and $\varphi'(w_0, w_1)$ is defined by induction on φ, as follows:

- Let φ be an atomic formula.

- If φ has the form $s = t$, then $\varphi'(w_0, w_1)$ is

$$f(s) = f(t) \wedge (f(s) = w_0 \vee f(s) = w_1) \wedge (f(t) = w_0 \vee f(t) = w_1)$$

  where $f(t)$ ($f(s)$) denotes w_0, w_1 or t (s), depending on whether t (s) is the constant symbol 0, 1 or a first-order variable, respectively.

- If φ has the form $V(x_1, \ldots, x_r)$, then $\varphi'(w_0, w_1)$ is

$$V(x_1, \ldots, x_r) \wedge (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)$$

• Let φ be a compound formula.

- If φ has the form $\neg\psi$, then $\varphi'(w_0, w_1)$ is $\neg\psi'(w_0, w_1)$.
- If φ has the form $\psi \vee \gamma$, then $\varphi'(w_0, w_1)$ is $\psi'(w_0, w_1) \vee \gamma'(w_0, w_1)$.
- If φ has the form $\psi \wedge \gamma$, then $\varphi'(w_0, w_1)$ is $\psi'(w_0, w_1) \wedge \gamma'(w_0, w_1)$.
- If φ has the form $\exists x(\psi)$, then $\varphi'(w_0, w_1)$ is

$$\exists x((x = w_0 \vee x = w_1) \wedge \psi'(w_0, w_1))$$

- If φ has the form $\forall x(\psi)$, then $\varphi'(w_0, w_1)$ is

$$\forall x((x = w_0 \vee x = w_1) \rightarrow \psi'(w_0, w_1))$$

- If φ has the form $\exists X(\psi)$, then $\varphi'(w_0, w_1)$ is

$$\exists X(\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge$$

$$(x_r = w_0 \vee x_r = w_1)) \wedge \psi'(w_0, w_1))$$

- If φ has the form $\forall X(\psi)$, then $\varphi'(w_0, w_1)$ is

$$\forall X(\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge$$

$$(x_r = w_0 \vee x_r = w_1)) \rightarrow \psi'(w_0, w_1))$$

Now, for each valuation v on the Boolean model $\mathbf{B}$ and each ρ-structure $\mathbf{A}$ with at least two elements, we define a corresponding valuation v_A on $\mathbf{A}$. We define v_A in such a way that for every second-order formula φ, it holds that $\mathbf{B}, v \models \varphi$ iff $\mathbf{A}, v_A \models \hat{\varphi}$.

Definition 6.6. We define $\mathcal{B}_\rho|_{\geq 2} = \{\mathbf{A}_i \in \mathcal{B}_\rho : |dom(\mathbf{A}_i)| \geq 2\}$, i.e., $\mathcal{B}_\rho|_{\geq 2}$ is the class of ρ-structures with at least two elements. We denote $min(\mathbf{A})$ and $max(\mathbf{A})$ the least and last element, respectively, in the linear order $<^{\mathbf{A}}$ of the ρ-structure $\mathbf{A}$. Let v be a valuation on $\mathbf{B}$, and let $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, we define the valuation v_A on $\mathbf{A}$, as follows:

- If x is a first-order variable, then

$$v_A(x) = \begin{cases} min(\mathbf{A}) & \text{if } x \text{ is the variable } w_0 \\ max(\mathbf{A}) & \text{if } x \text{ is the variable } w_1 \\ min(\mathbf{A}) & \text{if } x \text{ is a variable other than } w_0 \text{ and } w_1, \text{ and } v(x) = 0^{\mathbf{B}} \\ max(\mathbf{A}) & \text{if } x \text{ is a variable other than } w_0 \text{ and } w_1, \text{ and } v(x) = 1^{\mathbf{B}} \end{cases}$$

- If X is a second-order variable of arity r, then v_A assigns to X an r-ary relation $R \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$ such that $(a_1, \ldots, a_r) \in R$ iff $(g(a_1), \ldots, g(a_r)) \in v(X)$ for $g(a_i) = 0^{\mathbf{B}}$ if $a_i = min(\mathbf{A})$ and $g(a_i) = 1^{\mathbf{B}}$ if $a_i = max(\mathbf{A})$.

We show next that a valuation v on the Boolean model satisfies a second-order formula φ iff in every ρ-structure $\mathbf{A}$ with at least two elements the corresponding valuation v_A satisfies $\hat{\varphi}$. We do that in two steps. First, we show in Lemma 6.7 the following: if there is a ρ-structure $\mathbf{A}$ with at least two elements such that $\mathbf{A}, v_A \models \hat{\varphi}$, then for every ρ-structure $\mathbf{A}'$ with at least two elements, $\mathbf{A}', v_{A'} \models \hat{\varphi}$. Note that the converse is trivially true since $\mathcal{B}_\rho|_{\geq 2}$ is *not* empty. Thus, if for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$, then there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}, v_A \models \hat{\varphi}$. Secondly, we show in Lemma 6.8 that a valuation v on the Boolean model satisfies a second-order formula φ iff there is a ρ-structure $\mathbf{A}$ with at least two elements such that the corresponding valuation v_A satisfies $\hat{\varphi}$.

Lemma 6.7. *Let φ be any second-order formula of the Boolean vocabulary, let v be a valuation on $\boldsymbol{B}$, let $\hat{\varphi}$ be as in Definition 6.5, and let $\mathcal{B}_\rho|_{\geq 2}$ and v_A be as in Definition 6.6. If there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}, v_A \models \hat{\varphi}$, then for every $\boldsymbol{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, it holds that $\boldsymbol{A}_i, v_{A_i} \models \hat{\varphi}$.*

Proof. We proceed by induction on φ. Consider first the case of atomic formulae:

- Let φ be $s = t$. Since $\mathbf{A}, v_A \models \hat{\varphi}$, we know that $v_A(f(s)) = v_A(f(t)) = min(\mathbf{A})$ or $v_A(f(s)) = v_A(f(t)) = max(\mathbf{A})$. Let's assume w.l.o.g. that $v_A(f(s)) = v_A(f(t)) = min(\mathbf{A})$. It follows that either $f(s)$ is w_0 or $v(f(s)) = 0$. In either case, we get that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $v_{A_i}(f(s)) = min(\mathbf{A}_i)$. By the same argument, we also get that $v_{A_i}(f(t)) = min(\mathbf{A}_i)$. Therefore, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, it holds that $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

- Let φ be $V(t_1, \ldots, t_r)$. Since $\mathbf{A}, v_A \models \hat{\varphi}$, we have $(v_A(f(t_1)), \ldots, v_A(f(t_r))) \in v_A(V)$ and, for every $1 \leq j \leq r$, either $v_A(f(t_j)) = min(\mathbf{A})$ or $v_A(f(t_j)) = max(\mathbf{A})$. Let us assume w.l.o.g. that for every $1 \leq j \leq r$, $v_A(f(t_j)) = min(\mathbf{A})$. It follows that, for every $1 \leq j \leq r$, either $f(t_j)$ is w_0 or $v(f(t_j)) = 0$. In either case, we get that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $v_{A_i}(f(t_j)) = min(\mathbf{A}_i)$. It also follows that,
$$\underbrace{(min(\mathbf{A}), \ldots, min(\mathbf{A}))}_{r} \in v_A(V), \quad \text{and therefore that}$$
$$\underbrace{(0^{\mathbf{B}}, \ldots, 0^{\mathbf{B}})}_{r} \in v(V) \quad \text{and} \quad \underbrace{(min(\mathbf{A}_i), \ldots, min(\mathbf{A}_i))}_{r} \in v_{A_i}(V),$$
Thus, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, it holds that $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

Assume then as an induction hypothesis that for every sub-formula ψ of φ and every valuation v on $\mathbf{B}$, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}, v_A \models \hat{\psi}$, then for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, it holds that $\mathbf{A}_i, v_{A_i} \models \hat{\psi}$.

- Let φ be $\neg\psi$. Towards a contradiction, let us assume that there is an $\mathbf{A}_x \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}_x, v_{A_x} \not\models \hat{\varphi}$. Thus, $\mathbf{A}_x, v_{A_x} \models \neg(\exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \neg\psi'(w_0, w_1)))$. By the well known relation between universal and existential quantifiers and by De Morgan laws, $\mathbf{A}_x, v_{A_x} \models \forall w_0 w_1 (\neg\psi_{min}(w_0) \vee \neg\psi_{max}(w_1) \vee \neg w_0 \neq w_1 \vee \psi'(w_0, w_1))$. It follows that for every valuation v'_{A_x}, which is $\{w_0, w_1\}$-equivalent to v_{A_x}, we have $\mathbf{A}_x, v'_{A_x} \models \neg\psi_{min}(w_0) \vee \neg\psi_{max}(w_1) \vee \neg w_0 \neq w_1 \vee \psi'(w_0, w_1)$. Since v_{A_x} is $\{w_0, w_1\}$-equivalent to itself and $\mathbf{A}_x, v_{A_x} \models \psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1$, it is necessarily the case that $\mathbf{A}_x, v_{A_x} \models \psi'(w_0, w_1)$. Therefore, $\mathbf{A}_x, v_{A_x} \models \hat{\psi}$ and by inductive hypothesis for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\psi}$. But then $\mathbf{A}, v_A \models \hat{\psi}$, and consequently $\mathbf{A}, v_A \not\models \hat{\varphi}$, which contradicts our hypothesis.

- Let φ be $\psi \vee \gamma$. Since $\mathbf{A}, v_A \models \hat{\varphi}$, we have that $\mathbf{A}, v_A \models \hat{\psi}$ or $\mathbf{A}, v_A \models \hat{\gamma}$. By induction hypothesis, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\psi}$ or, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\gamma}$. Thus, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\psi} \vee \hat{\gamma}$, or equivalently $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

- Let φ be $\psi \wedge \gamma$. Since $\mathbf{A}, v_A \models \hat{\varphi}$, we have that $\mathbf{A}, v_A \models \hat{\psi}$ and $\mathbf{A}, v_A \models \hat{\gamma}$. By induction hypothesis, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\psi}$ and, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\gamma}$. Thus, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\psi} \wedge \hat{\gamma}$, or equivalently $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

- Let φ be $\exists x(\psi)$. If $\mathbf{A}, v_A \models \varphi$, then there is a valuation u, which is $\{x\}$-equivalent to v_A, such that $\mathbf{A}, u \models \exists w_0 w_1(\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge (x = w_0 \vee x = w_1) \wedge \psi'(w_0, w_1))$. Since u is $\{x\}$-equivalent to v_A and $u(x) = v_A(w_0)$ or $u(x) = v_A(w_1)$, there is a valuation v' on $\mathbf{B}$, which is $\{x\}$-equivalent to v, such that the corresponding valuation v'_A on $\mathbf{A}$ is precisely u. Given that $\mathbf{A}, v'_A \models \hat{\psi}$, it follows by inductive hypothesis that, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v'_{A_i} \models \hat{\psi}$. Since v' is $\{x\}$-equivalent to v, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, we have that v'_{A_i} is $\{x\}$-equivalent to v_{A_i}. It follows that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

- Let φ be $\forall x(\psi)$. If $\mathbf{A}, v_A \models \varphi$ then for every valuation u, which is $\{x\}$-equivalent to v_A, $\mathbf{A}, u \models \exists w_0 w_1(\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge ((x = w_0 \vee x = w_1) \rightarrow \psi'(w_0, w_1)))$. Moreover, if $u(x) = max(\mathbf{A})$ or $u(x) = min(\mathbf{A})$, then $\mathbf{A}, u \models \hat{\psi}$, and there is a valuation v' on $\mathbf{B}$, which is $\{x\}$-equivalent to v, and such that the corresponding valuation v'_A on $\mathbf{A}$, is precisely u. On the other hand, if v' is a valuation on $\mathbf{B}$, which is $\{x\}$-equivalent to v, then for every $A_i \in \mathcal{B}_\rho|_{\geq 2}$, the corresponding valuation v'_{A_i} on $\mathbf{A}_i$, is $\{x\}$-equivalent to v_{A_i} and $v'_{A_i}(x) = max(\mathbf{A})$ or $v'_{A_i}(x) = min(\mathbf{A})$. Therefore, if v' is a valuation on $\mathbf{B}$, which is $\{x\}$-equivalent to v, then $\mathbf{A}, v'_A \models \hat{\psi}$, and by induction hypothesis, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $A_i, v'_{A_i} \models \hat{\psi}$. Since then, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, if u is a valuation on $\mathbf{A}_i$ which is $\{x\}$-equivalent to v_{A_i}, then $\mathbf{A}_i, u \models \exists w_0 w_1(\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1)$, and furthermore if $u(x) = max(\mathbf{A}_i)$ or $u(x) = min(\mathbf{A}_i)$, then $\mathbf{A}_i, u \models \psi'(w_0, w_1)$, we get that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

- Let φ be $\exists X(\psi)$. If $\mathbf{A}, v_A \models \varphi$, then there is a valuation u, which is $\{X\}$-equivalent to v_A, such that $\mathbf{A}, u \models \exists w_0 w_1(\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \wedge \psi'(w_0, w_1))$. Since u is $\{X\}$-equivalent to v_A and $u(X) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$, there is a valuation v' on $\mathbf{B}$, which is $\{X\}$-equivalent to v, such that the corresponding valuation v'_A on $\mathbf{A}$ is precisely u. Given that $\mathbf{A}, v'_A \models \hat{\psi}$, it follows by inductive hypothesis that, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v'_{A_i} \models \hat{\psi}$. Since

v' is $\{X\}$-equivalent to v, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, we have that v'_{A_i} is $\{X\}$-equivalent to v_{A_i}. It clearly follows that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

- Let φ be $\forall X(\psi)$. If $A, v_A \models \varphi$ then for every valuation u, which is $\{X\}$-equivalent to v_A, it holds that $\mathbf{A}, u \models \exists w_0 w_1(\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \rightarrow \psi'(w_0, w_1))$. Moreover, if $u(X) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$, then $\mathbf{A}, u \models \hat{\psi}$, and there is a valuation v' on $\mathbf{B}$, which is $\{X\}$-equivalent to v, and such that the corresponding valuation v'_A on $\mathbf{A}$, is precisely u. On the other hand, if v' is a valuation on $\mathbf{B}$, which is $\{X\}$-equivalent to v, then for every $A_i \in \mathcal{B}_\rho|_{\geq 2}$, the corresponding valuation v'_{A_i} on $\mathbf{A}_i$, is $\{X\}$-equivalent to v_{A_i} and $v'_{A_i}(x) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$. Therefore, if v' is a valuation on $\mathbf{B}$, which is $\{X\}$-equivalent to v, then $\mathbf{A}, v'_A \models \hat{\psi}$, and by induction hypothesis, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $A_i, v'_{A_i} \models \hat{\psi}$. Since then, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, if u is a valuation on $\mathbf{A}_i$ which is $\{X\}$-equivalent to v_{A_i}, then $\mathbf{A}_i, u \models \exists w_0 w_1(\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1)$, and furthermore if $u(X) \subseteq \{min(\mathbf{A}_i), max(\mathbf{A}_i)\}^r$, then $\mathbf{A}_i, u \models \psi'(w_0, w_1)$, we get that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\varphi}$.

$\square$

Lemma 6.8. *Let φ be any second-order formula of the Boolean vocabulary, let v be a valuation on $\boldsymbol{B}$, let $\hat{\varphi}$ be as in Definition 6.5, and let $\mathcal{B}_\rho|_{\geq 2}$ and v_A be as in Definition 6.6. Then, it holds that $\boldsymbol{B}, v \models \varphi$ iff there is an $\boldsymbol{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\boldsymbol{A}, v_A \models \hat{\varphi}$.*

Proof. We use induction on the formula φ. Consider first the case of atomic formulae:

- Let φ be $s = t$. Let $\mathbf{A}$ be in $\mathcal{B}_\rho|_{\geq 2}$. If $\mathbf{B}, v \models \varphi$, then by definition of v_A, it clearly follows that $v_A(f(s)) = v_A(f(t))$ and thus $\mathbf{A}, v_A \models \hat{\varphi}$.

  On the other hand, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then there is a valuation v'_A, which is $\{w_0, w_1\}$-equivalent to v_A, such that $\mathbf{A}, v'_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \varphi'(w_0, w_1)$. Since $\mathbf{A}, v'_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1)$, by definition $v_A(w_0) = min(\mathbf{A})$ and $v_A(w_1) = max(\mathbf{A})$, and v'_A is $\{w_0, w_1\}$-equivalent v_A, it turns out that v_A and v'_A are exactly the same valuation, and consequently that $\mathbf{A}, v_A \models \varphi'(w_0, w_1)$. Therefore, either $v_A(f(s)) =$

$v_A(f(t)) = min(\mathbf{A})$ or $v_A(f(s)) = v_A(f(t)) = max(\mathbf{A})$. Let us assume w.l.o.g. that $v_A(f(s)) = v_A(f(t)) = min(\mathbf{A})$. Since $v_A(f(s)) = min(\mathbf{A})$ then by definition of v_A, either $f(s)$ is w_0 and thus s is the constant symbol 0, or $f(s)$ is s –a first-order variable other than w_0 and w_1– and $v(s) = 0^{\mathbf{B}}$. Equivalently, either $f(t)$ is w_0 and t is the constant symbol 0, or $f(t)$ is t and $v(t) = 0^{\mathbf{B}}$. Thus $\mathbf{B}, v \models \varphi$.

- Let φ be $V(x_1, \dots, x_r)$. Let $\mathbf{A}$ be in $\mathcal{B}_\rho|_{\geq 2}$. If $\mathbf{B}, v \models \varphi$, then we know that $(v(x_1), \dots, v(x_r)) \in v(V)$, and thus by definition of v_A, we also know that $(v_A(x_1), \dots, v_A(x_r)) \in v_A(V)$. It clearly follows that $\mathbf{A}, v_A \models \hat{\varphi}$.

  Conversely, let us assume that there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$. It follows that, there is a valuation v'_A, which is $\{w_0, w_1\}$-equivalent to v_A, such that $\mathbf{A}, v'_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \varphi'(w_0, w_1)$. Since $\mathbf{A}, v'_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1)$, by definition $v_A(w_0) = min(\mathbf{A})$ and $v_A(w_1) = max(\mathbf{A})$, and v'_A is $\{w_0, w_1\}$-equivalent v_A, it turns out that v_A and v'_A are exactly the same valuation, and consequently that $\mathbf{A}, v_A \models \varphi'(w_0, w_1)$. Therefore, $(v_A(x_1), \dots, v_A(x_r)) \in v_A(V)$ and for every $1 \leq i \leq r$, either $v_A(x_i) = min(\mathbf{A})$ or $v_A(x_i) = max(\mathbf{A})$. Let us assume w.l.o.g. that for every $1 \leq i \leq r$, $v_A(x_i) = min(\mathbf{A})$. It follows that,

$$\underbrace{(min(\mathbf{A}), \dots, min(\mathbf{A}))}_{r} \in v_A(V), \text{ and thus } \underbrace{(0^{\mathbf{B}}, \dots, 0^{\mathbf{B}})}_{r} \in v(V).$$

  It also follows that for every $1 \leq i \leq r$, $v(x_i) = 0^{\mathbf{B}}$. Therefore $\mathbf{B}, v \models \varphi$.

Assume then as an induction hypothesis that for every sub-formula ψ of φ and valuation v on $\mathbf{B}$, $\mathbf{B}, v \models \psi$ iff there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}, v_A \models \hat{\psi}$.

- Let φ be $\neg\psi$. If $\mathbf{B}, v \models \varphi$, then $\mathbf{B}, v \not\models \psi$. By inductive hypothesis, we obtain that for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \not\models \hat{\psi}$. Thus, let $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}, v_A \models \neg(\exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \psi'(w_0, w_1)))$. By the well known relation between universal and existential quantifiers and by De Morgan laws, $\mathbf{A}, v_A \models \forall w_0 w_1 (\neg\psi_{min}(w_0) \vee \neg\psi_{max}(w_1) \vee \neg w_0 \neq w_1 \vee \neg\psi'(w_0, w_1))$. It follows that for every valuation v'_A, which is $\{w_0, w_1\}$-equivalent to v_A, we have $\mathbf{A}, v'_A \models \neg\psi_{min}(w_0) \vee \neg\psi_{max}(w_1) \vee \neg w_0 \neq w_1 \vee \neg\psi'(w_0, w_1)$. Since v_A is $\{w_0, w_1\}$-equivalent to itself and $\mathbf{A}, v_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1$, it is necessarily the case that $\mathbf{A}, v_A \models \neg\psi'(w_0, w_1)$. It clearly follows that $\mathbf{A}, v_A \models \hat{\varphi}$.

Conversely, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then there is a valuation v'_A, which is $\{w_0, w_1\}$-equivalent to v_A, such that $\mathbf{A}, v'_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \neg\psi'(w_0, w_1)$. Since $\mathbf{A}, v'_A \models \psi_{min}(w_0) \wedge \psi_{max}(w_1)$, by definition $v_A(w_0) = min(\mathbf{A})$ and $v_A(w_1) = max(\mathbf{A})$, and v'_A is $\{w_0, w_1\}$-equivalent to v_A, it turns out that v_A and v'_A are exactly the same valuation, and consequently that $\mathbf{A}, v_A \models \neg\psi'(w_0, w_1)$. Therefore by Lemma 6.7, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \not\models \hat{\psi}$, and by inductive hypothesis, $\mathbf{B}, v \not\models \psi$. It follows that $\mathbf{B}, v \models \varphi$.

- Let φ be $\psi \vee \gamma$. If $\mathbf{B}, v \models \varphi$ then $\mathbf{B}, v \models \psi$ or $\mathbf{B}, v \models \gamma$. If $\mathbf{B}, v \models \psi$, then by induction hypothesis there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\psi}$. Otherwise, if $\mathbf{B}, v \not\models \psi$, then $\mathbf{B}, v \models \gamma$, and by induction hypothesis there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\gamma}$. In either case there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}, v_A \models \hat{\psi} \vee \hat{\gamma}$, or equivalently such that $\mathbf{A}, v_A \models \hat{\varphi}$.

  On the other hand, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then $\mathbf{A}, v_A \models \hat{\psi}$ or $\mathbf{A}, v_A \models \hat{\gamma}$. If $\mathbf{A}, v_A \models \hat{\psi}$, then by induction hypothesis $\mathbf{B}, v \models \psi$. Otherwise, if $\mathbf{A}, v_A \not\models \hat{\psi}$, then $\mathbf{A}, v_A \models \hat{\gamma}$, and by induction hypothesis $\mathbf{B}, v \models \gamma$. In either case, $\mathbf{B}, v \models \psi \vee \gamma$.

- If φ is $\psi \wedge \gamma$. If $\mathbf{B}, v \models \varphi$ then $\mathbf{B}, v \models \psi$ and $\mathbf{B}, v \models \gamma$. By induction hypothesis, we get that there is an $\mathbf{A}_x \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}_x, v_{A_x} \models \hat{\psi}$, and there is an $\mathbf{A}_y \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}_y, v_{A_y} \models \hat{\gamma}$. By Lemma 6.7, we then have that, for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i, v_{A_i} \models \hat{\psi}$ and $\mathbf{A}_i, v_{A_i} \models \hat{\gamma}$. Thus, there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}, v_A \models \hat{\psi} \wedge \hat{\gamma}$, or equivalently such that $\mathbf{A}, v_A \models \hat{\varphi}$.

  On the other hand, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then $\mathbf{A}, v_A \models \hat{\psi}$ and $\mathbf{A}, v_A \models \hat{\gamma}$. By induction hypothesis, it follows that $\mathbf{B}, v \models \psi$ and $\mathbf{B}, v \models \gamma$, and therefore that $\mathbf{B}, v \models \psi \wedge \gamma$.

- Let φ be $\exists x(\psi)$. If $\mathbf{B}, v \models \varphi$, then there is a valuation v', which is $\{x\}$-equivalent to v, such that $\mathbf{B}, v' \models \psi$. By induction hypothesis, it follows that there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v'_A \models \hat{\psi}$. Given that $\mathbf{A}, v'_A \models \hat{\psi}$ and furthermore $v'_A(x) = min(\mathbf{A})$ or $v'_A(x) = max(\mathbf{A})$, $\mathbf{A}, v'_A \models \exists w_0 w_1(\psi_{min}(w_0) \wedge$

$\psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \psi'(w_0, w_1) \wedge (w_0 = x \vee w_1 = x))$. Since v' is $\{x\}$-equivalent to v, it clearly follows that v'_A is $\{x\}$-equivalent to v_A and thus that $\mathbf{A}, v_A \models \hat{\varphi}$.

Conversely, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then there is a valuation u, which is $\{x\}$-equivalent to v_A, such that $\mathbf{A}, u \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge (x = w_0 \vee x = w_1) \wedge \psi'(w_0, w_1))$. Since u is $\{x\}$-equivalent to v_A and $u(x) = v_A(w_0)$ or $u(x) = v_A(w_1)$, there is a valuation v' on $\mathbf{B}$, which is $\{x\}$-equivalent to v, such that the corresponding valuation v'_A on $\mathbf{A}$ is precisely u. Therefore $\mathbf{A}, v'_A \models \hat{\psi}$, and by induction hypothesis $\mathbf{B}, v' \models \psi$. It follows that $\mathbf{B}, v \models \exists x(\psi)$.

- Let φ be $\forall x(\psi)$. If $\mathbf{B}, v \models \varphi$, then for every valuation v', which is $\{x\}$-equivalent to v, we have that $\mathbf{B}, v' \models \psi$. Let $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, and let u be a valuation on $\mathbf{A}$, which is $\{x\}$-equivalent to v_A, and such that $u(x) = min(\mathbf{A})$ or $u(x) = max(\mathbf{A})$, it follows that there is a valuation u' on $\mathbf{B}$, which is $\{x\}$-equivalent to v, such that the corresponding valuation u'_A is exactly u. Since u' is $\{x\}$-equivalent to v, then by induction hypothesis there is an $\mathbf{A}_x \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}_x, u'_{A_x} \models \hat{\psi}$, and thus by Lemma 6.7, $\mathbf{A}, u'_A \models \hat{\psi}$. It follows that, for every valuation u on $\mathbf{A}$, which is $\{x\}$-equivalent to v_A, $\mathbf{A}, u \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge ((x = w_0 \vee x = w_1) \rightarrow \psi'(w_0, w_1))$. Therefore $\mathbf{A}, v_A \models \hat{\varphi}$.

  On the other hand, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then for every valuation u, which is $\{x\}$-equivalent to v_A, $\mathbf{A}, u \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge ((x = w_0 \vee x = w_1) \rightarrow \psi'(w_0, w_1))$. Furthermore, if v' is a valuation on $\mathbf{B}$, which is $\{x\}$-equivalent to v, then the corresponding valuation v'_A on $\mathbf{A}$, is $\{x\}$-equivalent to v_A and $v'_A(x) = v'_A(w_0) = min(\mathbf{A})$ or $v'_A(x) = v'_A(w_1) = max(\mathbf{A})$. Therefore, if v' is a valuation on $\mathbf{B}$, which is $\{x\}$-equivalent to v, then $\mathbf{A}, v'_A \models \hat{\psi}$ and by induction hypothesis, $\mathbf{B}, v' \models \psi$. It follows that, $B, v \models \forall x(\psi)$.

- Let φ be $\exists X(\psi)$. If $\mathbf{B}, v \models \varphi$, then there is a valuation v', which is $\{X\}$-equivalent to v, such that $\mathbf{B}, v' \models \psi$. By induction hypothesis, it follows that there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v'_A \models \hat{\psi}$.
  Given that $\mathbf{A}, v'_A \models \hat{\psi}$ and by definition $v'_A(X) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$, it clearly follows that $\mathbf{A}, v'_A \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \psi'(w_0, w_1) \wedge \forall x_1 \ldots x_r (X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1))$.

Since v' is $\{X\}$-equivalent to v, we know that v'_A is $\{X\}$-equivalent to v_A and thus that $\mathbf{A}, v_A \models \hat{\varphi}$.

Conversely, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then there is a valuation u, which is $\{X\}$-equivalent to v_A, such that $\mathbf{A}, u \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \psi'(w_0, w_1) \wedge \forall x_1 \ldots x_r (X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1))$. Since u is $\{X\}$-equivalent to v_A and $u(X) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$, there is a valuation v' on $\mathbf{B}$, which is $\{X\}$-equivalent to v, such that the corresponding valuation v'_A on $\mathbf{A}$ is precisely u. Therefore $\mathbf{A}, v'_A \models \hat{\psi}$, and by induction hypothesis $\mathbf{B}, v' \models \psi$. It follows that $\mathbf{B}, v \models \exists X(\psi)$.

- Let φ be $\forall X(\psi)$. If $\mathbf{B}, v \models \varphi$, then for every valuation v', which is $\{X\}$-equivalent to v, we have that $\mathbf{B}, v' \models \psi$. Let $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, and let u be a valuation on $\mathbf{A}$, which is $\{X\}$-equivalent to v_A, and such that $u(X) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$, it follows that there is a valuation u' on $\mathbf{B}$, which is $\{X\}$-equivalent to v, such that the corresponding valuation u'_A is exactly u. Since u' is $\{X\}$-equivalent to v, then by induction hypothesis there is an $\mathbf{A}_x \in \mathcal{B}_\rho|_{\geq 2}$ such that $\mathbf{A}_x, u'_{A_x} \models \hat{\psi}$, and thus by Lemma 6.7, $\mathbf{A}, u'_A \models \hat{\psi}$. It follows that, for every valuation u on $\mathbf{A}$, which is $\{X\}$-equivalent to v_A, $\mathbf{A}, u \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \forall x_1 \ldots x_r (X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \rightarrow \psi'(w_0, w_1))$. Therefore $\mathbf{A}, v_A \models \hat{\varphi}$.

  On the other hand, if there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A}, v_A \models \hat{\varphi}$, then for every valuation u, which is $\{X\}$-equivalent to v_A, $\mathbf{A}, u \models \exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1) \wedge w_0 \neq w_1 \wedge \forall x_1 \ldots x_r (X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \rightarrow \psi'(w_0, w_1))$. Furthermore, if v' is a valuation on $\mathbf{B}$, which is $\{X\}$-equivalent to v, then the corresponding valuation v'_A on $\mathbf{A}$, is $\{X\}$-equivalent to v_A and $v'_A(X) \subseteq \{min(\mathbf{A}), max(\mathbf{A})\}^r$. Therefore, if v' is a valuation on $\mathbf{B}$, which is $\{X\}$-equivalent to v, then $\mathbf{A}, v'_A \models \hat{\psi}$ and by induction hypothesis, $\mathbf{B}, v' \models \psi$. It follows that, $B, v \models \forall X(\psi)$.

□

Finally, we show that $AUTOSAT(\Sigma^1_m)$ is hard for Σ^2_m by a reduction from Σ^1_m-Th($\mathbf{B}$) to $AUTOSAT(\Sigma^1_m[\rho])$. The strategy is to show that for each Σ^1_m-sentence

φ of the Boolean vocabulary, we can build in polynomial time $\hat{\hat{\varphi}}$, a Σ^1_m-sentence of vocabulary ρ, which is equivalent to $\hat{\varphi}$. Since $\hat{\varphi}$ and $\hat{\hat{\varphi}}$ are equivalent, we get from Lemma 6.7, Lemma 6.8, and the fact that $\mathcal{B}_\rho|_{\leq 2}$ is not empty, the following: Let $\varphi \in \Sigma^1_m$ be a sentence of the Boolean vocabulary,

- $\mathbf{B} \models \varphi$ iff, for every ρ-structure $\mathbf{A} \in \mathcal{B}_\rho|_{\leq 2}$, it holds that $\mathbf{A} \models \hat{\hat{\varphi}}$, and
- $\mathbf{B} \not\models \varphi$ iff, for every ρ-structure $\mathbf{A} \in \mathcal{B}_\rho|_{\leq 2}$, it holds that $\mathbf{A} \not\models \hat{\hat{\varphi}}$.

Given that in particular the ρ-structure $\mathbf{I}_{\hat{\hat{\varphi}}'}$ which encodes $\hat{\hat{\varphi}}$ has more than two elements, $\mathbf{B} \models \varphi$ iff $\mathbf{I}_{\hat{\hat{\varphi}}'} \models \hat{\hat{\varphi}}$.

Proposition 6.9. *For every $m \geq 1$, $AUTOSAT(\Sigma^1_m[\rho])$ is hard for Σ^2_m under polynomial time reductions.*

Proof. (By a reduction from Σ^1_m-Th($\mathbf{B}$) to $AUTOSAT(\Sigma^1_m[\rho])$). Given φ, a Σ^1_m-sentence of the Boolean vocabulary, we produce in polynomial time $\hat{\hat{\varphi}}$, a sentence in $\Sigma^1_m[\rho]$, such that $\varphi \in \Sigma^1_m$-Th($\mathbf{B}$) iff $\mathbf{I}_{\hat{\hat{\varphi}}'} \in AUTOSAT(\Sigma^1_m[\rho])$, or equivalently, $\mathbf{B} \models \varphi$ iff $\mathbf{I}_{\hat{\hat{\varphi}}'} \models \hat{\hat{\varphi}}$.

Let $\hat{\varphi}$ be the sentence obtained from φ as indicated in Definition 6.5. We build $\hat{\hat{\varphi}}$ from $\hat{\varphi}$ and φ. First, we eliminate all second-order quantifiers in $\varphi'(w_0, w_1)$, obtaining $\varphi''(X_{11}, \ldots, X_{1s_1}, X_{21}, \ldots, X_{2s_2}, \ldots, X_{m1}, \ldots, X_{ms_m}, w_0, w_1)$, where for $1 \leq j \leq m$, $s_j \geq 1$, and $X_{11}, \ldots, X_{ms_m}$ is the set of second-order variables which appear free in φ''. Note that, the set of second-order variables which appear free in φ'' (i.e., all variables in φ''), coincides with the set of all second-order variables in $\varphi'(w_0, w_1)$, which in turn coincides with the set of second-order variables in the Σ^1_m-sentence φ.

We then write $\hat{\hat{\varphi}}$ as

$$\exists X_{11} \ldots \exists X_{1s_1} \forall X_{21} \ldots \forall X_{2s_2} \ldots QX_{m1} \ldots QX_{ms_m} \big(\exists w_0 w_1 (\psi_{min}(w_0) \wedge \psi_{max}(w_1)$$

$$\wedge w_0 \neq w_1 \wedge \varphi''(X_{11}, \ldots, X_{1s_1}, X_{21}, \ldots, X_{2s_2}, \ldots, X_{m1}, \ldots, X_{ms_m}, w_0, w_1)\big),$$

where $\exists X_{11} \ldots \exists X_{1s_1} \forall X_{21} \ldots \forall X_{2s_2} \ldots QX_{m1} \ldots QX_{ms_m}$ is the prefix of second-order quantifiers in φ.

Clearly $\hat{\varphi}$ can be built in polynomial time, and given φ followed by $\hat{\varphi}$ as input, $\hat{\hat{\varphi}}$ can be built in linear time.

Also clearly $\hat{\hat{\varphi}}$ is a $\Sigma^1_m[\rho]$-sentence. Furthermore, $\hat{\hat{\varphi}}$ is equivalent to $\hat{\varphi}$. This can be seen by noting that, if α is a formula in $\Sigma^1_m[\rho] \cup \Pi^1_m[\rho]$, then for every ρ-structure

$\mathbf{A}$ and valuation v on $\mathbf{A}$,

$$\mathbf{A}, v \models \alpha'(w_0, w_1) \quad \text{iff} \quad \mathbf{A}, v \models Q_1X_1 \ldots Q_kX_k(\alpha''(X_1, \ldots, X_k, w_0, w_1)), \tag{1}$$

where $\alpha''(X_1, \ldots, X_k, w_0, w_1)$ is the formula obtained by eliminating all second-order quantifiers in $\alpha'(w_0, w_1)$, and $Q_1X_1 \ldots Q_kX_k$ is the prefix of second-order quantifiers of α.

To prove (1), we use induction on α. If α is an atomic formula, or α is of the form $\neg\psi$, $\psi \vee \gamma$, $\psi \wedge \gamma$, $\exists x(\psi)$ or $\forall x(\psi)$, then $Q_1X_1 \ldots Q_1X_k(\alpha''(X_1, \ldots, X_k, w_0, w_1))$ and $\alpha'(w_0, w_1)$ are exactly the same formula, and our proposition is trivially true.

If α is of the form $\exists X(\psi)$, then $\alpha'(w_0, w_1)$ is $\exists X(\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \wedge \psi'(w_0, w_1))$. We want to show that this is equivalent to $\exists X Q_1X_1 \ldots Q_kX_k(\alpha''(X, X_1, \ldots, X_k, w_0, w_1))$, where $\alpha''(X, X_1, \ldots, X_k, w_0, w_1)$ is the formula obtained by eliminating all second-order quantifiers in $\alpha'(w_0, w_1)$, and $\exists X Q_1X_1 \ldots Q_kX_k$ is the prefix of second-order quantifiers of α. If $\mathbf{A}, v \models \alpha'(w_0, w_1)$ then there is a valuation v', which is $\{X\}$-equivalent to v, such that $\mathbf{A}, v' \models \forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \wedge \psi'(w_0, w_1)$. Since $\mathbf{A}, v' \models \psi'(w_0, w_1)$, it follows by induction hypothesis that, $\mathbf{A}, v' \models Q_1X_1 \ldots Q_kX_k(\psi''(X_1, \ldots, X_k, w_0, w_1))$, where $\psi''(X_1, \ldots, X_k, w_0, w_1)$ is the formula obtained by eliminating all second-order quantifiers in $\psi'(w_0, w_1)$, and $Q_1X_1 \ldots Q_kX_k$ is the prefix of second-order quantifiers of ψ. Given that, none of the second order variables $X_1, \ldots, X_k$ appear free in the formula $\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1))$, we have that $\mathbf{A}, v' \models Q_1X_1 \ldots Q_kX_k(\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \wedge \psi''(X_1, \ldots, X_k, w_0, w_1))$. Therefore, $\mathbf{A}, v \models \exists X Q_1X_1 \ldots Q_kX_k(\alpha''(X, X_1, \ldots, X_k, w_0, w_1))$.

On the other hand, if $\mathbf{A}, v \models \exists X Q_1X_1 \ldots Q_kX_k(\alpha''(X, X_1, \ldots, X_k, w_0, w_1))$, then there is a valuation v', which is $\{X\}$-equivalent to v, such that
$\mathbf{A}, v' \models Q_1X_1 \ldots Q_kX_k(\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)) \wedge \psi''(X_1, \ldots, X_k, w_0, w_1))$, where $\psi''(X_1, \ldots, X_k, w_0, w_1)$ is the formula obtained by eliminating all second-order quantifiers in $\psi'(w_0, w_1)$, and $Q_1X_1 \ldots Q_kX_k$ is the prefix of second-order quantifiers of ψ. It follows that $\mathbf{A}, v' \models Q_1X_1 \ldots Q_kX_k(\psi''(X_1, \ldots, X_k, w_0, w_1))$, and by induction hypothesis that $\mathbf{A}, v' \models \psi'(w_0, w_1)$. Since the valuation v' on $\mathbf{A}$ also satisfies, $\forall x_1 \ldots x_r(X(x_1, \ldots, x_r) \rightarrow (x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1))$, we have that $\mathbf{A}, v \models \alpha'(w_0, w_1)$.

If α is a formula of the form $\forall X(\psi)$, the proof is completely analogous to the previous case.

Summarising, for each Σ^1_m-sentence φ of the Boolean vocabulary, we can build in polynomial time $\hat{\hat{\varphi}}$, a Σ^1_m-sentence of vocabulary ρ, which is equivalent to $\hat{\varphi}$. Since $\hat{\varphi}$ and $\hat{\hat{\varphi}}$ are equivalent, we get by Lemma 6.8 that $\mathbf{B} \models \varphi$ iff there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A} \models \hat{\varphi}$. And by Lemma 6.7 and the fact that $\mathcal{B}_\rho|_{\leq 2}$ is not empty, we also get that there is an $\mathbf{A} \in \mathcal{B}_\rho|_{\geq 2}$, such that $\mathbf{A} \models \hat{\hat{\varphi}}$ iff for every $\mathbf{A}_i \in \mathcal{B}_\rho|_{\geq 2}$, $\mathbf{A}_i \models \hat{\hat{\varphi}}$. Given that for every $\varphi \in \Sigma^1_m[\rho]$, we have that $\mathbf{I}_{\hat{\hat{\varphi}}'} \in \mathcal{B}_\rho|_{\geq 2}$, it clearly follows that $\mathbf{B} \models \varphi$ iff $\mathbf{I}_{\hat{\hat{\varphi}}'} \models \hat{\hat{\varphi}}$. $\square$

6.3 $AUTOSAT(\Sigma^1_m)$ is in Σ^2_m

To prove that for every $m \geq 1$, $AUTOSAT(\Sigma^1_m[\rho])$ is in Σ^2_m, we build a *nondeterministic* Turing machine with an oracle in Σ^p_{m-1} which decides in time $\mathcal{O}(2^{(n^c)})$ whether an arbitrary input ρ-structure $\mathbf{I}_{\varphi'}$ is in $AUTOSAT(\Sigma^1_m[\rho])$.

Recall that by Theorem 3.8, Σ^2_m captures $\bigcup_{c\in\mathbb{N}} \mathrm{NTIME}(2^{n^c})^{\Sigma^p_{m-1}}$, where n is the size of the input and Σ^p_{m-1} the m-th level of the polynomial-time hierarchy.

Note that the proof is similar to the proof of Theorem 3.3 in [HT05].

Proposition 6.10. *For every $m \geq 1$, $AUTOSAT(\Sigma^1_m[\rho])$ is in Σ^2_m.*

Proof. We define a Turing machine $\mathcal{M}$ which decides whether an arbitrary input ρ-structure $\mathbf{I}_{\varphi'}$ encodes a Σ^1_m-sentence φ of vocabulary ρ such that $\mathbf{I}_{\varphi'} \models \varphi$, and which works in $\mathrm{NTIME}(2^{n^c})^{\Sigma^p_{m-1}}$.

Note that the number of variables in φ as well as the arity of the quantified second-order variables, is $\mathcal{O}(n)$, where n is the size of (the domain of) the input structure $\mathbf{I}_{\varphi'}$.

Let φ be a Σ^1_m-formula, we can think of φ as a formula where all second-order quantifiers are existential, and are grouped together at the beginning with $m-1$ interleaving negation symbols. This is clearly possible by the well known relation between existential and universal quantifiers.

Let the ρ-structure $\mathbf{I}_{\varphi'}$ be the input to $\mathcal{M}$ and n the size of $\mathbf{I}_{\varphi'}$. The machine works as follows:

1. $\mathcal{M}$ writes in its work tape the formula φ encoded by $I_{\varphi'}$. This requires space $\mathcal{O}(n)$ and can be done working deterministically in polynomial time.

2. $\mathcal{M}$ deterministically checks that φ is a well formed sentence in $\Sigma^1_m[\rho]$. Again this takes polynomial time.

3. Let φ be $\exists X_1 \dots \exists X_{s_1}(\psi)$, where $X_1, \dots, X_{s_1}$ are second-order variables of arity r for some $s_1, r \geq 1$, and ψ is either a first-order formula or the *negation* of a formula which starts with a block of existential second-order quantifiers. $\mathcal{M}$ guesses a sequence of relations as possible values for the second-order variables $X_1, \dots, X_{s_1}$, respectively. The space needed for the guessed relations is $\mathcal{O}(s_1 \cdot r \cdot n^r \cdot \log n)$. Since the length of φ is bounded by n, we know that r and s_1 are also bounded by n. Therefore, the sequence of guessed relations needs space bounded by n^{n+3}, i.e., $\mathcal{M}$ needs space $\mathcal{O}(n^{n^c})$, which is $\mathcal{O}(2^{n^c})$ for some constant c. The nondeterministic time needed for this step is also $\mathcal{O}(2^{n^c})$.

4. Now $\mathcal{M}$ evaluates the formula ψ substituting the guessed relations for the second-order variables $X_1, \dots, X_{s_1}$, respectively. According to what we said above regarding the formula ψ, for its evaluation we have to consider two different cases:

   (a) ψ is a first-order formula:
   Then $\mathcal{M}$ evaluates ψ deterministically, which takes time polynomial in the size of the guessed relations. Thus, this step takes time $(\mathcal{O}(2^{n^c}))^{c_2}$ for some constant c_2, which is still $\mathcal{O}(2^{n^{c'}})$ for some constant c'.
   Note that, to evaluate ψ we do not need to use oracles (as we do in the second case, see below), since as the quantified variables in ψ are first-order variables, we can afford the time needed to evaluate the negation of an existential quantifier *deterministically* as a universal quantifier (note that there are 2^{n^r} relations of arity r). So, the evaluation of the formula ψ is done completely in a deterministic way, and still in time $\mathcal{O}(2^{n^{c'}})$ for some constant c'.

   (b) ψ is the *negation* of a formula which starts with a block of existential second-order quantifiers:
   Say, $\psi \equiv \neg(\exists Y_1 \dots \exists Y_{s_2}(\psi'))$. Then, $\mathcal{M}$ calls an oracle Turing machine $\mathcal{M}_\psi$ which will evaluate the sub-formula $\exists Y_1 \dots \exists Y_{s_2}(\psi')$ over the structure $\mathbf{I}_{\varphi'}$ extended with the second-order relations guessed by $\mathcal{M}$. When $\mathcal{M}_\psi$ ends, the machine $\mathcal{M}$ proceeds by inverting the result of the compu-

tation of $\mathcal{M}_\psi$. That is, $\mathcal{M}$ accepts the input structure $\mathbf{I}_{\varphi'}$ iff $\mathcal{M}_\psi$ rejects its input.

The way in which the oracle machine $\mathcal{M}_\psi$ works is exactly the same as the way in which the original machine $\mathcal{M}$ works. First, $\mathcal{M}_\psi$ guesses a sequence of s_2 relations of arity r, as possible values for the second-order variables $Y_1, \ldots, Y_{s_2}$, respectively. Then, $\mathcal{M}_\psi$ evaluates the formula ψ' substituting the guessed relations for the second-order variables $Y_1, \ldots, Y_{s_2}$, respectively. If in the quantifier prefix of the sub-formula ψ' there is a negation symbol before an existential second-order quantifier, then $\mathcal{M}_\psi$ calls in turn another oracle Turing machine which will work like $\mathcal{M}$ and $\mathcal{M}_\psi$.

This process is followed until there are no more negations before second-order existential quantifiers in the remaining sub-formulae.

Note that the oracle machine $\mathcal{M}_\psi$ needs the same space and time as the original machine $\mathcal{M}$, i.e., $\mathcal{O}(2^{n^c})$, where n is the size of $\mathbf{I}_{\varphi'}$, which is the input structure to the machine $\mathcal{M}$. However, it is not necessary for $\mathcal{M}_\psi$ to work in NTIME(2^{n^c}). We define $\mathcal{M}_\psi$ in such a way that it works in NTIME(n^c) (i.e., in NP).

To evaluate ψ, $\mathcal{M}_\psi$ needs to know the input structure $\mathbf{I}_{\varphi'}$, as well as the values for the second-order variables $X_1, \ldots, X_{s_1}$ which were guessed by the machine $\mathcal{M}$. Therefore, the input to the oracle machine $\mathcal{M}_\psi$ is as follows: a) the formula ψ, which requires space $\mathcal{O}(n)$, b) the input structure $\mathbf{I}_{\varphi'}$, which also requires space $\mathcal{O}(n)$, and c) the guessed values for the variables $X_1, \ldots, X_{s_1}$, which requires space bounded by $\mathcal{O}(2^{n^c})$ as seen above.

So, the necessary space for the query to the oracle in the oracle tape is $\mathcal{O}(2^{n^c})$. In fact, we use padding to make sure that the input to the oracle is of size 2^{n^c}. That is, the query to the oracle consists of the formula ψ, the input structure $\mathbf{I}_{\varphi'}$ and the guessed values for $X_1, \ldots, X_{s_1}$ padded with enough "quasiblanks" to make the total length of the input string to the oracle to be $\mathcal{O}(2^{n^c})$.

Hence, if the oracle machine $\mathcal{M}_\psi$ works in polynomial time, it will work actually in NTIME($(2^{n^c})^{c'}$), where n is the size of $\mathbf{I}_{\varphi'}$. Then an oracle in NP is enough.

As to the other oracle machines, which could eventually be called in turn by $\mathcal{M}_\psi$ or any other oracle machine in the chain, the input in the oracle tape should be built in the same way, except for the input structure, which should be extended with the guessed values for all the second-order variables which are quantified in the prefix of the original formula φ, before the sub-formula which is to be evaluated by the given oracle. The space required in the oracle tape, though, is still $\mathcal{O}(2^{n^c})$.

So, we have got a non deterministic Turing machine $\mathcal{M}$ which decides $AUTO$-$SAT(\Sigma^1_m[\rho])$ in time $\mathcal{O}(2^{n^c})$, for some constant c, and which calls a chain of oracle machines, each belonging to the class NP. Clearly, the depth of nesting of the chain of successive calls to oracles is given by the number of negations which appear in the prefix of the formula φ, minus 1.

Therefore, $\mathcal{M}$ is in the class $\bigcup_{c\in\mathbb{N}} \mathrm{NTIME}(2^{n^c})^{\Sigma^p_{m-1}}$.

$\square$

Chapter 7

Relational Complexity and Second-Order Logic

In [Daw98], Dawar introduced a restricted version of second-order logic SO^ω in which the second-order quantifiers range over relations that are closed under the equivalence relation $\equiv^k$ of k variable equivalence, for some k (see Section 2.2.2). Among other results, he proved that the existential fragment of SO^ω is equivalent to the nondeterministic inflationary fixed-point logic NFP. Since by a result of Abiteboul, Vardi and Vianu [AVV97], NFP captures *relational* NP (NP_r, see Section 2.5), it then follows that the existential fragment of SO^ω captures NP_r.

Aiming to further explore the connection between relational complexity and higher-order logics, we give a direct proof, in the style of the proof of Fagin's theorem, of the fact that the existential fragment of SO^ω captures NP_r. Then we define formally the concept of relational machine with *relational oracle* and show the exact correspondence between the prenex fragments of SO^ω and the levels of the *relational* polynomial-time hierarchy.

This last result is the *relational* equivalent of Stockmeyer characterization of the polynomial-time hierarchy. It was already pointed out by Dawar in [Daw98] where he observed that, if we close the logic NFP simultaneously under negation and the operation of taking nondeterministic fixed-points, we obtain a logic equivalent to SO^ω, though he did not prove it. Moreover, the alternations of negations and fixed-points correspond exactly to the second-order quantifier alternations in the prenex fragments of SO^ω. Since by a result in [AVV97], the closure of NFP under negation and the operation of taking nondeterministic fixed-points captures the relational polynomial-time hierarchy, and the correspondence between the levels of the relational polynomial-time hierarchy and the alternation of negations and fixed points also holds, then the correspondence between the prenex fragments of SO^ω and the levels of the *relational* polynomial hierarchy follows.

However, up to our knowledge, this is the first attempt to formally define the *relational* polynomial-time hierarchy in terms of relational machines with *relational oracles*, i.e., oracles consisting on classes of structures closed under isomorphisms instead of sets of strings. This allows us to establish a direct connection between the *relational* polynomial hierarchy and SO^ω, without using the Abiteboul and Vianu normal form for relational machines (see [AV95]). That is, we do not need to encode a canonical representation of the input structure in the Turing machine tape of the relational machine as in [AVV97].

Note that along this chapter we refer frequently to the material in Sections 2.2.2 and 2.5 as we make extensive use of the model theoretic concept of *type* and the notion of *relational* complexity.

7.1 A Restricted Second-Order Logic

Motivated by the study of the finite model theory of the infinitary logic with finitely many variables ($\mathcal{L}^\omega_{\infty\omega}$), Dawar defined in [Daw98] a restricted version of second-order logic SO^ω which is contained within $\mathcal{L}^\omega_{\infty\omega}$. This is obtained by restricting the interpretation of the second-order quantifiers to relations closed under the equivalence relation $\equiv^k$, for some k (see Definitions 2.3 and 2.4).

We define next the syntax and semantics of SO^ω.

Definition 7.1. In addition to the symbols of first-order logic, the alphabet of SO^ω contains, for each $k \geq 1$, a *second-order quantifier* $\exists^k$ and countably many k-ary *relation variables* $V^k_1, V^k_2, \ldots$ To denote relation variables we use letters $X, Y, \ldots$.

Let $m \geq 1$ and let σ be a relational vocabulary, we denote by $\Sigma^{1,\omega}_m[\sigma]$ the class of formulae of the form

$$\exists^{k^1_1} X_{11} \ldots \exists^{k^1_{s_1}} X_{1s_1} \forall^{k^2_1} X_{21} \ldots \forall^{k^2_{s_2}} X_{2s_2} \ldots Q^{k^m_1} X_{m1} \ldots Q^{k^m_{s_m}} X_{ms_m}(\varphi),$$

where for $i, j \geq 1$ we have $s_i, k^i_j \geq 1$ and $arity(X_{ij}) \leq k^i_j$, Q is either $\exists$ or $\forall$, depending on whether m is odd or even, respectively, and φ is a first-order formula of vocabulary $\sigma \cup \{X_{11}, \ldots, X_{1s_1}, X_{21}, \ldots, X_{2s_2}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$. As usual, $\forall^k X(\varphi)$ abbreviates $\neg\exists^k X(\neg\varphi)$.

Similarly, we denote by $\Pi^{1,\omega}_m[\sigma]$ the class of formulae of the form

$$\forall^{k^1_1} X_{11} \ldots \forall^{k^1_{s_1}} X_{1s_1} \exists^{k^2_1} X_{21} \ldots \exists^{k^2_{s_2}} X_{2s_2} \ldots Q^{k^m_1} X_{m1} \ldots Q^{k^m_{s_m}} X_{ms_m}(\varphi),$$

where for $i, j \geq 1$ we have $s_i, k^i_j \geq 1$ and $arity(X_{ij}) \leq k^i_j$, Q is either $\forall$ or $\exists$, depending on whether m is odd or even, respectively, and φ is a first-order formula of vocabulary $\sigma \cup \{X_{11}, \ldots, X_{1s_1}, X_{21}, \ldots, X_{2s_2}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$.

The set of formulae of SO^ω is then defined as the union of the set of first-order formulae with $\bigcup_{m\geq 1} \Sigma^{1,\omega}_m$.

The notion of *satisfaction* in SO^ω extends the notion of satisfaction in first-order with the following rule:

- $\mathbf{I} \models \exists^k X(\varphi)$ where $k \geq 1$, X is a relation variable of arity $r \leq k$, φ is a wff of vocabulary $\sigma \cup \{X\}$ and $\mathbf{I}$ is a σ-structure, iff there is an $R \subseteq I^r$ such that R is closed under the equivalence relation $\equiv^k$ in $\mathbf{I}$, and $(\mathbf{I}, R) \models \varphi$. Here $(\mathbf{I}, R)$ is the $(\sigma \cup \{X\})$-structure expanding $\mathbf{I}$, in which X is interpreted as R.

Note that, for each $\Sigma^{1,\omega}_m$-formula $\varphi \equiv \exists^{k_1} X_1 \ldots Q^{k_s} X_s(\psi)$, there is a formula $\hat{\varphi} \in \Sigma^1_m$ which is equivalent to φ. The corresponding formula $\hat{\varphi}$ is simply,

$$\exists X_1 \ldots Q X_s \Big(\psi \wedge \bigwedge_{1\leq i\leq s} \gamma^{k_i}(X_i)\Big),$$

where $\gamma^{k_i}(X_i)$ expresses that X_i is $\equiv^{k_i}$-closed. Since $\text{IFP} \subseteq \Sigma^1_1 \cap \Pi^1_1$, it clearly follows from Theorem 2.6 that γ^k is definable in Σ^1_m.

It is is important to mention that the restricted second-order logic SO^ω is not really restricted over ordered structures.

Theorem 7.2 ([Daw98]). *On ordered structures, for every $m \geq 1$, $\Sigma^{1,\omega}_m = \Sigma^1_m$ and $\Pi^{1,\omega}_m = \Pi^1_m$.*

7.2 Relational Machines

We already introduced the relational machine in Section 2.5, Chapter 2. But, for the detail of the proofs in this chapter, we need a formal definition.

Recall that the relational machine consists of a Turing machine augmented with a finite set of fixed-arity relations forming a *relational store* (*rs*). We assume that the Turing machine component of our relational machine consists of a *finite control* which has a finite set of internal states, plus a one-way infinite *tape* equipped with a read/write *tape head* which can move right or left.

Definition 7.3. We formally define a *deterministic relational machine* as a eleven-tuple, $\langle Q, \Sigma, \delta, q_0, \mathfrak{b}, F, \tau, \sigma, T, \Omega, \Phi \rangle$, where:

1. Q is the finite set of internal states;
2. Σ is the tape alphabet;
3. $\mathfrak{b} \in \Sigma$ is the symbol denoting blank;
4. $q_0 \in Q$ is the initial state;
5. $F \subseteq Q$ is the set of accepting final states;
6. τ is the vocabulary of the *rs*;
7. $\sigma \subset \tau$ is the vocabulary of the input structure;
8. $T \in \tau \setminus \sigma$ is the output relation;
9. Ω is a finite set of first-order sentences of vocabulary τ;
10. Φ is a finite set of first-order formulas of vocabulary τ, and
11. $\delta : Q \times \Sigma \times \Omega \to \Sigma \times Q \times \{R, L\} \times \Phi \times \tau$ is a partial function called the transition function.

Transitions are based on:

i. the current state;

ii. the content of the current tape cell; and

iii. the answer to a Boolean first-order query evaluated on the τ-structure held in the *rs*.

Situations in which the transition function is undefined indicate that the computation must stop. Otherwise, the result of the transition function is interpreted as follows:

i. the first component is the symbol to be written on the scanned cell of the tape;

ii. the second component is the new state;

iii. the third component specifies the moves of the tape head: R means moving one cell to the right and L means moving one cell to the left;

iv. the fourth component specifies an n-ary ($n \geq 1$) first-order query φ to be evaluated on the τ-structure held in the *rs*; and

v. the fifth component is an n-ary relation symbol in τ, which specifies the n-ary relation in the *rs* to be replaced by the relation obtained from the evaluation of φ.

We can now introduce for relational machines the analogous to the concepts of configuration (also called instantaneous description or snapshot) and computation of Turing machines.

Definition 7.4. Given a relational machine M, a *configuration* of M is a description of the whole status of the computation: it includes the contents of the tape, the position of the tape head, the current internal state of the finite control, and the contents of the relational store. Formally, a configuration of M is a 3-tuple $(q, w, \mathbf{A})$ where q is the current internal state of M, $w \in \Sigma^* \# \Sigma^*$ represents the current contents of the tape, and $\mathbf{A}$ is the current τ-structure held in the *rs*. The symbol "#" is supposed not to be in Σ, and marks the position of the tape head (by convention, the head scans the symbol immediately at the right of the "#"). All symbols in the infinite tape not appearing in w are assumed to be the particular symbol blank "$\mathfrak{b}$".

The machine starts in the initial state, with the input in the designated relations of the relational store, and an empty tape.

Definition 7.5. The *initial configuration* of a relational machine M on an input structure $\mathbf{I}$ of vocabulary σ is $(q_0, \#, \mathbf{A})$, where $\mathbf{A}$ is the τ-structure which extends $\mathbf{I}$ with an empty relation $R_i^{\mathbf{A}}$ for each relation symbol R_i in τ. The head is assumed to be in the left-most position of the tape.

The fact of accepting an input is indicated by an accepting configuration.

Definition 7.6. An *accepting configuration* is a configuration whose state is an accepting state.

A computation of a relational machine can now be defined as a sequence of configurations:

Definition 7.7. Given a relational machine M and an input structure $\mathbf{I}$, a *partial computation* of M on $\mathbf{I}$ is a (finite or infinite) sequence of configurations of M, in which each step from a configuration to the next obeys the transition function. A *computation* is a partial computation which start with the initial configuration, and ends in a configuration in which no more steps can be performed. An *accepting computation* is a computation ending in an accepting configuration, and in this case the input structure $\mathbf{I}$ is *accepted*.

Relational machines give rise to three types of computational devices. First, we can think of a relational machine M as an *acceptor* of a *relational language*, i.e., a class of structures closed under isomorphisms. In this case the *relational language accepted* by M, denoted $\mathcal{L}(M)$, is the set of input structures accepted by M. We can also think of a relational machine M as computing a *relational function* from input structures to output relations. The relational function computed by M is defined on the relational language accepted by M, and for each accepted input structure the value of the relational function is the relation held in the output relation T in the *rs* when the machine stops in an accepting state. Finally, we can think of a relational machine M as computing a *mixed function*, i.e., a function from structures to strings where the output is written on the machine's tape. Again the function is defined on the relational language accepted by M. For each accepted input structure the value of this function is the word which appears in the tape when the machine stops in an accepting state.

The arity of a relational machine is the maximum number of variables which appear in any formula in its finite control.

Definition 7.8. Let $M = \langle Q, \Sigma, \delta, q_0, \mathfrak{b}, F, \tau, \sigma, T, \Omega, \Phi \rangle$ be a relational machine, the *arity* of M, denoted as $arity(M)$, is $max(\{|var(\varphi)| : \varphi \in \Omega \cup \Phi\})$.

7.2.1 Nondeterministic Relational Machines

In analogy with nondeterministic Turing machines, we can define nondeterministic relational machines.

Definition 7.9. A *nondeterministic relational machine* is a eleven-tuple, $\langle Q, \Sigma, \delta, q_0, \mathfrak{b}, F, \sigma, \tau, T, \Omega, \Phi\rangle$, where each component is as in the deterministic case, with the exception that the transition function is defined by

$$\delta : Q \times \Sigma \times \Omega \to \mathcal{P}(\Sigma \times Q \times \{R, L\} \times \Phi \times \tau)$$

where, for any set A, $\mathcal{P}(A)$ denotes the powerset of A.

All the definitions and remarks made for the deterministic case about configurations and computations, apply in the same manner to the nondeterministic model. However, on a given input structure there is now not only one computation, but a set of possible computations. Acceptance for nondeterministic relational machines is therefore defined as follows.

Definition 7.10. An input structure $\mathbf{I}$ is *accepted* by an nondeterministic relational machine M iff there exists a computation of M on $\mathbf{I}$ ending in an accepting configuration. We denote by $\mathcal{L}(M)$ the *relational language accepted* by M, i.e., the class of σ-structures accepted by M.

7.2.2 Relational Oracle Machines

Definition 7.11. A *relational oracle machine* is a relational machine with a distinguished set of relations in its rs, called oracle relations, and three distinguished states $q_?$, the query state, and q_{YES}, q_{NO}, the answer states.

Similarly to the case of oracle Turing machines, the computation of an oracle relational machine requires that an oracle language be fixed previously to the computation. But, since we are working with relational machines, it is natural to think of a *relational oracle language*, i.e., a class of structures closed under isomorphisms, rather than a set of strings. Let $\mathcal{C}$ be an arbitrary class of structures of some vocabulary σ^o which is closed under isomorphisms. The computation of a relational oracle machine M with oracle $\mathcal{C}$ and distinguished set of oracle relation symbols σ^o, proceeds like in an ordinary relational machine, except for transitions from the query state. From the query state M transfers into the state q_{YES} if the relational structure of vocabulary σ^o formed by the domain of the input structure and the distinguished set of oracle relations currently held in the rs, belongs to $\mathcal{C}$; otherwise, M transfers into the state q_{NO}.

7.2.3 Relational Polynomial-Time Hierarchy

The time complexity of oracle relational machines is defined precisely in the same way as with ordinary relational machines. Each query step counts as one ordinary step. Thus if $\mathcal{C}$ is any deterministic or nondeterministic relational time complexity class and $\mathcal{A}$ is a relational language, we can define $\mathcal{C}^{\mathcal{A}}$ to be the class of all relational languages accepted by halting relational machines of the same sort and time bound as in $\mathcal{C}$, only that the machines have now an oracle $\mathcal{A}$.

Definition 7.12. The levels of the *relational polynomial-time hierarchy* are defined as follows:

- $\Delta_0^{\mathrm{P}_r} = \Sigma_0^{\mathrm{P}_r} = \Pi_0^{\mathrm{P}_r} = \mathrm{P}_r$
- and for $m > 0$, $\quad \Delta_{m+1}^{\mathrm{P}_r} = \mathrm{P}_r^{\Sigma_m^{\mathrm{P}_r}} \quad \Sigma_{m+1}^{\mathrm{P}_r} = \mathrm{NP}_r^{\Sigma_m^{\mathrm{P}_r}} \quad \Pi_{m+1}^{\mathrm{P}_r} = \mathrm{coNP}_r^{\Sigma_m^{\mathrm{P}_r}}$.

The relational complexity class PH_r is the union of all relational complexity classes in the relational polynomial time hierarchy, i.e., $\mathrm{PH}_r = \bigcup_{m \in \mathbb{N}} \Sigma_m^{\mathrm{P}_r}$.

7.3 The Relational Complexity of SO^ω

We know by the work of Dawar that the expressive power of the fragment $\Sigma_1^{1,\omega}$ of SO^ω equals the expressive power of the nondeterministic inflationary fixed point logic.

Theorem 7.13 ([Daw98])**.** $\Sigma_1^{1,\omega} = \mathrm{NFP}$.

It clearly follows from this result and Theorem 2.15 that the classes of relational structures which are finitely axiomatizable in $\Sigma_1^{1,\omega}$, are exactly those classes which belong to the relational complexity class NP_r.

Theorem 7.14 ([Daw98])**.** $\Sigma_1^{1,\omega}$ *captures* NP_r.

We give next a direct proof of this result. That is, we show that for every relational vocabulary σ, every $\Sigma_1^{1,\omega}[\sigma]$-sentence φ can be evaluated in NP_r, and vice versa that every NP_r property of finite relational structures can be expressed in $\Sigma_1^{1,\omega}$. But first, we need some preparation.

The next lemma is a direct consequence of Theorems 2.15 and 2.7.

Lemma 7.15. *For every relational vocabulary σ and every $k \geq 1$, there is a deterministic relational machine $M_{\leq^k}$ of arity $k' \geq 2k$, such that on any input structure $\boldsymbol{I}$ of vocabulary σ, $M_{\leq^k}$ computes the preorder $\preceq^k$ of Theorem 2.7 working in time bounded by a polynomial in $size_{k'}(\boldsymbol{I})$.*

The following two facts are variations of Facts 3.2 and 3.1 in [Tur01b], respectively. Anyway, we prove them for their better understanding.

Fact 7.16. *Let $\mathbf{I}$ be a relational structure of some vocabulary σ, and let $\bar{a} = (a_1, \ldots, a_k)$ and $\bar{b} = (b_1, \ldots, b_k)$ be two k-tuples on $\mathbf{I}$. Let $1 \leq r \leq k$ and let $(i_1, \ldots i_r)$ be a tuple of projection coordinates, where for $1 \leq j < r$, we have $1 \leq i_j < i_{j+1} \leq k$. Let $\bar{a}' = (a_{i_1}, \ldots, a_{i_r})$ and $\bar{b}' = (b_{i_1}, \ldots, b_{i_r})$. It follows that, if $\bar{a} \equiv^k \bar{b}$, then both $\bar{a}' \equiv^k \bar{b}'$ and $\bar{a}' \equiv^r \bar{b}'$ hold.*

Proof. By definition of the FO^k-type of a tuple, for every FO^k formula $\varphi \in tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a})$, with $free(\varphi) \subseteq \{x_{i_1}, \ldots, x_{i_r}\}$, it holds that $\mathbf{I} \models \varphi[\bar{a}]$. Since the valuation represented by the sub-tuple $\bar{a}' = (a_{i_1}, \ldots, a_{i_r})$ assigns to the free variables in φ the same elements from I as the valuation represented by the tuple $\bar{a}$, it also holds that $\mathbf{I} \models \varphi[\bar{a}']$. Thus we can define $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a}')$ as the set of FO^k formulae $\varphi \in tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a})$ such that $free(\varphi) \subseteq \{x_{i_1}, \ldots, x_{i_r}\}$. Given that $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{b}')$ can be defined in the same way, if $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a}) = tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{b})$, then $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a}') = tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{b}')$.

To prove that $\bar{a}' \equiv^r \bar{b}'$, we use the same argument, except that we consider FO^r formulae $\varphi \in tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a})$ and FO^r formulae $\varphi \in tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{b})$, instead of FO^k formulae, to build $tp_{\mathbf{I}}^{\mathrm{FO}^r}(\bar{a}')$ and $tp_{\mathbf{I}}^{\mathrm{FO}^r}(\bar{b}')$, respectively. □

Fact 7.17. *Let $\mathbf{I}$ be a relational structure of some vocabulary σ and let φ be a $\mathrm{FO}^k[\sigma]$-formula with $1 \leq r \leq k$ free variables. The relation that φ defines on $\mathbf{I}$, i.e. $\varphi^{\mathbf{I}}$, is closed under $\equiv^k$.*

Proof. Let $\bar{a}$ and $\bar{b}$ be two r-tuples on $\mathbf{I}$. We show that, if $\bar{a} \in \varphi^{\mathbf{I}}$ and $\bar{a} \equiv^k \bar{b}$, then $\bar{b} \in \varphi^{\mathbf{I}}$. Since $\bar{a} \in \varphi^I$, then by definition of the FO^k-type of a tuple and definition of $\varphi^{\mathbf{I}}$, we have that φ is in $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a})$. Given that $\bar{a} \equiv^k \bar{b}$, φ is also in $tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{b})$, and therefore $I \models \varphi[\bar{b}]$. Hence, $\bar{b} \in \varphi^{\mathbf{I}}$. □

The following is a straightforward consequence of Fact 7.16.

Fact 7.18. *Let $\mathbf{I}$ be a relational structure of some vocabulary σ. For every $1 \leq r \leq k$, it holds that $size_r(\mathbf{I}) \leq size_k(\mathbf{I})$.*

We can now proceed with the first part of our direct proof of Dawar's result regarding the relational complexity of $\Sigma_1^{1,\omega}$, i.e., Theorem 7.14.

Proposition 7.19. *Every class of relational structures definable in* $\Sigma_1^{1,\omega}$ *is in* NP_r.

Proof. We show that for every relational vocabulary σ, every $\Sigma_1^{1,\omega}[\sigma]$-sentence φ can be evaluated in NP_r. Suppose that φ is $\exists^{k_1} X_1 \ldots \exists^{k_s} X_s(\psi)$, where ψ is a first-order formula of vocabulary $\sigma \cup \{X_1, \ldots, X_s\}$. We build a nondeterministic relational machine M_φ which evaluates φ on input structures of vocabulary σ. For $1 \leq i \leq s$, let $k'_i \geq 2k_i$ be the arity of the relational machine $M_{\leq^{k_i}}$ of Lemma 7.15 which computes the preorder $\leq^{k_i}$ of Theorem 2.7. The arity k of M_φ is $max(\{k'_1, \ldots, k'_s\})$. The vocabulary τ of the relational store is $\sigma \cup \{X_1, \ldots, X_s, \leq^{k_1}, \ldots, \leq^{k_s}, S_1, \ldots, S_s\}$, where for $1 \leq i \leq s$, the arity of X_i is exactly the same as the arity $r_i \leq k_i$ of the corresponding quantified relation variable in φ, the arity of $\leq^{k_i}$ is $2k_i$, and the arity of S_i is k_i. Let $\mathbf{I}$ be the input structure to M_φ. The machine works as follows:

- For each $1 \leq i \leq s$, M_φ builds the preorder $\leq^{k_i}$ of Theorem 2.7 in its rs. To complete this step, M_φ simply emulates, for each preorder $\leq^{k_i}$, the corresponding deterministic relational machine $M_{\leq^{k_i}}$ of arity $k'_i \geq 2k_i$ of Lemma 7.15. Therefore, M_φ can compute each preorder $\leq^{k_i}$ working deterministically in time bounded by a polynomial in $size_{k'_i}(\mathbf{I})$, and as $k \geq k'_i$, all preorders $\leq^{k_i}$ working deterministically in time bounded by a polynomial in $size_k(\mathbf{I})$ (which by Fact 7.18 is $\geq size_{k'_i}(\mathbf{I})$).

- By stepping through the equivalence classes of the relation $\equiv^{k_i}$ in the order given by $\leq^{k_i}$, M_φ computes $size_{k_i}(\mathbf{I})$ for every $1 \leq i \leq s$. Clearly the computation can be carried out by working deterministically in a number of steps polynomial in $size_k(\mathbf{I})$. See also Proposition 2.13.

- For $1 \leq i \leq s$, M_φ guesses and writes over its Turing machine tape a tuple $\bar{a}_i = (a_{i1}, \ldots, a_{isize_{k_i}(\mathbf{I})}) \in \{0,1\}^{size_{k_i}(\mathbf{I})}$. Since for each $1 \leq i \leq s$, M_φ can perform this task working nondeterministically in time $size_{k_i}(\mathbf{I})$ (which by Fact 7.18 is $\leq size_k(\mathbf{I})$), this computation takes time polynomial in $size_k(\mathbf{I})$.

- Using the binary tuples guessed in the previous step, M_φ generates, for every $1 \leq i \leq s$, a relation which is placed in X_i in its rs and is closed under the equivalence relation $\equiv^{k_i}$ in $\mathbf{I}$.

```
For $i = 1$ to $s$.
    For $j = 1$ to $size_{k_i}(\mathbf{I})$
        Begin
        If $a_{ij} = 1$ then
            $X_i := X_i(x_1, \ldots, x_{r_i}) \vee (\exists x_{r_i+1} \ldots x_{k_i}(\neg S_i(x_1, \ldots, x_{k_i}) \wedge$
                $\forall y_1 \ldots y_{k_i}(\leq^{k_i} (y_1, \ldots, y_{k_i}, x_1, \ldots, x_{k_i}) \wedge$
                $\neg \leq^{k_i} (x_1, \ldots, x_{k_i}, y_1, \ldots, y_{k_i}) \rightarrow S_i(y_1, \ldots y_{k_i}))))$;
        $S_i := S_i(\bar{x}) \vee (\neg S_i(\bar{x}) \wedge \forall \bar{y}(\leq^{k_i} (\bar{y}, \bar{x}) \wedge \neg \leq^{k_i} (\bar{x}, \bar{y}) \rightarrow S_i(\bar{y})))$;
        End;
```

M_φ can clearly perform this task working deterministically in time bounded by a polynomial in $size_k(\mathbf{I})$.

- Finally, M_φ evaluates ψ on the τ-structure currently held in its rs. M_φ accepts the input structure $\mathbf{I}$ iff ψ evaluates to true.

Hence, φ can be evaluated in NP$_r$. □

Before proving the converse of the previous result, we need two additional facts.

Fact 7.20. *Let* $\mathbf{I}$ *be a relational structure. For* $1 \leq i \leq n$, *let* $k_i \geq r_i \geq 1$ *and let* $\mathcal{C}_i$ *be the set of equivalence classes of* r_i*-tuples determined by* $\equiv^{k_i}$ *on* $\mathbf{I}$. *If follows that, for every* $\mathcal{C}_R \subseteq \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$, *the relation*

$$R^{\mathbf{I}} = \{(a_{11}, \ldots, a_{1r_1}, a_{21}, \ldots, a_{2r_2}, \ldots, a_{n1}, \ldots a_{nr_n}) \in I^{r_1+r_2+\ldots+r_n} :$$

$$([(a_{11}, \ldots, a_{1r_1})], [(a_{21}, \ldots, a_{2r_2})], \ldots, [(a_{n1}, \ldots a_{nr_n})]) \in \mathcal{C}_R\}$$

is closed under $\equiv^{k_1+k_2+\cdots+k_n}$ *on* $\mathbf{I}$.

Proof. Suppose that there is a $\mathcal{C}_R \subseteq \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$ such that its corresponding relation $R^{\mathbf{I}}$ is not closed under $\equiv^{k_1+k_2+\cdots+k_n}$ on $\mathbf{I}$. Then, there are two $(r_1+r_2+\cdots+r_n)$-tuples

$$\bar{a} = (a_{11}, \ldots, a_{1r_1}, a_{21}, \ldots, a_{2r_2}, \ldots, a_{n1}, \ldots, a_{nr_n}), \text{ and}$$

$$\bar{b} = (b_{11}, \ldots, b_{1r_1}, b_{21}, \ldots, b_{2r_2}, \ldots, b_{n1}, \ldots, b_{nr_n})$$

on $\mathbf{I}$ such that $\bar{a} \in R^{\mathbf{I}}$, $\bar{b} \notin R^{\mathbf{I}}$ and $\bar{a} \equiv^{k_1+k_2+\cdots+k_n} \bar{b}$. Since $\bar{a} \equiv^{k_1+k_2+\cdots+k_n} \bar{b}$, for $1 \leq i \leq n$, it holds that $(a_{i1}, \ldots, a_{ir_i}) \equiv^{k_i} (b_{i1}, \ldots, b_{ir_i})$. But then, by definition of $R^{\mathbf{I}}$, we have that $\bar{b} \in R^{\mathbf{I}}$ which contradicts our hypothesis. □

Fact 7.21. *Let M be a relational machine of arity k and let σ and τ be the vocabularies of the input structure and the rs, respectively. In every configuration in a computation of M on an input structure $\mathbf{I}$, every relation R_i of arity $r_i \leq k$ held in the rs of M is closed under the equivalence relation $\equiv^k$ on $\mathbf{I}$.*

Proof. We proceed by induction on the sequence of configurations of a computation of M on an input structure $\mathbf{I}$. W.l.o.g. we can assume that the relations of the input structure $\mathbf{I}$, i.e., the relations in the *rs* which interpret the relation symbols in σ, are not modified through any computation of M. This poses no problem since in case that such a modification is needed, we can just get another copy of the relation which would be in $\sigma \setminus \tau$ and modify the new relation instead.

In the initial configuration of M, the τ-structure $\mathbf{A}_0$ held in the *rs* is $\mathbf{I}$ extended with an empty relation $R_i^{\mathbf{A}_0}$ for each relation symbol R_i in $\tau \setminus \sigma$. Note that, by Fact 7.16, if a relation R of arity $r \leq k$ is closed under $\equiv^r$, then R is also closed under $\equiv^k$, hence we only need to show that for every relation symbol $R_i \in \sigma$ of arity r_i, the relation $R_i^{\mathbf{I}}$ is closed under $\equiv^{r_i}$ on $\mathbf{I}$. Let us assume that there is a $R_i \in \sigma$ such that $R_i^{\mathbf{I}}$ is not closed under $\equiv^{r_i}$ on $\mathbf{I}$. If that is the case, there are two r_i-tuples $\bar{a}, \bar{b} \in I^{r_i}$ such that $\bar{a} \in R_i^{\mathbf{I}}$, $\bar{b} \notin R_i^{\mathbf{I}}$ and $\bar{a} \equiv^{r_i} \bar{b}$. But then there is a FO^{r_i}-formula of vocabulary σ, namely $R_i(x_1, \ldots, x_{r_i})$, such that $\mathbf{I} \models R_i(x_1, \ldots, x_{r_i})[\bar{a}]$ while $\mathbf{I} \not\models R_i(x_1, \ldots, x_{r_i})[\bar{b}]$, which clearly contradicts our assumption that $\bar{a} \equiv^{r_i} \bar{b}$.

For a configuration other than the initial configuration, we assume as inductive hypothesis that, in the preceding configuration in the sequence, every relation $R_i^{\mathbf{A}_{n-1}}$ of arity $r_i \leq k$ in the τ-structure $\mathbf{A}_{n-1}$ held in the *rs* of M, is closed under $\equiv^k$ on $\mathbf{I}$. Let $\mathbf{A}_n$ be the τ-structure held in the *rs* of M in the current configuration. Note that in each step from a configuration to the next M updates exactly one relation in its *rs*. Let $R_x^{\mathbf{A}_n}$ of arity r_x be the relation updated in the step from the previous to the current configuration in the sequence, and let $\varphi_{R_x} \in \mathrm{FO}^k[\tau]$ be the formula used for the update. We show below that there is a formula $\varphi'_{R_x} \in \mathrm{FO}^k[\sigma]$ such that $\varphi_{R_x}^{\mathbf{A}_{n-1}} = \varphi'^{\mathbf{I}}_{R_x}$, i.e., φ'_{R_x} defines in $\mathbf{I}$ the same relation that φ_{R_x} defines in $\mathbf{A}_{n-1}$. Since the FO^k-formula φ'_{R_x} of vocabulary σ defines on $\mathbf{I}$ the relation $R_x^{\mathbf{A}_n}$, it follows by Fact 7.17 that $R_x^{\mathbf{A}_n}$ is closed under $\equiv^k$ on $\mathbf{I}$.

By inductive hypothesis, if $R_i^{\mathbf{A}_{n-1}}$ is a relation of arity r_i in $\mathbf{A}_{n-1}$, then it is closed under $\equiv^k$ on $\mathbf{I}$. Let $T_{R_i} = \{tp_{\mathbf{I}}^{\mathrm{FO}^k}(\bar{a}) : \bar{a} \in R_i^{\mathbf{A}_{n-1}}\}$ be the set of FO^k-types realized by $R_i^{\mathbf{A}_{n-1}}$ on $\mathbf{I}$. By Lemma 2.5, for every type $t_j \in T_{R_i}$, there is an FO^k formula α_{t_j} of vocabulary σ which isolates t_j. Therefore, the FO^k-formula $\psi_{R_i} \equiv \bigvee_{t_j \in T_{R_i}} \alpha_{t_j}$

of vocabulary σ defines $R_i^{\mathbf{A}_{n-1}}$ on $\mathbf{I}$. The formula φ'_{R_x} is built from φ_{R_x} with every occurrence of a sub-formula of the form $R_i(x_1, \ldots, x_{r_i})$, where $R_i \in \tau \setminus \sigma$, replaced by the corresponding sub-formula $\psi_{R_i}(x_1, \ldots, x_{r_i})$. $\square$

We complete the proof of Theorem 7.14 showing that every NP_r property of finite relational structures can be expressed in $\Sigma_1^{1,\omega}$. The proof is close to the proof of Fagin's theorem in [Lib04], but we need to bear in mind that we can only quantify relational variables which are closed under the equivalence relation $\equiv^k$ for some k, and we have to take into account the rs of the machine.

Proposition 7.22. *Every class of relational structures (relational language) in* NP_r *is definable in* $\Sigma_1^{1,\omega}$.

Proof. Let $q : \mathcal{B}_\sigma \to \{0,1\}$ be a Boolean query which is computed by a nondeterministic relational machine $M = \langle Q, \Sigma, \delta, q_0, \mathfrak{b}, F, \sigma, \tau, R_l, \Omega, \Phi \rangle$ of arity k, working in polynomial time in the k-size of the input structure of vocabulary σ. We assume that M works in time $(size_k(\mathbf{I}))^s$ for some $s \geq 1$ and $\mathbf{I} \in \mathcal{B}_\sigma$. Here

- $Q = \{q_0, \ldots, q_m\}$ is the finite set of internal states; $q_0 \in Q$ is the initial state;
- $\Sigma = \{0, 1, \mathfrak{b}\}$ is the tape alphabet; $\mathfrak{b} \in \Sigma$ is the symbol denoting blank;
- $F = \{q_m\}$ is the set of accepting final states;
- $\tau = \{R_0, \ldots, R_l\}$ is the vocabulary of the rs; r_i $(0 \leq i \leq l)$ is the arity of R_i;
- $\sigma = \{R_0, \ldots, R_u\}$, where $u < l$, is the vocabulary of the input structure;
- R_l is the output relation;
- $\Omega = \{\alpha_0, \ldots, \alpha_v\}$ is a finite set of first-order sentences of vocabulary τ;
- $\Phi = \{\gamma_0, \ldots, \gamma_w\}$ is a finite set of first-order formulas of vocabulary τ; and
- $\delta : Q \times \Sigma \times \Omega \to \mathcal{P}(\Sigma \times Q \times \{R, L\} \times \Phi \times \tau)$ is the transition function of M.

The sentence φ_M expressing acceptance by M on input $\mathbf{I} \in \mathcal{B}_\sigma$ has the form

$$\exists^{2\cdot k} O \, \exists^{2\cdot s\cdot k} T_0 \, \exists^{2\cdot s\cdot k} T_1 \, \exists^{2\cdot s\cdot k} T_{\mathfrak{b}} \, \exists^{2\cdot s\cdot k} H_{q_0} \ldots \exists^{2\cdot s\cdot k} H_{q_m} \, \exists^{(s+1)\cdot k} S_0 \ldots \exists^{(s+1)\cdot k} S_l \, (\psi),$$

where ψ is a first-order formula of vocabulary $\{O, T_0, T_1, T_{\mathfrak{b}}, H_{q_0}, \ldots, H_{q_m}, S_0, \ldots, S_l\} \cup \sigma$. The arity of O is $2 \cdot k$; the arity of $T_0, T_1, T_{\mathfrak{b}}$ as well as the arity of H_{q_i} for

$0 \leq i \leq m$, is $2 \cdot s \cdot k$; and the arity of S_i for $0 \leq i \leq l$ is $r_i + s \cdot k$. The intended interpretation of these relation symbols is as follows:

- O is a preorder of k-tuples over $\mathbf{I}$.

Note that, by Theorem 2.7, there is a preorder $\leq^k$ such that the corresponding equivalence relation is $\equiv^k$, and that such preorder is also a linear order over the set of equivalence classes of k-tuples $\mathcal{C}$ determined by $\equiv^k$. Thus, with $\leq^k$ we can define a lexicographic linear order of the s-tuples in $\mathcal{C}^s$. Since M runs in time bounded by $(size_k(\mathbf{I}))^s$ and visits at most $(size_k(\mathbf{I}))^s$ cells, we can model time ($\bar{t}$) as well as position on the tape ($\bar{p}$) by s-tuples of equivalence classes in $\mathcal{C}$. We actually do that by using $(s \cdot k)$-tuples of elements of the domain I instead of s-tuples of equivalence classes in $\mathcal{C}$. Under this approach, two $(s \cdot k)$-tuples $\bar{a} = (\bar{a}_1, \ldots, \bar{a}_s)$ and $\bar{b} = (\bar{b}_1, \ldots, \bar{b}_s)$ are considered equivalent iff, for $1 \leq i \leq s$, their corresponding k-tuples $\bar{a}_i$ and $\bar{b}_i$ belong to the same equivalence class, i.e., iff $\bar{a}_i \leq^k \bar{b}_i$ and $\bar{b}_i \leq^k \bar{a}_i$.

Having this considerations in mind, we define the intended interpretation of the remaining relation symbols as follows:

- T_0, T_1, and $T_\mathfrak{b}$ are *tape* relations; for $x \in \{0, 1, \mathfrak{b}\}$, $T_x(\bar{p}, \bar{t})$ indicates that position $\bar{p}$ at time $\bar{t}$ contains x.

- H_q's are *head* relations; for $q \in Q$, $H_q(\bar{p}, \bar{t})$ indicates that at time $\bar{t}$, the machine M is in state q, and its head is in position $\bar{p}$.

- S_i's are rs relations; for $0 \leq i \leq l$, $S_i(\bar{a}, \bar{t})$ indicates that at time $\bar{t}$, the relation R_i in the rs contains the r_i-tuple $\bar{a}$.

The sentence ψ must now express that when M starts with an empty tape and an input $\mathbf{I}$ in the designated relations of its rs, the relations T_x's, H_q's and S_i's encode its computation, and eventually M reaches an accepting state.

We define ψ to be the conjunction of the following sentences:

- A sentence expressing that O defines a pre-order of k-tuples.
  $\forall \bar{x}(O(\bar{x}, \bar{x}))$ "O is reflexive" $\wedge$
  $\forall \bar{x}\bar{y}\bar{z}(O(\bar{x}, \bar{y}) \wedge O(\bar{y}, \bar{z}) \rightarrow O(\bar{x}, \bar{z}))$ "O is transitive" $\wedge$
  $\forall \bar{x}\bar{y}(O(\bar{x}, \bar{y}) \vee O(\bar{y}, \bar{x}))$ "O is connex".

- A sentence defining the initial configuration of M.
  $\exists\bar{x}\forall\bar{y}\Big(\bar{x}\preceq\bar{y}) \rightarrow \big(H_{q_0}(\bar{x},\bar{x}) \wedge \forall\bar{p}(T_{\mathfrak{b}}(\bar{p},\bar{x}))\big)\Big)$
  "At time 0, M is in state q_0, the head is in the left-most position of the tape, and the tape contains only blanks" $\wedge$
  $\exists\bar{t}\forall\bar{x}\Big(\bar{t}\preceq\bar{x} \rightarrow \bigwedge_{0\leq i\leq u}\big(\forall a_1\ldots a_{r_i}(S_i(a_1,\ldots,a_{r_i},\bar{t}) \leftrightarrow R_i(a_1,\ldots,a_{r_i}))\big)\wedge$
  $\bigwedge_{u<i\leq l}\big(\forall a_1\ldots a_{r_i}(\neg S_i(a_1,\ldots,a_{r_i},\bar{t}))\big)\Big)$
  "the relations in the rs hold a τ-structure $\mathbf{A}$ which extends $\mathbf{I}$ with an empty relation $S_i^{\mathbf{A}}$ for each relation symbol R_i in $\tau\setminus\sigma$".
  Here, for $\bar{x}=(\bar{x}_1,\ldots,\bar{x}_s)$ and $\bar{y}=(\bar{y}_1,\ldots,\bar{y}_s)$, where $\bar{x}_i$ and $\bar{y}_i$ $(1\leq i\leq s)$ are k-tuples of individual variables, we say that $\bar{x}\preceq\bar{y}$ iff
  $\big(O(\bar{x}_1,\bar{y}_1)\wedge\neg O(\bar{y}_1,\bar{x}_1)\big)\vee$
  $\big(O(\bar{x}_1,\bar{y}_1)\wedge O(\bar{y}_1,\bar{x}_1)\wedge O(\bar{x}_2,\bar{y}_2)\wedge\neg O(\bar{y}_2,\bar{x}_2)\big)\vee\ldots\vee$
  $\big(O(\bar{x}_1,\bar{y}_1)\wedge O(\bar{y}_1,\bar{x}_1)\wedge\ldots\wedge O(\bar{x}_{s-1},\bar{y}_{s-1})\wedge O(\bar{y}_{s-1},\bar{x}_{s-1})\wedge$
  $O(\bar{x}_s,\bar{y}_s)\wedge\neg O(\bar{y}_s,\bar{x}_s)\big)\vee\bar{x}\sim\bar{y},$
  where $\bar{x}\sim\bar{y}$ is simply $O(\bar{x}_1,\bar{y}_1)\wedge O(\bar{y}_1,\bar{x}_1)\wedge\ldots\wedge O(\bar{x}_s,\bar{y}_s)\wedge O(\bar{y}_s,\bar{x}_s)$. Informally, $\bar{x}\preceq\bar{y}$ if $\bar{x}$ precedes or equals $\bar{y}$ in the lexicographic order induced by O, and $\bar{x}\sim\bar{y}$ if they share the same position.

- A sentence stating that in every configuration of M, each cell of the tape contains exactly one element of Σ.
  $\forall\bar{p}\bar{t}\Big(\big(T_0(\bar{p},\bar{t})\leftrightarrow\forall\bar{x}\bar{y}(\bar{x}\sim\bar{p}\wedge\bar{y}\sim\bar{t}\rightarrow T_0(\bar{x},\bar{y})\wedge\neg T_1(\bar{x},\bar{y})\wedge\neg T_{\mathfrak{b}}(\bar{x},\bar{y}))\big)\wedge$
  $\big(T_1(\bar{p},\bar{t})\leftrightarrow\forall\bar{x}\bar{y}(\bar{x}\sim\bar{p}\wedge\bar{y}\sim\bar{t}\rightarrow T_1(\bar{x},\bar{y})\wedge\neg T_0(\bar{x},\bar{y})\wedge\neg T_{\mathfrak{b}}(\bar{x},\bar{y}))\big)\wedge$
  $\big(T_{\mathfrak{b}}(\bar{p},\bar{t})\leftrightarrow\forall\bar{x}\bar{y}(\bar{x}\sim\bar{p}\wedge\bar{y}\sim\bar{t}\rightarrow T_{\mathfrak{b}}(\bar{x},\bar{y})\wedge\neg T_0(\bar{x},\bar{y})\wedge\neg T_1(\bar{x},\bar{y}))\big)\Big).$

- A sentence stating that at any time the machine is in exactly one state.
  $\forall\bar{t}\exists\bar{p}\Big(\bigvee_{q\in Q}\big(H_q(\bar{p},\bar{t})\wedge\forall\bar{x}(H_q(\bar{x},\bar{t})\leftrightarrow\bar{x}\sim\bar{p})\big)\Big)\wedge$
  $\neg\exists\bar{p}\bar{t}\bar{x}\bar{y}\Big(\bigvee_{q,q'\in Q,q\neq q'}\big(H_q(\bar{p},\bar{t})\wedge\bar{p}\sim\bar{x}\wedge\bar{t}\sim\bar{y}\wedge H_{q'}(\bar{x},\bar{y})\big)\Big).$

- Sentences expressing that the relations T_i's, H_q's and S_i's respect the transitions of M. For every $a\in\Sigma$, $q\in Q$ and $\alpha\in\Omega$ for which the transition function δ is defined, we have a sentence of the form
  $$\bigvee_{(b,q',m,\gamma,R)\in\delta(q,a,\alpha)}\chi(q,a,\alpha,b,q',m,\gamma,R),$$

where $\chi(q, a, \alpha, b, q', m, \gamma, R)$ is the sentence describing the transition in which, upon reading a in state q, if α evaluates to true in the τ-structure $\mathbf{A}$ currently held in the rs, then the machine writes b, enters state q', makes the move m, and replaces the relation $R^{\mathbf{A}}$ by $\gamma^{\mathbf{A}}$ in the rs. Assume that $m = L$ and S_j is the relation variable which encodes R, we write $\chi(q, a, \alpha, b, q', m, \gamma, R)$ as the conjunction of:

$$\begin{aligned}
&\forall \bar{p} \forall \bar{t} \Big(\neg(\forall \bar{x}(\bar{p} \preceq \bar{x})) \wedge T_a(\bar{p}, \bar{t}) \wedge H_q(\bar{p}, \bar{t}) \wedge \hat{\alpha}(\bar{t}) \rightarrow \\
&\qquad \Big(T_b(\bar{p}, \bar{t}+1) \wedge H_{q'}(\bar{p}-1, \bar{t}+1) \wedge \\
&\qquad\quad \forall \bar{x} \big(\neg(\bar{x} \sim \bar{p}) \rightarrow \big(\textstyle\bigwedge_{i \in \{0,1,\mathfrak{b}\}} T_i(\bar{x}, \bar{t}+1) \leftrightarrow T_i(\bar{x}, \bar{t}) \big) \big) \wedge \\
&\qquad\quad \forall x_1 \ldots x_{r_j} \big(S_j(x_1, \ldots, x_{r_j}, \bar{t}+1) \leftrightarrow \hat{\gamma}(x_1, \ldots, x_{r_j}, \bar{t}) \wedge \\
&\qquad\quad \textstyle\bigwedge_{0 \leq i \leq l, i \neq j} \big(\forall x_1 \ldots x_{r_i} (S_i(x_1, \ldots, x_{r_i}, \bar{t}) \leftrightarrow S_i(x_1, \ldots, x_{r_i}, \bar{t}+1)) \big) \Big) \Big)
\end{aligned}$$

and

$$\begin{aligned}
&\forall \bar{p} \forall \bar{t} \Big(\forall \bar{x}(\bar{p} \preceq \bar{x}) \wedge T_a(\bar{p}, \bar{t}) \wedge H_q(\bar{p}, \bar{t}) \wedge \hat{\alpha}(\bar{t}) \rightarrow \\
&\qquad \Big(T_b(\bar{p}, \bar{t}+1) \wedge H_{q'}(\bar{p}, \bar{t}+1) \wedge \\
&\qquad\quad \forall \bar{x} \big(\neg(\bar{x} \sim \bar{p}) \rightarrow \big(\textstyle\bigwedge_{i \in \{0,1,\mathfrak{b}\}} T_i(\bar{x}, \bar{t}+1) \leftrightarrow T_i(\bar{x}, \bar{t}) \big) \big) \wedge \\
&\qquad\quad \forall x_1 \ldots x_{r_j} \big(S_j(x_1, \ldots, x_{r_j}, \bar{t}+1) \leftrightarrow \hat{\gamma}(x_1, \ldots, x_{r_j}, \bar{t}) \wedge \\
&\qquad\quad \textstyle\bigwedge_{0 \leq i \leq l, i \neq j} \big(\forall x_1 \ldots x_{r_i} (S_i(x_1, \ldots, x_{r_i}, \bar{t}) \leftrightarrow S_i(x_1, \ldots, x_{r_i}, \bar{t}+1)) \big) \Big) \Big)
\end{aligned}$$

where $\hat{\alpha}(\bar{t})$ and $\hat{\gamma}(x_1, \ldots, x_{r_j}, \bar{t})$ are the formulae obtained by replacing in α and $\gamma(x_1, \ldots, x_{r_j})$, respectively, each atomic sub-formula of the form $R_i(y_1, \ldots, y_{r_i})$ $(0 \leq i \leq l)$ by $S_i(y_1, \ldots, y_{r_i}, \bar{t})$. We use abbreviations $\bar{p} - 1$ and $\bar{t} + 1$ for the predecessor of $\bar{p}$ and the successor of $\bar{t}$ in the lexicographic order induced by O, respectively; these are clearly definable in first-order logic. The second formula above is very similar to the first one, and handles the case when $\bar{p}$ is the left-most cell of the tape: then the head does not move and stays in p.

- Finally, a sentence stating that at some point, M is in an accepting final state.

$$\exists \bar{p} \exists \bar{t} (H_{q_m}(p, t)).$$

We show next that, M accepts a given σ-structure $\mathbf{I}$ iff there are relations closed under equivalence of FO^k-types of tuples as required by the SO$^\omega$ quantifiers in the prefix of φ_M, which assigned to the relation variables $O, T_0, T_1, T_{\mathfrak{b}}, H_{q_0}, \ldots, H_{q_m}, S_0, \ldots, S_l$ satisfy ψ.

Let $\mathcal{C}$ be the set of equivalence classes of k-tuples on $\mathbf{I}$ determined by the equivalence relation $\equiv^k$. Let $\leq^k$ be the partial order of Theorem 2.7 and let "$\leq^k$" be $\{([(a_{11}, \ldots, a_{1k})], [(a_{21}, \ldots, a_{2k})]) \in \mathcal{C}^2 : (a_{11}, \ldots, a_{1k}, a_{21}, \ldots, a_{2k}) \in \leq^k\}$. Since the relation "$\leq^k$" is a subset of $\mathcal{C}^2$, it follows from Fact 7.20 that there is a relation with the intended interpretation for O which is closed under $\equiv^{2k}$ on $\mathbf{I}$.

Regarding $T_0, T_1, T_{\mathfrak{b}}, H_{q_0}, \ldots, H_{q_m}$. If M accepts a given σ-structure $\mathbf{I}$, then for each $R_i \in \{T_0, T_1, T_{\mathfrak{b}}, H_{q_0}, \ldots, H_{q_m}\}$ there is a $\mathcal{C}_{R_i} \subseteq \mathcal{C}^{2 \cdot s}$ such that the corresponding relation $R_i^{\mathbf{I}} = \{(a_{11}, \ldots, a_{1k}, a_{21}, \ldots, a_{2k}, \ldots, a_{(2 \cdot s)1}, \ldots, a_{(2 \cdot s)k}) \in I^{2 \cdot s \cdot k} : ([(a_{11}, \ldots, a_{1k})], [(a_{21}, \ldots, a_{2k})], \ldots, [(a_{(2 \cdot s)1}, \ldots, a_{(2 \cdot s)k})]) \in \mathcal{C}_{R_i}\}$ meets the intended interpretation and, by Fact 7.20, is closed under $\equiv^{2 \cdot s \cdot k}$ on $\mathbf{I}$.

For each $R_i \in \tau$ of arity r_i, let $\mathcal{C}_i$ be the set of equivalence classes of r_i-tuples determined by $\equiv^k$ on $\mathbf{I}$. By Fact 7.21, in every configuration in a computation of a relational machine M of arity k on an input σ-structure $\mathbf{I}$, every relation R_i of arity r_i in its rs is closed under $\equiv^k$ on $\mathbf{I}$. Thus, if M accepts a given σ-structure $\mathbf{I}$, then for each relation variable $S_0, \ldots, S_l$ used to model the content of the relational store, there is a $\mathcal{C}_{S_i} \subseteq \mathcal{C}_i \times \mathcal{C}^s$ such that the corresponding relation

$$S_i^{\mathbf{I}} = \{(a_1, \ldots, a_{r_i}, a_{11}, \ldots, a_{1k}, a_{21}, \ldots, a_{2k}, \ldots, a_{s1}, \ldots, a_{sk}) \in I^{r_i + s \cdot k} : \\ ([(a_1, \ldots, a_{r_i})], [(a_{11}, \ldots, a_{1k})], [(a_{21}, \ldots, a_{2k})], \ldots, [(a_{s1}, \ldots, a_{sk})]) \in \mathcal{C}_{S_i}\},$$

meets the intended interpretation. Again by Fact 7.20, each of these relations $S_i^{\mathbf{I}}$ is closed under $\equiv^{(s+1) \cdot k}$ on $\mathbf{I}$.

We conclude that, for every $\mathbf{I} \in \mathcal{B}_\sigma$, $\mathbf{I} \models \varphi_M$ iff M accepts $\mathbf{I}$. $\square$

7.3.1 SO$^\omega$ Captures the Relational Polynomial-Time Hierarchy

We show in this section the exact correspondence between the prenex fragments of SO$^\omega$ and the levels of the *relational* polynomial-time hierarchy.

To prove the following lemma, we adapt the strategy used for Turing machines in [Sto76] to the case of relational machines.

Lemma 7.23. *If M is a nondeterministic relational machine with an oracle $\mathcal{A}$ in $\Sigma_m^{\mathrm{P}_r}$, for some $m \geq 0$, then there is a nondeterministic relational machine M' which is equivalent to M and which, in any computation, asks at most one query to an* *oracle $\mathcal{A}'$ which is also in $\Sigma_m^{\mathrm{P}_r}$.*

Proof. We assume that $M = \langle Q, \Sigma, \delta, q_0, \mathfrak{b}, F, \sigma, \tau, T, \Omega, \Phi \rangle$ works in nondeterministic time bounded by $(size_k(\mathbf{I}))^s$ for some $s \geq 1$ and input relational structure $\mathbf{I}$ of vocabulary $\sigma = \{E_1, \ldots, E_l\}$. We also assume that the subset of distinguished oracle relation symbols in the vocabulary τ of M is $\sigma^o = \{R_1^o, \ldots, R_n^o\}$, where for $1 \leq i \leq n$, the arity of R_i^o is r_i. We denote as $M_{\mathcal{A}}$ the relational machine in $\Sigma_m^{\mathrm{P}_r}$ which decides $\mathcal{A}$. We assume that the arity of $M_{\mathcal{A}}$ is k' and that it works in time bounded by $(size_{k'}(\mathbf{I}_o))^{s'}$ for some $s' \geq 1$ and input relational structure $\mathbf{I}_o$ of vocabulary σ^o.

M' works as follows. First, it guesses a sequence of oracle queries, i.e., a sequence of σ^o-structures, as well as their corresponding answers. M' does this by guessing and writing over its Turing machine tape a sequence of tuples of the form $((\bar{a}_1, \ldots, \bar{a}_n), A)$, where A is either Y or N and for $1 \leq i \leq n$, $\bar{a}_i \in \{0,1\}^{size_{r_i}(\mathbf{I})}$. Each n-tuple $(\bar{a}_1, \ldots, \bar{a}_n)$ represents a query to the oracle and A is the answer to that query. The σ^o-structure corresponding to a given n-tuples $(\bar{a}_1, \ldots, \bar{a}_n)$ is obtained by interpreting $R_i^o \in \sigma^o$ with the following corresponding relation

$$\{\bar{b} \in (dom(\mathbf{I}))^{r_i} : \bar{b} \text{ is in the } j\text{-th equivalence class in the order given by } \leq^{r_i} \text{ and}$$
$$\text{the } j\text{-th component of } \bar{a}_i \text{ equals } 1\}.$$

Note that, as shown in the proof of Proposition 7.19, given a tuple $\bar{a}_i \in \{0,1\}^{size_{r_i}(\mathbf{I})}$, M' can compute the corresponding relation and store it in its rs working in time bounded by a polynomial in $size_{r_i}(\mathbf{I})$.

Then, M' proceeds as M, except that every time that M makes a query to the oracle, M' just takes the answer guessed for that query at the beginning of the computation.

Let us fix an arbitrary computation of M' on the input structure $\mathbf{I}$. At the end of the computation, M' must check that the guessed queries match the sequence of real queries of M in the fixed computation, and that the guessed answers for those queries coincide with the actual answers from the oracle $\mathcal{A}$. If any of those conditions is not met, M' stops in a rejecting state.

To check whether the guessed yes answers are correct, M' simply takes each guessed tuple of the form $((\bar{a}_1, \ldots, \bar{a}_n), Y)$, stores their corresponding relations in its rs, and works as $M_{\mathcal{A}}$ to check whether the structure of domain $dom(\mathbf{I})$ formed by those relations, is in the oracle $\mathcal{A}$. M' does not need to call an oracle to do this since the answer guessed for those queries is yes.

As to the queries for which the guessed answer is no, M' does need to use an oracle, but it suffices to make just one query to it. To do this, M' encodes the guessed oracle queries with guessed answers no as a structure of vocabulary $\sigma^N = \{E_1^o, \ldots, E_l^o, S_1^o, \ldots, S_n^o\}$, where for $1 \leq i \leq j$, $arity(E_i^o) = arity(E_i)$, and for $1 \leq i \leq n$, $arity(S_i^o) = r_i + s \cdot k$. Actually, σ^N is the set of distinguished oracle relation symbols in the vocabulary τ' of the rs of M'.

Let $(\bar{a}_{11}, \ldots, \bar{a}_{1n}), \ldots, (\bar{a}_{m1}, \ldots, \bar{a}_{mn})$ be the sequence of tuples corresponding to the queries with guessed negative answers, for each $1 \leq u \leq n$, M' stores in its rs the following relation:

$$\begin{aligned} S_u^o := \bigcup_{1 \leq i \leq m} \{(\bar{b}, \bar{t}) \in (dom(\mathbf{I}))^{r_u + s \cdot k} : \; & \bar{b} \text{ is in the } j\text{-th equivalence class in the} \\ & \text{order given by } \leq^{r_u}, \text{ the } j\text{-th component of } \bar{a}_{iu} \text{ equals } 1, \text{ and} \\ & \bar{t} \text{ belongs to the } i\text{-th equivalence class in the order given by } \leq^{s \cdot k}\}. \end{aligned}$$

Furthermore, for $1 \leq i \leq l$, M' stores in E_i^o in its rs the relation $E_i^{\mathbf{I}}$ in the input structure $\mathbf{I}$. Then it asks the oracle $\mathcal{A}'$ the query represented by the σ^N-structure of domain $dom(\mathbf{I})$ formed by those relations. If the answer of the oracle $\mathcal{A}'$ is yes, then the no answers guessed by M' are all correct.

Let $\mathbf{J} \in \mathcal{B}_{\sigma^N}$, let $\mathbf{J}_\sigma$ be the structure $\mathbf{J}$ restricted to the vocabulary $\{E_1^o, \ldots, E_l^o\}$, and let $\mathcal{C}$ be the set of equivalence classes of k-tuples determined by the equivalence relation $\equiv^k$ on $\mathbf{J}_\sigma$. For $([\bar{t}_1], \ldots, [\bar{t}_s]) \in \mathcal{C}^s$ and $1 \leq i \leq n$, we denote as $T^{\mathbf{J}}_{i,([\bar{t}_1], \ldots, [\bar{t}_s])}$ the following relation:

$$\begin{aligned} \{(a_1, \ldots, a_{r_i}) \in (dom(\mathbf{J}))^{r_i} : & \text{ for } \bar{t}'_1 \in [\bar{t}_1], \ldots, \bar{t}'_s \in [\bar{t}_s], \\ & \text{it holds that } \mathbf{J} \models S_i^o(a_1, \ldots, a_{o_i}, \bar{t}'_1, \ldots, \bar{t}'_s)\} \end{aligned}$$

Accordingly, we denote as $\mathbf{J}_{(\bar{t}_1, \ldots, \bar{t}_s)}$ the structure of vocabulary σ^o and domain $dom(\mathbf{J})$ which is obtained by interpreting the relation symbols $R_1^o, \ldots, R_n^o$ with the relations $T^{\mathbf{J}}_{1,([\bar{t}_1], \ldots, [\bar{t}_s])}, \ldots T^{\mathbf{J}}_{n,([\bar{t}_1], \ldots, [\bar{t}_s])}$, respectively.

A given structure $\mathbf{J} \in \mathcal{B}_{\sigma^N}$ belongs to the new oracle $\mathcal{A}'$ iff, for every s-tuple $([\bar{t}_1], \ldots, [\bar{t}_s])$ of equivalence classes in $\mathcal{C}^s$, it holds that $\mathbf{J}_{(\bar{t}_1, \ldots, \bar{t}_s)} \notin \mathcal{A}$.

The Machine $M_{\mathcal{A}'}$ which decides the relational language $\mathcal{A}'$ works as follows:

1. $M_{\mathcal{A}'}$ computes the preorder $\leq^k$ of Theorem 2.7 on $\mathbf{J}_\sigma$.

2. Using $\leq^k$, $M_{\mathcal{A}'}$ computes $size_k(\mathbf{J}_\sigma)$.

3. Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_s)$ and $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_s)$ be $(k \cdot s)$-tuples, we say that $\bar{x} \preceq \bar{y}$ if $([\bar{x}_1], \ldots, [\bar{x}_s])$ precedes or equals $([\bar{y}_1], \ldots, [\bar{y}_s])$ in the lexicographic order induced by $\leq^k$ on $\mathcal{C}^s$. If $\bar{x} \preceq \bar{y}$ and $\bar{y} \preceq \bar{x}$, then we say that $\bar{x} \sim \bar{y}$.

```
$X := \forall \bar{y}(\bar{x} \preceq \bar{y})$;
$Y := \neg \bar{x} = \bar{x}$;
While $\exists \bar{x}(\neg Y(\bar{x}))$;
Begin
        $R_1 := \exists \bar{x}(X(\bar{x}) \wedge S^o_1(y_1, \ldots, y_{r_1}, \bar{x}))$;
        $\vdots$
        $R_n := \exists \bar{x}(X(\bar{x}) \wedge S^o_n(y_1, \ldots, y_{r_n}, \bar{x}))$;
        $M_{\mathcal{A}'}$ works as $M_{\mathcal{A}}$ taking as input the $\sigma^o$-structure of domain $dom(\mathbf{J})$
        formed by the relations $R_1, \ldots, R_n$ held in its $rs$;
        If "$M_{\mathcal{A}}$ accepts" then
            $M_{\mathcal{A}'}$ stops in a rejecting state;
        $Y := Y(\bar{x}) \vee X(\bar{x})$;
        $X := \neg Y(\bar{x}) \wedge$
                $\forall \bar{y}\big((Y(\bar{y}) \wedge \forall \bar{z}(Y(\bar{z}) \to \bar{z} \preceq \bar{y})) \to$
                    $(\bar{y} \preceq \bar{x} \wedge \forall \bar{z}((\bar{y} \preceq \bar{z} \wedge \bar{z} \preceq \bar{x}) \to (\bar{y} \sim \bar{z} \vee \bar{z} \sim \bar{x})))\big)$;
End;
$M_{\mathcal{A}'}$ stops in an accepting state.
```

Since the "while loop" in the previous algorithm is executed $(size_k(\mathbf{J}_\sigma))^s$ times, it is not difficult to see that $M_{\mathcal{A}'}$ works in nondeterministic time bounded by a polynomial in $size_{k'}(\mathbf{J})$, where $k' \geq s \cdot k$ is the arity of $M_{\mathcal{A}'}$. $\square$

Theorem 7.24. *For $m \geq 1$, $\Sigma^{1,\omega}_m$ captures $\Sigma^{\mathrm{P}_r}_m$.*

Proof. **a) $\Longrightarrow$:** First, we show that for every relational vocabulary σ, every $\Sigma^{1,\omega}_m[\sigma]$-sentence φ can be evaluated in $\Sigma^{\mathrm{P}_r}_m$.

Suppose that φ is $\exists^{k_{11}} X_{11} \ldots \exists^{k_{1s_1}} X_{1s_1} \forall^{k_{21}} X_{21} \ldots \forall^{k_{2s_2}} X_{2s_2} \exists^{k_{31}} X_{31} \ldots \exists^{k_{3s_3}} X_{3s_3} \ldots Q^{k_{m1}} X_{m1} \ldots Q^{k_{ms_m}} X_{ms_m}(\psi)$, where Q is either $\exists$ or $\forall$, depending on whether m is odd or even, respectively, and ψ is a first-order formula of vocabulary $\sigma \cup \{X_{11}, \ldots, X_{1s_1}, X_{21}, \ldots, X_{2s_2}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$.

We build a nondeterministic relational oracle machine M_φ which evaluates φ on input structures of vocabulary σ. For $1 \leq j \leq s_1$, let $k'_{1j} \geq 2k_{1j}$ be the arity of the relational machine $M_{\leq^{k_{1j}}}$ of Lemma 7.15 which computes the preorder $\leq^{k_{1j}}$ of

Theorem 2.7. The arity k of M_φ is $max(\{k'_{11}, \ldots, k'_{1s_1}\})$. The vocabulary τ of the relational store is $\sigma \cup \sigma^{o_{m-1}} \cup \{\leq^{k_{11}}, \ldots, \leq^{k_{1s_1}}, S_1, \ldots, S_{s_1}\}$, where for $1 \leq j \leq s_1$, the arity of $\leq^{k_{1j}}$ is $2k_{1j}$ and the arity of S_j is k_{1j}, and $\sigma^{o_{m-1}} = \{R^{o_{m-1}} : R \in \sigma\} \cup \{X_{11}^{o_{m-1}}, \ldots, X_{1s_1}^{o_{m-1}}\}$ is the set of distinguished oracle relation symbols. For every $R \in \sigma$, the arity of $R^{o_{m-1}}$ is the same as the arity of R, and for $1 \leq j \leq s_1$, the arity of $X_{1j}^{o_{m-1}}$ is the same as the arity $r_{1j} \leq k_{1j}$ of X_{1j}.

Let φ'_{m-1} be the following sentence:

$$\exists^{k_{21}} X_{21} \ldots \exists^{k_{2s_2}} X_{2s_2} \neg(\exists^{k_{31}} X_{31} \ldots \exists^{k_{3s_3}} X_{3s_3} \ldots Q^{k_{m1}} X_{m1} \ldots Q^{k_{ms_m}} X_{ms_m}(\psi'_{m-1})),$$

where ψ'_{m-1} is ψ with every occurrence of a relation symbol $R \in \sigma$ replaced by the corresponding relation symbol $R^{o_{m-1}} \in \sigma^{o_{m-1}}$, and every occurrence of a relation variable X_{1j} $(1 \leq j \leq s_1)$ replaced by the corresponding relation symbol $X_{1j}^{o_{m-1}} \in \sigma^{o_{m-1}}$. The oracle $\mathcal{C}_{m-1}$ of M_φ is the relational language $\{\mathbf{A} \in \mathcal{B}_{\sigma^{o_{m-1}}} : \mathbf{A} \models \varphi'_{m-1}\}$.

On an input structure $\mathbf{I}$, M_φ works as follows:

1. For every $1 \leq j \leq s_1$, M_φ builds the preorder $\leq^{k_{1j}}$ of Theorem 2.7 in its rs.

2. M_φ computes $size_{k_{1j}}(\mathbf{I})$ for every $1 \leq j \leq s_1$.

3. For every $1 \leq j \leq s_1$, M_φ guesses and writes over its Turing machine tape a tuple $\bar{a}_j \in \{0, 1\}^{size_{k_{1j}}(\mathbf{I})}$.

4. Using the binary tuples guessed in the previous step, M_φ generates, for every $1 \leq j \leq s_1$, a relation which is placed in the distinguish oracle relation X_{1j}^o of its rs and is closed under the equivalence relation $\equiv^{k_{1j}}$ in $\mathbf{I}$. M_φ works by storing in X_{1j}^o the tuples in all l-th equivalence classes in the order given by $\leq^{k_{1j}}$ for which the l-th component of $\bar{a}_j$ equals 1.

5. Finally, for every $R \in \sigma$, M_φ stores the relation $R^{\mathbf{I}}$ into the corresponding oracle relation $R^{o_{m-1}}$ and moves to the oracle query state $q_?$. M_φ accepts the input structure $\mathbf{I}$ iff the relational structure of domain I formed by the distinguished set of oracles relations currently held in its rs does not belongs to the oracle set $\mathcal{C}_{m-1}$, i.e., iff M_φ transfers from the state $q_?$ into the state q_{NO}.

As shown in the proof of Proposition 7.19, M_φ can perform tasks 1 to 4 working in time bounded by a polynomial in $size_k(\mathbf{I})$. Furthermore, task 5 can clearly be performed in constant time by M_φ.

Therefore, it only remains to show that the oracle $\mathcal{C}_{m-1}$ is in $\Sigma^{\mathrm{P}_r}_{m-1}$, i.e., that there is a nondeterministic relational machine $M_{\varphi'_{m-1}}$ such that $\mathcal{L}(M_{\varphi'_{m-1}}) = \mathcal{C}_{m-1}$ and $\mathcal{L}(M_{\varphi'_{m-1}}) \in \Sigma^{\mathrm{P}_r}_{m-1}$.

$M_{\varphi'_{m-1}}$ evaluates φ'_{m-1} on input structures of vocabulary $\sigma^{o_{m-1}}$. The vocabulary τ'_1 of the relational store is $\sigma^{o_{m-1}} \cup \sigma^{o_{m-2}} \cup \{\leq^{k_{21}}, \ldots, \leq^{k_{2s_2}}, S_1, \ldots, S_{s_2}\}$, where for $1 \leq j \leq s_2$, the arity of $\leq^{k_{2j}}$ is $2k_{2j}$ and the arity of S_j is k_{2j}, and $\sigma^{o_{m-2}} = \{R^{o_{m-2}} : R \in \sigma^{o_{m-1}}\} \cup \{X^{o_{m-2}}_{21}, \ldots, X^{o_{m-2}}_{2s_2}\}$ is the set of distinguished oracle relation symbols. For every $R \in \sigma^{o_{m-1}}$, the arity of $R^{o_{m-2}}$ is the same as the arity of R, and for $1 \leq j \leq s_2$, the arity of $X^{o_{m-2}}_{2j}$ is the same as the arity $r_{2j} \leq k_{2j}$ of X_{2j}.

The oracle $\mathcal{C}_{m-2}$ of $M_{\varphi'_{m-1}}$ is the relational language $\{\mathbf{A} \in \mathcal{B}_{\sigma^{o_{m-2}}} : \mathbf{A} \models \varphi'_{m-2}\}$, where φ'_{m-2} is $\exists^{k^3_1} X_{31} \ldots \exists^{k^3_{s_3}} X_{3s_3} \ldots Q^{k^m_1} X_{m1} \ldots Q^{k^m_{s_m}} X_{ms_m}(\psi'_{m-2})$. Here ψ'_{m-2} is ψ'_{m-1} with every occurrence of a relation symbol $R \in \sigma^{o_{m-1}}$ replaced by the corresponding relation symbol $R^{o_{m-2}} \in \sigma^{o_{m-2}}$, and every occurrence of a relation variable X_{2j} ($1 \leq j \leq s_2$) replaced by the corresponding relation symbol $X^{o_{m-2}}_{2j} \in \sigma^{o_{m-2}}$.

The way in which the machine $M_{\varphi'_{m-1}}$ works is exactly the same as the way in which the original machine M_φ works, i.e., $M_{\varphi'_{m-1}}$ executes steps 1 to 5 adapted to the vocabulary τ'_1 of its rs and for $1 \leq j \leq s_2$. Therefore, for every input structure $\mathbf{I}'$ of vocabulary $\sigma^{o_{m-1}}$, $M_{\varphi'_{m-1}}$ works in nondeterministic relational time bounded by a polynomial in $size_{k'}(\mathbf{I}')$, where $k' = max(\{k'_{21}, \ldots, k'_{2s_2}\})$ is the arity of $M_{\varphi'_{m-1}}$, and for $1 \leq j \leq s_2$, $k'_{2j} \geq 2k_{2j}$ is the arity of the relational machine $M_{\leq^{k_{2j}}}$ of Lemma 7.15 which computes the preorder $\leq^{k_{2j}}$ of Theorem 2.7.

This process continues in the same way for the blocks 3 to $m-1$ of quantifiers in φ. Since for the last block m of quantifiers the resulting φ'_1 is either

$$\exists^{k^m_1} X_{m1} \ldots \exists^{k^m_{s_m}} X_{ms_m}(\psi'_1) \qquad \text{or} \qquad \exists^{k^m_1} X_{m1} \ldots \exists^{k^m_{s_m}} X_{ms_m}(\neg\psi'_1),$$

it follows by Theorem 7.14 that the oracle $\mathcal{C}_1$ of $M_{\varphi'_2}$ (i.e., the relational language $\{\mathbf{A} \in \mathcal{B}_{\sigma^{o_1}} : \mathbf{A} \models \varphi'_1\}$) is in $\mathrm{NP}_r = \Sigma^{\mathrm{P}_r}_1$.

Hence, φ can be evaluated in $\Sigma^{\mathrm{P}_r}_m$.

b) $\Longleftarrow$: Next, we show that every $\Sigma^{\mathrm{P}_r}_m$ property of finite relational structures can be expressed in $\Sigma^{1,\omega}_m$.

We use induction on m. The base case is Proposition 7.22. Now consider a Boolean query $q : \mathcal{B}_\sigma \to \{0, 1\}$ in $\Sigma^{\mathrm{P}_r}_m$ where $m > 1$. Let M be the nondeterministic relational machine with an oracle in $\Sigma^{\mathrm{P}_r}_{m-1}$ which computes q. Let $\{\mathbf{I} \in \mathcal{B}_{\sigma^o} : q_o(\mathbf{I}) =$

$1\}$, where σ^o is the set of distinguished oracle relation symbols of M and q_o is a boolean query in $\Sigma^{\mathrm{P}_r}_{m-1}$, be the oracle of M.

By inductive hypothesis, for every boolean query q_i in $\Sigma^{P_r}_{m-1}$, there is a sentence $\alpha_{q_i} \in \Sigma^{1,\omega}_{m-1}$ which express q_i. In particular, there is a sentence $\alpha_{q_o} \in \Sigma^{1,\omega}_{m-1}$ of vocabulary σ^o, which express the boolean query q_o. Let α_{q_o} be σ^o-sentence $\exists^{k_{21}} X_{21} \ldots \exists^{k_{2s_2}} X_{2s_2} \forall^{k_{31}} X_{31} \ldots \forall^{k_{3s_3}} X_{3s_3} \ldots Q^{k_{m1}} X_{m1} \ldots Q^{k_{ms_m}} X_{ms_m}(\psi_o)$, where Q is either $\exists$ or $\forall$, depending on whether m is odd or even, respectively, and ψ_o is a first-order sentence of vocabulary $\sigma^o \cup \{X_{21}, \ldots, X_{2s_2}, X_{31}, \ldots, X_{3s_3}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$.

We show how to modify the formula φ_M in Proposition 7.22, i.e., the formula corresponding to the nondeterministic relational machine, to reflect the interaction of M with its oracle. We assume that M works in time $(size_k(\mathbf{I}))^s$ for some $s \geq 1$ and $\mathbf{I} \in \mathcal{B}_\sigma$.

First, we add to the prefix of φ_M the existential quantification $\exists^{(s+1)\cdot k} S^o_1 \ldots \exists^{(s+1)\cdot k} S^o_n$. Let r^o_i denote the arity of the distinguished oracle relation $R^o_i \in \sigma^o = \{R^o_1, \ldots, R^o_n\}$. For $1 \leq i \leq n$, the arity of the relation variable S^o_i is $r^o_i + s \cdot k$. The intended interpretation of $S^o_i(\bar{a}, \bar{t})$ is that at time $\bar{t}$, the distinguished oracle relation R^o_i in the rs contains the r^o_i-tuple $\bar{a}$.

The sub-formula ψ of φ_M treats the variables $S^o_1, \ldots, S^o_n$ corresponding to the distinguished oracle relations in the rs of M in exactly the same way as the variables $S_1, \ldots, S_l$ which correspond to the other relations in the rs of M. We only need to add a special case to the sub-formula of ψ which express that the relations T_i's, H_q's, S_i's and S^o_i's respect the transition function of M. When M is in the oracle query state $q_?$, $\chi(q_?, a, \alpha, b, q', m, \gamma, R)$ is the sentence describing the transition in which upon entering the query state $q_?$, the machine moves to state q_{YES} if the σ^o-structure held in the rs is in the oracle of M, or to state q_{NO} if it is not. W.l.o.g., we assume that the contents of the rs as well as of the working tape of M and the position of its read/write head remain unchanged.

The more "natural" way of expressing $\chi(q_?, a, \alpha, b, q', m, \gamma, R)$ is probably as follows:

$$\begin{aligned}\forall \bar{p} \forall \bar{t} \Big(H_{q_?}(\bar{p}, \bar{t}) \to \big(\hat{\alpha}_{q_o}(\bar{t}) \to H_{q_{YES}}(\bar{p}, \bar{t}+1)\big) \wedge \big(\neg\hat{\alpha}_{q_o}(\bar{t}) \to H_{q_{NO}}(\bar{p}, \bar{t}+1)\big) \wedge \\ \forall \bar{x} \big(\textstyle\bigwedge_{i \in \{0,1,\mathfrak{b}\}} T_i(\bar{x}, \bar{t}+1) \leftrightarrow T_i(\bar{x}, \bar{t})\big) \wedge \\ \textstyle\bigwedge_{0 \leq i \leq l} \big(\forall x_1 \ldots x_{r_i}(S_i(x_1, \ldots, x_{r_i}, \bar{t}) \leftrightarrow S_i(x_1, \ldots, x_{r_i}, \bar{t}+1))\big) \wedge \\ \textstyle\bigwedge_{0 \leq i \leq n} \big(\forall x_1 \ldots x_{r^o_i}(S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t}) \leftrightarrow S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t}+1))\big)\Big)\end{aligned}$$

where $\hat{\alpha}_{q_o}(\bar{t})$ is the formula obtained by replacing in α_{q_o} each atomic sub-formula of the form $R^o_i(y_1, \ldots, y_{r^o_i})$ $(0 \leq i \leq n)$ by $S^o_i(y_1, \ldots, y_{r^o_i}, \bar{t})$. But, we need the resulting formula to be in prenex normal form. Unfortunately, equivalences such as $\forall x \, Q \, \gamma(x) \leftrightarrow \forall X \, Q \, \big(\exists! x X(x) \rightarrow \forall x (X(x) \rightarrow \gamma(x))\big)$, where Q stands for an arbitrary sequence of first- and second-order quantifiers and $\exists! x X(x)$ means "there exists exactly one x such that $X(x)$", are no longer true for SO$^\omega$. Not all elements of the domain are distinguishable from each other in FO^k for a fixed k. Thus, it may well happen that there is an element a in the domain of a given structure $\mathbf{I}$ such that $\{a\}$ is not closed under under $\equiv^k$ on $\mathbf{I}$.

In order to write $\chi(q_?, a, \alpha, b, q', m, \gamma, R)$ in a form such that the SO$^\omega$ quantifiers in α_{q_o} can be moved to the prefix of φ_M, we use Lemma 7.23 . That is, we assume that in any computation M makes at most one query to its oracle. Under this assumption, we can then write $\chi(q_?, a, \alpha, b, q', m, \gamma, R)$ as the conjunction of:

$$\begin{aligned}
&\exists^{k_{21}} X_{21} \ldots \exists^{k_{2s_2}} X_{2s_2} \forall^{k_{31}} X_{31} \ldots \forall^{k_{3s_3}} X_{3s_3} \ldots Q^{k_{m1}} X_{m1} \ldots Q^{k_{ms_m}} X_{ms_m} \\
&\qquad \forall \bar{p} \forall \bar{t} \Big(H_{q_?}(\bar{p}, \bar{t}) \wedge \hat{\psi}_o(\bar{t}) \rightarrow \\
&\qquad\qquad H_{q_{YES}}(\bar{p}, \bar{t}+1) \wedge \forall \bar{x} \big(\textstyle\bigwedge_{i \in \{0,1,\mathfrak{b}\}} T_i(\bar{x}, \bar{t}+1) \leftrightarrow T_i(\bar{x}, \bar{t}) \big) \wedge \\
&\qquad\qquad \textstyle\bigwedge_{0 \leq i \leq l} \big(\forall x_1 \ldots x_{r_i} (S_i(x_1, \ldots, x_{r_i}, \bar{t}) \leftrightarrow S_i(x_1, \ldots, x_{r_i}, \bar{t}+1)) \big) \wedge \\
&\qquad\qquad \textstyle\bigwedge_{0 \leq i \leq n} \big(\forall x_1 \ldots x_{r^o_i} (S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t}) \leftrightarrow S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t}+1)) \big) \Big)
\end{aligned}$$

and

$$\begin{aligned}
&\forall^{k_{21}} X'_{21} \ldots \forall^{k_{2s_2}} X'_{2s_2} \exists^{k_{31}} X'_{31} \ldots \exists^{k_{3s_3}} X'_{3s_3} \ldots Q^{k_{m1}} X'_{m1} \ldots Q^{k_{ms_m}} X'_{ms_m} \\
&\qquad \forall \bar{p} \forall \bar{t} \Big(H_{q_?}(\bar{p}, \bar{t}) \wedge \neg \hat{\psi}'_o(\bar{t}) \rightarrow \\
&\qquad\qquad H_{q_{NO}}(\bar{p}, \bar{t}+1) \wedge \forall \bar{x} \big(\textstyle\bigwedge_{i \in \{0,1,\mathfrak{b}\}} T_i(\bar{x}, \bar{t}+1) \leftrightarrow T_i(\bar{x}, \bar{t}) \big) \wedge \\
&\qquad\qquad \textstyle\bigwedge_{0 \leq i \leq l} \big(\forall x_1 \ldots x_{r_i} (S_i(x_1, \ldots, x_{r_i}, \bar{t}) \leftrightarrow S_i(x_1, \ldots, x_{r_i}, \bar{t}+1)) \big) \wedge \\
&\qquad\qquad \textstyle\bigwedge_{0 \leq i \leq n} \big(\forall x_1 \ldots x_{r^o_i} (S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t}) \leftrightarrow S^o_i(x_1, \ldots, x_{r^o_i}, \bar{t}+1)) \big) \Big)
\end{aligned}$$

where $\hat{\psi}_o(\bar{t})$ is the formula obtained by replacing in ψ_o each atomic sub-formula of the form $R^o_i(y_1, \ldots, y_{r^o_i})$ $(0 \leq i \leq n)$ by $S^o_i(y_1, \ldots, y_{r^o_i}, \bar{t})$, and $\hat{\psi}'_o(\bar{t})$ is the formula obtained by replacing in $\hat{\psi}_o$ each occurrence of a relation variable $X_{ij} \in \{X_{21}, \ldots, X_{2s_2}, X_{31}, \ldots, X_{3s_3}, \ldots, X_{m1}, \ldots, X_{ms_m}\}$ by X'_{ij}. Note that for the sentence above, we use the fact that $\neg \alpha_{q_o}$ is equivalent to

$$\forall^{k_{21}} X_{21} \ldots \forall^{k_{2s_2}} X_{2s_2} \exists^{k_{31}} X_{31} \ldots \exists^{k_{3s_3}} X_{3s_3} \ldots Q^{k_{m1}} X_{m1} \ldots Q^{k_{ms_m}} X_{ms_m} (\neg \psi_o).$$

It is not difficult to see that the SO$^\omega$ quantifiers in the sentence above, can now be safely moved to the prefix of ψ_M and rearranged in such a way that the resulting formula is in $\Sigma^{1,\omega}_m$. $\square$

Chapter 8

Conclusions and Future Work

The central results in this dissertation prove the properness of two different hierarchies of arity and alternation inside each higher-order logic of order greater than or equal to three, namely the AA^i and HAA^i hierarchies. Both results were obtained by exploiting the underlying idea of expressing the relationship of satisfaction for logics restricted to finite models, by means of logics restricted to finite models. This idea was independently formulated by M. Mostowski [Mos93, Mos01, Mos03] and by Makowsky and Pnueli [MP96]. However, the actual approach followed in those works is not exactly the same. While in the work of Makowsky and Pnueli the formulae are encoded as finite structures, in the work of M. Mostowski the formulae are encoded as natural numbers over sufficiently large finite models which have a suitable amount of arithmetical structure so that Gödelization can be carried out. From the point of view of our work though, the two approaches can be used to prove similar, although not exactly the same, kinds of results.

To separate the different levels of these hierarchies, we extended Makowsky and Pnueli approach to higher-orders logics. This involved a substantial amount of technical work which included among other things to express the $AUTOSAT$ query in higher-order logics beyond second-order. As a by-product of this exercise, intuition on expressing actual queries in higher-order logics was gained. We think it would be useful to have further examples of queries other than $AUTOSAT$ that are inexpressible at certain levels of the AA^i and HAA^i hierarchies in order to obtain more practical relevance. This would require to extract the essence of the $AUTOSAT$ problem via reduction. Something along the line of the reductions from theories of the Boolean model to $AUTOSAT$ shown in Chapter 6.

We studied the related approach of M. Mostowski, which was later on extended by Kolodziejczyk [Kol04b, Kol05], as an alternative to prove the properness of hierarchies in higher-order logics. We have seen that this approach involving FM-truth definitions can be used to prove the properness of the HAA^i hierarchies; indeed we used it to give a sufficient condition for the HAA^1-hierarchy of second-order logic formulae to be strict. But on the other hand, we could not use this approach with the AA^i hierarchies since the finite version of Tarski's theorem on the undefinability of truth does not allow us to prove the lower bounds for them.

Also we explored possible existential proofs of hierarchy theorems for higher-order logics by means of a complexity-theoretical type of argument involving variants of the time-hierarchy theorem for alternating Turing machines with bounded alternation. A drawback of this approach is that, to the best of our knowledge, there is *no* written source for the required variants of the time-hierarchy theorem. Furthermore, we would only get existential proofs of essentially the same sort of results, instead of constructive proofs as in the work of Makowsky and Pnueli for second-order logic and in our work for higher-order logics, where specific queries which separate the different levels of the hierarchies were exhibited.

While the result concerning the properness of the AA^i hierarchies generalizes for higher-order logics a result of Makowsky and Pnueli [MP96] for second-order logic, the result concerning the properness of the HAA^i hierarchies applies to logics of order greater than or equal three and it is *not* known whether it holds for second-order logic. It was only under the very strong assumption that for every vocabulary σ there is a fixed k such that the data complexity for FO is in NTIME(n^k), that we were able to prove the properness of the HAA^1 hierarchy of second-order logic.

We used the same diagonalization argument to prove both the lower bounds for the HAA^i hierarchies and the lower bounds for the AA^i hierarchies. Furthermore, as shown by Proposition 5.2, this argument also works for the HAA^1 hierarchy of second-order logic formulae. Thus, what gave us the key to prove the properness of the HAA^i hierarchies for $i \geq 2$, was mainly the fact that we were able to define $AUTOSAT(HAA^i(r, m))$ in a slightly higher layer of the HAA^i hierarchy (see Proposition 5.3). By contrast, the question of whether it is possible to do the same for $AUTOSAT(HAA^1(r, m))$ is still open.

Open Question 8.1. Is there some $c_1, c_2 \in \mathbb{N}$ such that for every $r, m \geq 1$, it holds that $AUTOSAT(HAA^1(r, m)[\rho])$ is definable in $HAA^1(r + c_1, m + c_2)[\rho]$?

This finding gives us hope that analogous problems for higher-order logics (of order greater than or equal three) of open problems regarding hierarchies of formulae in second-order logic, could in some cases be more easily approachable in higher-order logics. Of course this does not seem to be the general case. In fact, the standard combinatorial techniques used to separate the expressive power of logics over finite models, seem to be rather too complicated to apply to logics beyond monadic second-order (see for instance the proof on the lower bounds in quantifier rank for the parity property in [KT07]).

We also studied the complexity of $AUTOSAT$ and showed that $AUTOSAT(\Sigma^1_m)$ is complete for the prenex fragments Σ^2_m of third-order logic. We worked the proof of this result with a great deal of detail. This allowed us to develop some strong conjectures on the complexity of $AUTOSAT$ for further fragments of higher-order logics. We present these conjectures in the next section.

Finally, inspired by a work of Dawar [Daw98], we explored in detail the connection between the concept of relational complexity and the restricted second-order logic SO^ω. The aim here was to provide the basis for a new line of research in the area of higher-order logics in finite models. We have of course many open questions for future research in this topic. We comment on some of these questions in the second section.

8.1 Some Conjectures Regarding the Complexity of $AUTOSAT$

We know from [MP96] that for the prenex fragments of first-order logic Σ^0_m, $AUTO$-$SAT(\Sigma^0_m)$ is complete for the corresponding class Σ^p_m of the polynomial-time hierarchy, or equivalently, complete for the corresponding fragment Σ^1_m of second-order logic. In Chapter 6 we extended that result proving that for the prenex fragments of second-order logic Σ^1_m, $AUTOSAT(\Sigma^1_m)$ is complete for the corresponding fragment Σ^2_m of third-order logic. We strongly believe that this result can also be extended to higher orders.

Conjecture 8.2. For $i \geq 2$ and $m \geq 1$, $AUTOSAT(\Sigma^i_m)$ is complete for Σ^{i+1}_m under polynomial-time reductions.

We think that the same strategy that we used to prove Theorem 6.2, could also be used to prove this conjecture. Let us take for instance $AUTOSAT$ for the prenex fragments Σ^2_m of third-order logic. We could extend Definition 6.5 with the following cases:

- If φ has the form $\mathcal{X}(X_1, \ldots, X_r)$, where $\mathcal{X}$ is a third-order variable of arity r and $X_1, \ldots, X_r$ are second-order variables of arity r, then $\varphi'(w_0, w_1)$ is
  $\mathcal{X}(X_1, \ldots, X_r) \bigwedge_{1 \leq i \leq r} \forall x_1 \ldots x_r (X_i(x_1, \ldots, x_r) \rightarrow$
  $((x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)))$.
- If φ has the form $\exists \mathcal{X}(\psi)$, where $\mathcal{X}$ is a third-order variable of arity r, then $\varphi'(w_0, w_1)$ is
  $\exists \mathcal{X}\big(\forall X_1 \ldots X_r \big(\mathcal{X}(X_1, \ldots, X_r) \rightarrow$
  $\bigwedge_{1 \leq i \leq r} \forall x_1 \ldots x_r (X_i(x_1, \ldots, x_r) \rightarrow$
  $((x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)))\big) \wedge$
  $\psi'(w_0, w_1)\big)$.
- If φ has the form $\forall \mathcal{X}(\psi)$, where $\mathcal{X}$ is a third-order variable of arity r, then $\varphi'(w_0, w_1)$ is
  $\forall \mathcal{X}\big(\forall X_1 \ldots X_r \big(\mathcal{X}(X_1, \ldots, X_r) \rightarrow$
  $\bigwedge_{1 \leq i \leq r} \forall x_1 \ldots x_r (X_i(x_1, \ldots, x_r) \rightarrow$
  $((x_1 = w_0 \vee x_1 = w_1) \wedge \ldots \wedge (x_r = w_0 \vee x_r = w_1)))\big) \rightarrow$
  $\psi'(w_0, w_1)\big)$.

Note that by the definition of higher-order logic used in [HT05, HT06a] where Theorem 6.4 was proved, we do not need to consider third-order variables of a third-order type other than the types in $\{\underbrace{(\iota^r, \ldots, \iota^r)}_{r} : r \geq 1\}$, since these are the only kind of third-order variables allowed in the formulae in Σ^2_m in those papers.

We could also extend Definition 6.6 with the following case:

- If $\mathcal{X}$ is a third-order variable of arity r, then v_A assigns to $\mathcal{X}$ an r-ary third-order relation $\mathcal{R} \subseteq (\mathcal{P}(\{min(\mathbf{A}), max(\mathbf{A})\}^r))^r$ such that $(A_1, \ldots, A_r) \in \mathcal{R}$ iff $(g(A_1), \ldots, g(A_r)) \in v(\mathcal{X})$. Here $g(A_i) = \{(b_1, \ldots b_r) :$ there is a $(a_1, \ldots, a_r) \in A_i$ such that, for $1 \leq i \leq r$, $b_i = 0^{\mathbf{B}}$ if $a_i = min(\mathbf{A})$ and $b_i = 1^{\mathbf{B}}$ if $a_i = max(\mathbf{A})\}$.

Under these extended definitions, it seems to us that it would be a routine task to prove analogous results to Lemma 6.7 and 6.8 for the fragments Σ^2_m of third-order logic and to show that $AUTOSAT(\Sigma^2_m)$ is hard for the fragment Σ^3_m of fourth-order logic.

As to a possible argument to prove that $AUTOSAT(\Sigma^2_m)$ is in Σ^3_m, we also think that it should be a routine task to extend the proof of Proposition 6.10, since by Theorem 3.8, Σ^3_m captures $\bigcup_{c \in \mathbb{N}} \mathrm{NTIME}(2^{2^{n^c}})^{\Sigma^p_{m-1}}$, and the nondeterministic time needed to guess a third-order relation of arity r is $\mathcal{O}(2^{n^r})$.

On the other hand, if we bound the arity of the second-order variables in the $\Sigma^i_m \cup \Pi^i_m$ fragments, we know from [MP96] that $AUTOSAT(HAA^1(r, m))$ is complete for PSPACE. Our conjecture to this regard is that this result can also be generalized to higher orders.

Conjecture 8.3. For every $i \geq 2$ and every $r, m \geq 1$, $AUTOSAT(HAA^i(r, m))$ is complete for $\mathrm{DSPACE}(\exp_{i-1}(n^c))$.

To the best of our knowledge, there is no known complete problem for the class $\mathrm{DSPACE}(\exp_i(n^c))$ when $i \geq 1$. However, we think it is quite possible that in the same way as QSAT (quantified satisfiability) is complete for PSPACE, the union for every j of the Σ^i_j theory of the Boolean model ($i \geq 1$), i.e., $\bigcup_{j \geq 1} \Sigma^i_j{-}\mathrm{Th}(\mathbf{B})$, could be complete for $\mathrm{DSPACE}(\exp_i(n^c))$. If that is the case, then it should be relatively straightforward to reduce $\bigcup_{j \geq 1} \Sigma^i_j{-}\mathrm{Th}(\mathbf{B})$ to $AUTOSAT(HAA^{i+1}(r, m))$ for every $r, m \geq 1$. Also, space bounded by $\mathcal{O}(\exp_i(n^r))$ should be enough to decide $AUTOSAT(HAA^{i+1}(r, m))$ since the maximal-arity of the variables in $HAA^{i+1}(r, m)$ is restricted by r.

8.2 Some Considerations on Relational Complexity and Restricted Higher-Order Logics

First we consider a technical problem regarding the definition of SO^ω. Afterwards we explore two possible directions for research in the line of Chapter 7. Let C^k denote the logic which extends FO^k with *counting quantifiers*. The first direction consists in considering second-order quantifiers which range over relations that are closed under equivalence of C^k-types of tuples, for some k, instead of closed under equivalence of FO^k-type of tuples. The second direction consists in using the idea behind SO^ω to define restricted versions of higher-order logics of order higher than two.

Following [Daw98], we defined in Chapter 7 the set of formulae of SO^{ω} as $\mathrm{FO} \cup \bigcup_{m \geq 1} \Sigma_m^{1,\omega}$. Now, suppose that we define the syntax and semantics of this logic as below, i.e., in a more usual fashion.

Definition 8.4. In addition to the symbols of first-order logic, the alphabet of SO'^{ω} contains, for each $k \geq 1$, a pair of *second-order quantifiers* $\exists^k$ and $\forall^k$, and countably many k-ary *relation variables* $V_1^k, V_2^k, \ldots$ To denote relation variables we use letters $X, Y, \ldots$ Let σ be a relational vocabulary, we define the set of *SO'^{ω}* formulae over σ to be the set generated by the rules for first-order formulae with equality extended by the following rules:

- If X is r-ary and $x_1, \ldots, x_r$ are individual variables, then $X(x_1, \ldots, x_r)$ is an atomic formula.
- If φ is a wff and X is a relation variable of arity $r \leq k$, then $\exists^k X(\varphi)$ is a wff.

Let $\mathbf{I}$ be a σ-structure, and let *val* be a valuation on $\mathbf{I}$. The notion of *satisfaction* in SO'^{ω} extends the notion of satisfaction in first-order with the following rules:

- $\mathbf{I}, \mathit{val} \models X(x_1, \ldots, x_r)$ iff $(\mathit{val}(x_1), \ldots, \mathit{val}(x_r)) \in \mathit{val}(X)$.
- $\mathbf{I}, \mathit{val} \models \exists^k X(\varphi)$ where X is a relation variable of arity $r \leq k$ and φ is a wff, iff there is a valuation val', which is X-equivalent to val, such that $\mathit{val}'(X)$ is closed under the equivalence relation $\equiv^k$ in $\mathbf{I}$, and $\mathbf{I}, \mathit{val}' \models \varphi$.

Let SO''^{ω} be the restriction of SO'^{ω} to formulae without free relation (second-order) variables. We do not know whether or not SO''^{ω} and SO^{ω} are equivalent. Note that, the only formulae of SO''^{ω} which are allowed in Definition 7.1 of SO^{ω}, are those formulae which are in GSNF. As explained in part (b) of the proof of Theorem 7.24, equivalences such as $\forall x\, Q\, \gamma(x) \leftrightarrow \forall X\, Q\, \big(\exists! x X(x) \rightarrow \forall x (X(x) \rightarrow \gamma(x))\big)$, where Q stands for an arbitrary sequence of first- and second-order quantifiers and $\exists! x X(x)$ means "there exists exactly one x such that $X(x)$", are no longer true for SO''^{ω}. In the general case not all elements of the domain are definable in FO^k for a fixed k. Thus, it may well happen that there is an element a in the domain of a given structure $\mathbf{I}$ such that $\{a\}$ is not closed under under $\equiv^k$ on $\mathbf{I}$. Therefore, we think that the following is still open.

Open Question 8.5. Is every SO''^{ω}-formula equivalent to a $\Sigma_m^{1,\omega}$-formula for some m?

Next, we comment on $\mathrm{SO}^{\mathrm{C}^\omega}$, the logic similar to SO^ω that is obtained by considering C^k-types of tuples instead of FO^k-types of tuples. C^k is a well known logic in finite model theory which is obtained by adding to FO^k *counting quantifiers*, i.e., all existential quantifiers of the form $\exists^{\geq n}$ with $n \geq 1$. Informally, $\exists^{\geq n} x(\varphi)$ means that there are at least n *different* elements in the domain of the structure which satisfy φ.

For $k \geq r \geq 1$, we denote by $\equiv^{\mathrm{C}^k}$ the *equivalence relation* induced in the set of r-tuples over a given structure $\mathbf{I}$, by the equality of C^k-types of r-tuples.

We define the syntax of $\Sigma^{1,\mathrm{C}^\omega}_m$, $\Pi^{1,\mathrm{C}^\omega}_m$ and $\mathrm{SO}^{\mathrm{C}^\omega}$ in exactly the same way as the syntax of $\Sigma^{1,\omega}_m$, $\Pi^{1,\omega}_m$ and SO^ω, respectively (see Definition 7.1). As to the semantics, the only difference is that now the second-order quantifiers range over relations which are closed under the equivalence relation $\equiv^{\mathrm{C}^k}$ for some k, instead of the equivalence relation $\equiv^k$. That is, the notion of *satisfaction* in $\mathrm{SO}^{\mathrm{C}^\omega}$ extends the notion of satisfaction in first-order with the following rule:

- $\mathbf{I} \models \exists^k X(\varphi)$ where $k \geq 1$, X is a relation variable of arity $r \leq k$, φ is a wff of vocabulary $\sigma \cup \{X\}$ and $\mathbf{I}$ is a σ-structure, iff there is an $R \subseteq I^r$ such that R is closed under the equivalence relation $\equiv^{\mathrm{C}^k}$ in $\mathbf{I}$, and $(\mathbf{I}, R) \models \varphi$.

Considering the known results on C^k-types and FO^k-types, it seems clear that $\mathrm{SO}^{\mathrm{C}^\omega} \supset \mathrm{SO}^\omega$. However, as explained next, we think that it would be worth to carry on a more detailed study of the expressive power of $\mathrm{SO}^{\mathrm{C}^\omega}$. The necessary background material for this study, specifically for C^k-types and how they compare to FO^k-types, can be obtained from [CFI92, Gro98, Ott97], among others. Also [Tur06] is a relevant source to look at on this regard.

It is well known that in the absence of linear order many natural NP-complete problems as for instance Hamiltonicity and clique, are not definable in $\mathcal{L}^\omega_{\infty\omega}$. Since $\mathrm{SO}^\omega \subseteq \mathrm{PFP}$ and $\mathrm{PFP} \subset \mathcal{L}^\omega_{\infty\omega}$, nor are those problems definable in SO^ω. On the other hand, it is easy to see that there are NP-complete problems that can be expressed in $\Sigma^{1,\omega}_1$ since this logic is equivalent to Σ^1_1 on ordered structures and thus it captures NP over those structures. However, without assuming an ordered domain, there are also natural NP-complete problems that *are* expressible in $\Sigma^{1,\omega}_1$. As shown by Dawar [Daw98], the problem of inequivalence of nondeterministic finite automata (NFA) on a unary alphabet, which is known to be an NP-complete problem, is definable in $\Sigma^{1,\omega}_1$. The same is also true for the restriction of NFA inequivalence to a finite language.

Open Question 8.6. Continuing the line of the work of Dawar described above, we think it would be interesting to explore natural problems which are expressible, and natural problems which are not, in different fragments of $\mathrm{SO}^{\mathrm{C}^\omega}$. Let $\mathrm{C}^\omega_{\infty\omega}$ denote the infinitary logic with counting which is defined in the same way as $\mathcal{L}^\omega_{\infty\omega}$ with the addition of all counting quantifiers. If as we expect the class of problems expressible in $\mathrm{SO}^{\mathrm{C}^\omega}$ is strictly included in the class of problems expressible in $\mathrm{C}^\omega_{\infty\omega}$, then queries whose classification in the infinitary logics $\mathrm{C}^\omega_{\infty\omega}$ and $\mathcal{L}^\omega_{\infty\omega}$ is well known could serve as a good starting point for this research. Some well known examples of such queries are: the property of a graph having even cardinality which is in $\mathrm{C}^1_{\infty\omega}$ and is not in $\mathcal{L}^\omega_{\infty\omega}$; the property of a graph being regular as well as the property of a graph being Eulerian which are in $\mathrm{C}^2_{\infty\omega}$ and are not in $\mathcal{L}^\omega_{\infty\omega}$ ([Ott97]); the property of a graph being connected which is in $\mathcal{L}^3_{\infty\omega}$ and is not in $\mathrm{C}^2_{\infty\omega}$ ([Gro98]); and the property of a graph having an even number of connected components which is in $\mathrm{C}^\omega_{\infty\omega}$ and is not in $\mathcal{L}^\omega_{\infty\omega}$ ([KV95]).

In [Ott96], Otto defined a generalization of the relational machine of Abiteboul and Vianu that incorporates counting operations in a generic manner. He called this model of computation *relational machine with counting*, and used it to characterize the expressive power of fixed-point logics with *counting terms*. It seems to us that the relational complexity classes defined in terms of this model of computation could be used to characterize the expressive power of $\mathrm{SO}^{\mathrm{C}^\omega}$, in much the same way as the original relational complexity classes were used to characterize the expressive power of SO^ω.

Open Question 8.7. Let the C^k-size of a structure $\mathbf{I}$ be the number of $\equiv^{\mathrm{C}^k}$-classes of k-tuples over $\mathbf{I}$. Suppose we define the relational complexity classes of Section 2.5 using relational machines with counting instead of the original relational machines, and furthermore, using the C^k-size instead of the k-size as a basis for measuring the complexity. Does $\mathrm{SO}^{\mathrm{C}^\omega}$ capture the resulting "counting relational" polynomial-time hierarchy, i.e., the polynomial-time hierarchy redefined using relational machines with counting? Which is the relationship between the standard complexity classes and the resulting relational complexity classes defined in terms of relational machines with counting?

Finally, we discuss the extension of the correspondence between the *relational* polynomial hierarchy and the prenex fragments $\Sigma^{1,\omega}_m$ of SO^ω, to higher orders. We do that for third-order logic to simplify the exposition, since it is easy to extend the definitions for orders higher than three.

First we define what is a third-order relation closed under $\equiv^k$. For clarity we only consider third-order relations whose type has the form $\underbrace{(\iota^r, \ldots, \iota^r)}_{r}$ for some $r \geq 1$.

Definition 8.8. A relation $\mathcal{R}$ of *order* 3 and arity $r \leq k$ is *closed under* $\equiv^k$ iff, every relation R_i in a tuple $(R_1, \ldots, R_r) \in \mathcal{R}$, is closed under $\equiv^k$.

Next we define the syntax and semantics of the corresponding restricted third-order logic.

Definition 8.9. We denote by $\Sigma^{2,\omega}_m[\sigma]$ the class of formulae of the form

$\exists^{k^1_1}\mathcal{X}_{11} \ldots \exists^{k^1_{s_1}}\mathcal{X}_{1s_1}\forall^{k^2_1}\mathcal{X}_{21} \ldots \forall^{k^2_{s_2}}\mathcal{X}_{2s_2} \ldots Q^{k^m_1}\mathcal{X}_{m1} \ldots Q^{k^m_{s_m}}\mathcal{X}_{ms_m}(\varphi)$,

where $arity(\mathcal{X}_{ij}) \leq k^i_j$, Q is either $\exists$ or $\forall$, and φ is an SO^ω formula of vocabulary $\sigma \cup \{\mathcal{X}_{11}, \ldots, \mathcal{X}_{1s_1}, \mathcal{X}_{21}, \ldots, \mathcal{X}_{2s_2}, \ldots, \mathcal{X}_{m1}, \ldots, \mathcal{X}_{ms_m}\}$.

As usual, $\Pi^{2,\omega}_m[\sigma]$ is defined dually.

$\mathrm{TO}^\omega = \bigcup_{m \geq 1} \Sigma^{2,\omega}_m$.

The notion of *satisfaction* in TO^ω extends the notion of satisfaction in SO^ω with the following rule:

$\mathbf{I} \models \exists^k \mathcal{X}(\varphi)$, where $\mathcal{X}$ is third-order variable of arity r, iff there is an $\mathcal{R}^{\mathbf{I}} \subseteq \big(\mathcal{P}(I^r)\big)^r$ closed under $\equiv^k$ such that $(\mathbf{I}, \mathcal{R}^{\mathbf{I}}) \models \varphi$.

Now, suppose we want to characterize the expressive power of the prenex fragments $\Sigma^{2,\omega}_m$ of TO^ω in terms of relational complexity classes. It seems to us that we could do that either by using the same model of relational machine that we have been using till now, or by extending the model by allowing it to store third-order relations in its *rs*. In the first case we think that we would need to encode a canonical representation of the structure in the Turing tape of the relational machine by using Abiteboul and Vianu normal form for relational machines (see [AV95]), and then to work on the tape. Consequently, we would need to consider a relational machine with a classical oracle consisting in a set of strings, rather than a relational machine with a relational oracle.

If we choose to extend the relational machine model of computation by adding third-order relations to the *rs*, then the strategy could be closer to the strategy that we used in Chapter 7 to characterize the expressive power of the prenex fragments $\Sigma^{1,\omega}_m$ of SO^ω. This strategy would require, among other things, to extend the first-order language used by the relational machine to communicate with the rs so that the machine can interact with the third-order relations in its rs. Also, we would need to extend the concept of relational oracle language to include structures with third-order relations, and thus to adapt the concept of k-size of the input structures accordingly.

These considerations also apply to the restricted higher-order logics beyond second-order which would be obtained by considering higher-order relations closed under $\equiv^{\mathrm{C}^k}$ instead of $\equiv^k$.

We would like to conclude this section with a last observation. Let us assume that formulae of SO^ω with free second-order variables are allowed. Let $\mathrm{SO}^{\omega,k}$ be the fragment of SO^ω where only formulae with up to k different variables (counting all fist-order and second-order variables) are permitted. Let $\mathbf{I}$ be a relational structure of vocabulary σ, and let $\bar{A} = (A_1, \ldots, A_r)$ be an r-tuple in $(\mathcal{P}(I^r))^r$. We define the $\mathrm{SO}^{\omega,k}$-*type* of $\bar{A}$ in $\mathbf{I}$ as follows:

$$tp_{\mathbf{I}}^{\mathrm{SO}^{\omega,k}}(\bar{A}) = \{\varphi \in \mathrm{SO}^{\omega,k}[\sigma] : \mathit{free}(\varphi) \subseteq \{X_1, \ldots, X_r\} \text{ and } \mathbf{I} \models \varphi[A_1, \ldots, A_r]\}$$

Let $k \geq r \geq 1$ and let $\bar{A}$ and $\bar{B}$ be a pair of r-tuples in $(\mathcal{P}(I^r))^r$. We say that $\bar{A} \equiv^{\mathrm{SO}^{\omega,k}} \bar{B}$ iff $tp_{\mathbf{I}}^{\mathrm{SO}^{\omega,k}}(\bar{A}) = tp_{\mathbf{I}}^{\mathrm{SO}^{\omega,k}}(\bar{B})$. Consequently, a third-order relation $\mathcal{R}$ of arity $r \leq k$ is *closed* under the equivalence relation $\equiv^{\mathrm{SO}^{\omega,k}}$ iff, for every pair of r-tuples $\bar{A}$ and $\bar{B}$ in $(\mathcal{P}(I^r))^r$, if $\bar{A} \in \mathcal{R}$ and $\bar{A} \equiv^{\mathrm{SO}^{\omega,k}} \bar{B}$, then $\bar{B} \in \mathcal{R}$.

Then an alternative generalization of the idea behind SO^ω to higher-order logics could be obtained by defining the notion of *satisfaction* in TO^ω as the extension of the notion of satisfaction in SO^ω with the following rule:

$\mathbf{I} \models \exists^k \mathcal{X}(\varphi)$, where $\mathcal{X}$ is a third-order variable of arity r, iff there is an $\mathcal{R}^{\mathbf{I}} \subseteq \big(\mathcal{P}(I^r)\big)^r$ closed under $\equiv^{\mathrm{SO}^{\omega,k}}$ such that $(\mathbf{I}, \mathcal{R}^{\mathbf{I}}) \models \varphi$.

Bibliography

[AF90] Miklós Ajtai and Ronald Fagin. Reachability is harder for directed than for undirected finite graphs. *J. Symb. Log.*, 55(1):113–150, 1990.

[AHV94] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, Redwood City, CA, 1994.

[Ajt83] Miklós Ajtai. Σ^1_1 formulae on finite structures. *Ann. Pure Appl. Logic*, 24:1–48, 1983.

[AU79] Alfred V. Aho and Jeffrey D. Ullman. Universality of data retrieval languages. In *Proceedings of the Sixth Annual ACM Symposium of Principles of Programming Languages*, pages 110–120, New York, NY, USA, 1979. ACM Press.

[AV91a] Serge Abiteboul and Victor Vianu. Datalog extensions for database queries and updates. *J. Comput. System Sci.*, 43:62–124, 1991.

[AV91b] Serge Abiteboul and Victor Vianu. Generic computation and its complexity. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, pages 209–219, New York, NY, USA, 1991. ACM Press.

[AV95] Serge Abiteboul and Victor Vianu. Computing with first-order logic. *J. Comput. Syst. Sci.*, 50(2):309–335, 1995.

[AVV95] Serge Abiteboul, Moshe Y. Vardi, and Victor Vianu. Computing with infinitary logic. *Theor. Comput. Sci.*, 149(1):101–128, 1995.

[AVV97] Serge Abiteboul, Moshe Y. Vardi, and Victor Vianu. Fixpoint logics, relational machines, and computational complexity. *J. ACM*, 44(1):30–56, 1997.

[Bar77] Jon Barwise. On moschovakis closure ordinals. *J. Symb. Log.*, 42(2):292–296, 1977.

[BDG90] José Luis Balcázar, Joseph Díaz, and Joaquim Gabarró. *Structural Complexity II.* Texts in Theoretical Computer Science, EATCS. Springer, Berlin Heidelberg New York, 1990.

[BDG95] José Luis Balcázar, Joseph Díaz, and Joaquim Gabarró. *Structural Complexity I.* Texts in Theoretical Computer Science, EATCS. Springer, Berlin Heidelberg New York, 2 edition, 1995.

[Ben62] J. H. Bennett. *On Spectra.* PhD thesis, Princeton University, Princeton, NJ, 1962.

[BL76] Kellogg S. Booth and George S. Leuker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[Bör84] Egon Börger. Decision problems in predicate logic. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium*, volume 112 of *Studies in Logic and the Foundations of Mathematics*, pages 263–301. North Holland, 1984.

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik und Grund. Math*, 6:66–92, 1960.

[CFI92] Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

[CH80] Ashok K. Chandra and David Harel. Computable queries for relational data bases. *J. Comput. Syst. Sci.*, 21(2):156–178, 1980.

[Chr74] C. A. Christen. *Spektren und Klassen Elementarer Funktionen.* PhD thesis, ETH Zürich, 1974.

[Cod70] Edgar F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, USA, 1971. ACM Press.

[Coo72] Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192, Denver, Colorado, USA, May 1972. ACM.

[Daw93] Anuj Dawar. *Feasible Computation Through Model Theory.* PhD thesis, University of Pennsylvania, Philadelphia, 1993.

[Daw98] Anuj Dawar. A restricted second order logic for finite structures. *Inf. Comput.*, 143(2):154–174, 1998.

[DLW95] Anuj Dawar, Steven Lindell, and Scott Weinstein. Infinitary logic and inductive definability over finite structures. *Inf. Comput.*, 119(2):160–175, 1995.

[Ebb85] Heinz-Dieter Ebbinghaus. Extended logics: The general framework. In J. Barwise and S. Fefferman, editors, *Model Theoretic Logics*, pages 25–76. Springer-Verlag, 1985.

[EF99] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory.* Perspectives in Mathematical Logic. Springer, Berlin Heidelberg New York, 2nd edition, 1999.

[EGG96] Thomas Either, Georg Gottlob, and Yuri Gurevich. Normal forms for second-order logic over finite structures, and classification of NP optimization problems. *Ann. Pure Appl. Logic*, 78:111–125, 1996.

[Elg61] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 21:89–96, 1961.

[Fag74] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity of Computations*, volume 7 of *SIAM-AMS Proc.*, pages 27–41, Providence, RI, 1974. American Mathematical Society.

[Fag75] Ronald Fagin. A spectrum hierarchy. *Z. Math. Log. Grundl. Math.*, 21:123–134, 1975.

[Fag94] Ronald Fagin. Problem proposed in Oberwolfach meeting, 1994. Collection of Open Problems in Finite Model Theory. Available at http://www-mgi.informatik.rwth-aachen.de/FMT/problems.pdf.

[FPT05] Flavio Antonio Ferrarotti, Jorge A. Peri, and José María Turull Torres. Expresando propiedades sobre grafos en lógicas de orden superior. In Jorge E. Sagula, editor, *Proceedings of the 7th Symposium on Education in Mathematics*, Chivilcoy, Bs. As., Argentina, May 2005. EDUMAT. In CD-ROM.

[FT04] Flavio Antonio Ferrarotti and José María Turull Torres. Using higher order quantification in logical query languages. In *Proceedings of the 3rd Chilean Database Workshop, XII Jornadas Chilenas de Computación*, Arica, Chile, November 2004. University of Tarapacá and Chilean Society of Computer Sciences. In CD-ROM.

[FT05] Flavio Antonio Ferrarotti and Jose Maria Turull Torres. Arity and alternation of quantifiers in higher order logics. Technical Report 10/2005, Department of Information Systems, Massey University, New Zealand, September 2005.

[FT06] Flavio Antonio Ferrarotti and José María Turull Torres. Arity and alternation: A proper hierarchy in higher order logics. In Jürgen Dix and Stephen J. Hegner, editors, *Proceedings of the 4th International Symposium on Foundations of Information and Knowledge Systems*, volume 3861 of *Lec. Notes Comput. Sci.*, pages 92–115. Springer, February 2006.

[FT07] Flavio Antonio Ferrarotti and José María Turull Torres. Arity and alternation: A proper hierarchy in higher order logics. *Ann. Math. Artif. Intell.*, 50(1–2):111–141, 2007.

[GH64] Paul C. Gilmore and Alan J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

[GH96] Martin Grohe and Lauri Hella. A double arity hierarchy theorem for transitive closure logic. *Arch. Math. Log.*, 35:157–171, 1996.

[Gro93] Martin Grohe. Bounded-arity hierarchies in fixed-point logics. In Egon Börger, Yuri Gurevich, and Karl Meinke, editors, *Proceedings of the 7th*

Workshop on Computer Science Logic, volume 832 of *Lec. Notes Comput. Sci.*, pages 150–164, Swansea, United Kingdom, September 1993. Springer.

[Gro96] Martin Grohe. Arity hierarchies. *Ann. Pure Appl. Logic*, 82(2):103–163, 1996.

[Gro98] Martin Grohe. Finite variable logics in descriptive complexity theory. *Bulletin of Symbolic Logic*, 4(4):345–398, 1998.

[GS86] Yuri Gurevich and Saharon Shelah. Fixed-point extensions of first-order logic. *Ann. Pure Appl. Logic*, 32:265–280, 1986.

[Hel89] Lauri Hella. Definability hierarchies of generalized quantifiers. *Ann. Pure Appl. Logic*, 43(3):235–271, 1989.

[Hel92] Lauri Hella. Logical hierarchies in ptime. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science*, pages 360–368, Santa Cruz, California, USA, June 1992. IEEE Computer Society.

[HS91] Richard Hull and Jianwen Su. On the expressive power of database queries with intermediate types. *J. Comput. Syst. Sci.*, 43(1):219–267, 1991.

[HT03] Lauri Hella and José María Turull Torres. Expressibility of higher order logics. *Electr. Notes Theor. Comput. Sci.*, 84, 2003.

[HT05] Lauri Hella and Jose Maria Turull Torres. Complete problems for higher order logics. Technical Report 12/2005, Department of Information Systems, Massey University, New Zealand, September 2005.

[HT06a] Lauri Hella and José María Turull Torres. Complete problems for higher order logics. In *Proceedings of the 15th Annual Conference of the EACSL (and 20th International Workshop) on Computer Science Logic*, volume 4207 of *Lec. Notes Comput. Sci.*, pages 380–394, Szeged, Hungary, September 2006. Springer.

[HT06b] Lauri Hella and José María Turull Torres. Computing queries with higher-order logics. *Theor. Comput. Sci.*, 355(2):197–214, 2006.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Series in Computer Sciences. Addison-Wesley, Redwood City, CA, 1979.

[Imm82] Neil Immerman. Upper and lower bounds for first-order expressibility. *J. Comput. System Sci.*, 25:76–98, 1982.

[Imm86] Neil Immerman. Relational queries computable in polynomial time. *Inform. and Control*, 68:86–104, 1986.

[Imm99] Neil Immerman. *Descriptive Complexity.* Graduate Texts in Computer Science. Springer, Berlin Heidelberg New York, 1999.

[Kol04a] Leszek Aleksander Kolodziejczyk. A finite model-theoretical proof of a property of bounded query classes within PH. *J. Symb. Log.*, 69(4):1105–1116, 2004.

[Kol04b] Leszek Aleksander Kolodziejczyk. Truth definitions in finite models. *J. Symb. Log.*, 69(1):183–200, 2004.

[Kol05] Leszek Aleksander Kolodziejczyk. *Truth Definitions and Higher Order Logics in Finite Models.* PhD thesis, Institute of Philosophy, Warsaw University, Warsaw, Poland, February 2005.

[Kon06] Juha Kontinen. The hierarchy theorem for second-order generalized quantifiers. *J. Symb. Log.*, 71(1):188–202, 2006.

[KT07] Michal Krynicki and Jose Maria Turull Torres. Games on trees and syntactical complexity of formulas. *Logic Journal of the IGPL*, 15(5-6):653–687, 2007.

[KV88] Gabriel M. Kuper and Moshe Y. Vardi. On the complexity of queries in the logical data model. In Marc Gyssens, Jan Paredaens, and Dirk Van Gucht, editors, *Proceedings of the 2nd International Conference on Database Theory*, volume 326 of *Lec. Notes Comput. Sci.*, pages 267–280. Springer, 1988.

[KV92a] Phokion G. Kolaitis and Moshe Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science*, pages 46–57, Santa Cruz, California, USA, 1992. IEEE Computer Society.

[KV92b] Phokion G. Kolaitis and Moshe Y. Vardi. Infinitary logics and 0-1 laws. *Inform. and Comput.*, 98(2):258–294, 1992.

[KV95] Phokion G. Kolaitis and Jouko A. Väänänen. Generalized quantifiers and pebble games on finite structures. *Ann. Pure Appl. Logic*, 74(1):23–75, 1995.

[Lei89] Daniel Leivant. Descriptive characterizations of computational complexity. *J. Comput. Syst. Sci.*, 39(1):51–83, 1989.

[Lei94] Daniel Leivant. Higher order logic. In Dov M. Gabbay, Christopher J. Hogger, J. A. Robinson, and Jörg H. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2, pages 229–322. Oxford University Press, 1994.

[Lib04] Leonid Libkin. *Elements Of Finite Model Theory.* Texts in Theoretical Computer Science, EATCS. Springer, Berlin Heidelberg New York, 2004.

[Mat98] Oliver Matz. One quantifier will do in existential monadic second-order logic over pictures. In Lubos Brim, Jozef Gruska, and Jirí Zlatuska, editors, *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, volume 1450 of *Lec. Notes Comput. Sci.*, pages 751–759, Brno, Czech Republic, August 1998. Springer.

[MM03] Johann A. Makowsky and Julian Mariño. Tree-width and the monadic quantifier hierarchy. *Theor. Comput. Sci.*, 1(303):157–170, 2003.

[Mos93] Marcin Mostowski. Truth-definitions in finite models, 1993. Manuscript.

[Mos01] Marcin Mostowski. On representing concepts in finite models. *Math. Log. Q.*, 47(4):513–523, 2001.

[Mos03] Marcin Mostowski. On representing semantics in finite models. In A. Rojszczak, J. Cachro, and G. Kurczewski, editors, *Philosophical Dimensions of Logic and Science: Selected Contributed Papers from the 11th International Congress of Logic, Methodology, and Philosophy of Science*, pages 15–28, Krakow, 2003. Kluwer.

[MP96] Johann A. Makowsky and Yachin B. Pnueli. Arity and alternation in second-order logic. *Ann. Pure Appl. Logic*, 78(1-3):189–202, 1996.

[MST02] Oliver Matz, Nicole Schweikardt, and Wolfgang Thomas. The monadic quantifier alternation hierarchy over grids and graphs. *Inf. Comput.*, 179(2):356–383, 2002.

[MT97] Oliver Matz and Wolfgang Thomas. The monadic quantifier alternation hierarchy over graphs is infinite. In *Proc. 12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 236–244, Warsaw, Poland, 1997.

[Ott95] Martin Otto. An note on the number of monadic quantifiers in monadic Σ^1_1. *Inf. Process. Lett.*, 53(6):337–339, 1995.

[Ott96] Martin Otto. The expressive power of fixed-point logic with counting. *J. Symb. Log.*, 61(1):147–176, 1996.

[Ott97] Martin Otto. *Bounded variable logics and counting – A study in finite models*, volume 9. Springer, Berlin Heidelberg New York, 1997.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, MA, 1994.

[Poi82] Bruno Poizat. Deux ou trois choses que je sais de l_n. *J. Symb. Log.*, 47(3):641–658, 1982.

[Ros84] H.E. Rose. *Subrecursion: Functions and Hierarchies.* Oxford University Press, New York, USA, 1984.

[Sch94] Thomas Schwentick. Graph connectivity and monadic np. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 614–622, Santa Fe, New Mexico, USA, November 1994. IEEE.

[Sch97] Nicole Schweikardt. The monadic quantifier alternation hierarchy over grids and pictures. In Mogens Nielsen and Wolfgang Thomas, editors, *Proceedings of Computer Science Logic, 11th International Workshop*, volume 1414 of *Lec. Notes Comput. Sci.*, pages 441–460, Aarhus, Denmark, August 1997. Springer.

[SFM78] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978.

[Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[Tar33] Alfred Tarski. Pojęcie prawdy w językach nauk dedukcyjnch. *Prace Towarzystwa Naukowego Warzawskiego*, 1933. English translation in [Tar56]: The Concept of Truth in Formalized Languages.

[Tar56] Alfred Tarski. *Logic, Semantics, and Metamathematics.* Clarendon Press, Oxford, 1956.

[Tho82] Wolfgang Thomas. Classifying regular events in symbolic logic. *J. Comput. Syst. Sci.*, 25(3):360–376, 1982.

[Tur01a] José María Turull Torres. On the expressibility and the computability of untyped queries. *Ann. Pure Appl. Logic*, 108(1-3):345–371, 2001.

[Tur01b] José María Turull Torres. A study of homogeneity in relational databases. *Ann. Math. Artif. Intell.*, 33(2-4):379–414, 2001. Also see erratum in 42(4):443–444, 2004.

[Tur06] José María Turull Torres. Relational databases and homogeneity in logics with counting. *Acta Cybern.*, 17(3):485–511, 2006.

[Ull88] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I.* Computer Science Press, Rockville, Maryland, 1988.

[Ull89] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume II.* Computer Science Press, Rockville, Maryland, 1989.

[Var82] Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 137–146, San Francisco, California, USA, May 1982. ACM.

[Zák83] Stanislav Zák. A turing machine time hierarchy. *Theor. Comput. Sci.*, 26:327–333, 1983.

Index