UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

Francicleber Martins Ferreira

Expressividade e Complexidade em Lógicas Preferenciais, Híbridas e de Grau Limitado

Fortaleza
7 de Dezembro de 2012

Francicleber Martins Ferreira

Expressividade e Complexidade em Lógicas Preferenciais, Híbridas e de Grau Limitado

Tese apresentada ao Curso de Doutorado em Ciência da Computação do Departamento de Computação da Universidade Federal do Ceará, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Lógica

Orientadora: Prof.ª Dr.ª Ana Teresa de Castro Martins.

Fortaleza
7 de Dezembro de 2012

Aos meus pais,
Francisco Ferreira Manço e Maria Lucineide Martins Ferreira,
e à minha avó,
Ormesinda Pereira da Rocha
(2.2.1921 – 26.10.2008 – ∞).

Agradecimentos

Foi uma grande experincia cursar um doutorado. Eu conheci muitas pessoas que desempenharam um papel importante nesse processo. Eu gostaria de agradec-lo aqui, pelo menos aqueles que eu me lembro agora.

Eu gostaria de agradecer à CAPES a bolsa de estudos que me permitiu fazer esse trabalho. Gostaria de agradecer à CAPES e ao DAAD a bolsa de doutorado sanduíche que me permitiu fazer parte do meu doutorado na Alemanha.

Eu gostaria de agradecer aos Professores Mario Benevides, Carlos Fisch Brito, João Fernando, Hermann Haeusler and Marcelino Pequeno por terem participado do meu exame de qualificação e da minha defesa de tese.

Eu gostaria de agradecer minha orientadora Professora Ana Teresa pelo apoio e atenção que ela me deu durante o doutorado. Também gostaria de agradecer ao Professor Erich Grädel por ter aceitado me orientar durante o tempo que estive na Alemanha. Eu gostaria também de agradecer ao Professor Wolfgang Thomas pelas oportunidades em que pude conversar com ele.

Eu gostaria de agradecer aos meus amigos do Departamento de Computação da UFC e dos grupos MGI e I7 da RWTH Aachen na Alemanha e também aos demais amigos que fiz na Alemanha. Um agradecimento especial a Wied Pakusa. Eu tive a oportunidade de discutir com Wied sobre os problemas a respeito da quantificação de relaǫões de grau limitado e suas contribuições foram importantes para esclarecer alguns problemas, especialmente aqueles da Seção 4.4. E também a Roman Rabinovich toda a ajuda que ele me deu e, claro, a aulas de xadrez. Eu agradeço especialmente a Cibele Matos Freire toda a motivação que ela me deu. Ela me ajudou bastante, muitas vezes sem sequer saber.

Finalmente, gostaria de agradecer minha família, especialmente meus pais Francisco Ferreira Manço e Maria Lucineide Martins Ferreira e minha avó Maria Ormesinda Pereira Rocha.

Acknowledgments

It was a big experience to do a PhD. I met a lot of people who played an important role in this process. I would like to acknowledge them here, at least those that I remember.

I would like to thank CAPES for the scholarship that allowed me to do this work. I would like to thank CAPES and DAAD for the sandwich scholarship that allowed me to do part of my PhD in Germany.

I would like to thank the Professors Mario Benevides, Carlos Fisch Brito, João Fernando, Hermann Hausler and Marcelino Pequeno for their participation in committees of my qualification and my thesis defense.

I would like to thank my adviser Professor Ana Teresa for the support and attention she gave me during my PhD. I would like to thank Professor Erich Grädel for accepting to advise me during the time I spent in Germany. I also would like to thank Professor Wolfgang Thomas for the few but enjoyable conversations I could have with him.

I would like to thank my friends from the Computer Science Department at UFC and from the MGI and I7 groups at RWTH in Germany and also the friends I made in Germany. A special thank to Wied Pakusa. I had the opportunity to discuss with Wied about the problems regarding the quantification bounded-degree relations and his contributions were important to clarify some problems, especially those in Section 4.4. And to Roman Rabinovich for all the help he gave me and, of course, the chess classes. I specially thank Cibele Matos Freire for all the motivation she gave me. She helped me a lot, many times without even knowing.

Finally, I would like thank my family, specially my parents Francisco Ferreira Manço and Maria Lucineide Martins Ferreira and my grandmother Ormesinda Pereira da Rocha.

"Num tem nada não.
Também no meu coração
Vou ter relâmpo e truvão.
Minh'alma vai florecer."
Elomar

Resumo

Nós investigamos a teoria dos modelos de Lógicas Preferenciais, Lógica Híbrida e fragmentos da Lógica de Segunda-Ordem com relação a modelos finitos. A semânticas dessas lógicas diferem da abordagem clássica pelo uso de relações entre modelos ou por restringir a cardinalidade dos modelos a cardinais finitos.

Este trabalho tem três partes. Na primeira parte deste trabalho nós estudamos a teoria dos modelos de lógicas preferenciais. Lógicas preferenciais surgem no contexto do raciocínio não-monotônico em Inteligência Artificial. A principal característica dessas lógicas é a existência de uma relação entre modelos. Isso permite a definição de uma relação de consequência não monotônica considerando-se os modelos minimais de um conjunto de sentenças. Usando a abordagem da Teoria dos Modelos Abstrata, nós generalizamos alguns resultados de expressividade para classes de lógicas preferenciais. Nós mostramos que sempre que uma classe de modelos minimais de um conjunto finito de sentenças é axiomatizável, então tal classe é finitamente axiomatizável. Nós mostramos que se tal classe define implicitamente um símbolo do vocabulário, existe uma axiomatização finita de uma forma particular, a saber, o conjunto finito de sentenças inicial mais uma definição explícita para o símbolo definido.

Na segunda parte desse trabalho, nós investigamos a teoria dos modelos finitos da Lógica Híbrida. Lógicas Híbridas são extensões da lógica modal através de termos híbridos que se referem a estados individuais em um modelo de Kripke. Nós estudamos a complexidade computacional dos problemas de *model-* e *frame-checking* para a Lógica Híbrida. Nós mostramos que para cada problema de grafos na Hierarquia Polinomial e cada número n, existe uma fórmula que exprime esse problema para grafos de cardinalidade n. Nós mostramos que o tamanho das fórmulas é limitado por um polinôimio em n. Nós mostramos que podemos abrir mão das modalidades globais se nos limitarmos a grafos conexos com *loops*. Nós definimos fragmentos da Lógica Híbrida que correspondem a

cada nível da Hierarquia Polinomial. Isso nos leva a uma prova alternativa da NP-dificuldade do problema de *model-checking* para um fragmento específico de da Lógica Híbrida.

Na última parte desse trabalho, nós exploramos a complexidade descritiva da lógica obtida ao restringirmos a quantificação de segunda-order a relações de grau limitado. Baseados em trabalhos anteriores de Schwentick *et al.* e de Grandjean e Olive, nós introduzimos a Lógica de Segunda-Ordem de Grau Limitado e mostramos que ela captura a classe ALIN de classes de estruturas unárias aceitas por uma máquina de acesso randômico em tempo linear e um número fixo de alternâncias dependente apenas do problema. Nós estendemos essa lógica com o operador de fecho transitivo sobre relações de ordem superior sobre relações de grau limitado. Nós mostramos que a Lógica de Segunda-Ordem de Grau Limitado com Fecho Transitivo captura quantidade linear de registradores em uma máquina de acesso randômico não-determinística onde os valores armazenados em cada registrador durante a computação são limitados por uma função linear na cardinalidade da estrutura de entrada.

Abstract

We investigate the model theory of Preferential Logics, Hybrid Logic and fragments of Second-Order Logic with respect to finite models. The semantics of these logics differ from the semantics of classical logics either by using relations between models or by restricting the cardinality of the models considered.

This work has three main parts. In the first part of this work we study the model theory of preferential logics. Preferential logics arise in the context of nonmonotonic reasoning in Artificial Intelligence. The main characteristic of those logics is the existence of a relation between models. It allows the definition of a nonmonotonic consequence relation by considering the minimal models of a set of sentences. Using the approach of Abstract Model Theory we generalize some expressiveness results to classes of preferential logics. We show that whenever a class of minimal models of a finite set of sentences is axiomatizable, without considering the preference relation, then it is finitely axiomatizable. We also show that when such class of minimal models implicitly defines a symbol, then the finite axiomatization can be put in a very specific form, namely, the initial set of sentences plus a explicit definition for the symbol.

In the second part of this work, we investigate the finite model theory of Hybrid Logic. Hybrid Logics are extensions of modal logics with hybrid terms which refer to single states in a Kripke model. We study the complexity of the model- and frame-checking problems for Hybrid Logic. We show that for each graph problem in the Polynomial Hierarchy and each natural number n there is a formula which expresses this problem for graphs of cardinality n. We also show that the size of such formulas is bounded by a polynomial in n. We show that one can disregard the global modalities if one consider only connected graphs with loops. We define fragments which correspond to each degree of the Polynomial Hierarchy. This leads to an alternative proof of the NP-hardness of the model-checking problem for an specific fragment of Full Hybrid Logic.

In the last part of this work, we explore the descriptive complexity of the logic obtained by restricting second-order quantification to relations of bounded degree. Based on previous work from Schwentick et al. and Grandjean and Olive, we introduce the Bounded-Degree Second-Order Logic and show that it captures the class ALIN of classes of unary structures accepted by a alternating random access machine in linear time and bounded number of alternations. We also extend this logic with the transitive closure operator on high-order relations on bounded-degree relations. We show that the Bounded-Degree Second-Order Logic with Transitive Closure Operator captures linear number of registers in a nondeterministic random access machine provided that registers store values bounded by a linear function in the cardinality of the input structure.

Sumário

Contents

Chapter 1

Introduction

Model Theory is the branch of mathematical logic that studies the relation between structures and formal languages defined through satisfiability relations. One of the main tasks of the field is to establish the expressive power of the languages studied. That is, we want to characterize which classes of structures can be expressed by the sentences of the language. Classical Model Theory is mainly concerned with first- and high-order languages and the methods used heavily depend on the existence of infinite models. Typical results in this context are Löwenheim-Skolem theorem, compactness and categoricity [CK73].

From a semantical standpoint, a logic can be regarded as a triple $\mathcal{L} = (L, \mathbb{M}, \models)$, where L is a set of sentences, the language of $\mathcal{L}$, $\mathbb{M}$ is a set of interpretations for L, and $\models$ the satisfiability relation between interpretations and sentences. In classical model theory, interpretations are mathematical structures. A class of structures is expressible in $\mathcal{L}$ if it is the class of models of some sentence in L. Hence, in model theory we are mainly concerned with describing which classes of structures can be expressed by a formal language. Usually one makes no restriction on the cardinality of the structures involved and $\mathbb{M}$ is a simple set or class, that is, no previous relation between structures is regarded.

Research in Computer Science has turned our attention to problems where some previous relation among structures should be considered or there is some restriction on the cardinality of the structures. That is the case, for instance, of some logical approaches to nonmonotonic reasoning in Artificial Intelligence such as Circumscription and Preferential Logics, where structures are related through preference relations [Sho87, McC86], and in Database Theory and in Descriptive Complexity [Imm99], where structures are essentially finite.

In this work, we investigate the expressive power of logics whose model theory deviates from the classical approach by some of these two aspects: either their semantics depends on some relation between interpretations or interpretations are finite objects. Our objective is to explore their model theory with emphasis on the expressive power of the languages studied.

This thesis has three main parts. The overall motivation of this work is the study of the model theory of some logics whose semantics deviate from the classical approach, as said above. Each part has its own particular objectives motivated by specific questions. In Chapter 2, we present some results for classes of preferential logics, whose semantics involve a relation between interpretations, using the approach of Abstract Model Theory [GMV07, Bez07, BF85]. Here we are interested in the study of the expressiveness of a class of logics with respect to the preferential semantics, that is, which classes of models can be expressed. In Chapters 3 and 4 we investigate the finite model theory of Hybrid and Bounded-Degree Second-Order Logic, respectively. Here the focus is on the descriptive complexity, that is, how the logical resources with which logical languages are equipped, such as quantifiers, fixed-point operators, etc., are related to their ability to capture complexity classes [Imm99]. In the following sections, we introduce the reader to these three works.

1.1 Preference Relations

The necessity to deal with reasoning over incomplete information in Artificial Intelligence has led to the development of several logics to formalize nonmonotonic reasoning [Bre91, MTR93]. Intelligent behaviour involves making inferences beyond one can get by deduction over current information, when it is incomplete. In this case, it is possible that previously inferred conclusions are confronted with new, reliable information, but which contradicts those previous inferences. Hence, previously concluded assertions should be dropped in virtue of the new information acquired.

This behaviour of consequence relations is called nonmonotonicity, since old inferences are not necessarily preserved as the knowledge base increases. Nonmonotonic logics are logics whose consequence relation presents this behaviour. Examples of nonmonotonic logics are Reiter's Default Logic [Rei80, Rei82, Poo94], McCarthy's [McC80, McC86], Doyle and McDermott's Nonmonotonic Logic and Shoham's Preferential Models [Sho87, LM90]. Shoham's

Preferential Models are a semantical approach to nonmonotonic logic. It formalizes nonmonotonic consequence by using a preference relation between models. Each preference relation between models gives rise to a consequence relation. Hence, it allows us to generalize results for wide classes of nonmonotonic logics whose semantics are based on preferential models.

In recent years, the interest in unifying approaches to logics has increased [Bez06, Bez07]. Researchers are intended to develop techniques that allow us to generalize results for classes of logics. The term Universal Logic has been used to designate the general study of logics [Bez07]. In this context, a logic is treated as a structure $(L, \vdash)$, where $\vdash$ is intended to be a consequence relation. Classes of logics can be defined by establishing the properties that such structures have. Beside this, by treating such structures as objects of study, one can develop theories of logics in some formal language.

An alternative way to deal with logics as structures is used in Abstract Model Theory [GMV07, BF85]. Here, a logic is regarded as a triple $(L, \mathbb{M}, \models)$ where $\models$ is intended to be a satisfiability relation, as we said above. Some of the first results in this field are Lindström's theorems [Lin69, EFT94b]. They give a characterization of first-order by stating that any logic, regarded as such a triple, which attend to certain properties, such as compactness, Löwenheim-Skolem and to be closed under boolean connectives, to be at least so expressive as first- order logic. In [GMV07], Abstract Model Theory is suggested as the branch of Universal Logic where semantic methods are applied.

We are interested in the study of nonmonotonic logics using the framework of Abstract Model Theory. This approach will allow us to proof our results for classes of logics in a uniform way.

In Chapter 2, we investigate the expressive power of nonmonotonic logics defined through preferential models. We generalize results of [FM11b] for Circumscription to a class of preferential models following the approach of Abstract Model Theory [GMV07]. We introduce expressible preferential logics, whose preference relations can be defined by some abstract logic. We show that some well-known nonmonotonic logics are preferential. We prove that they are elementary, which means that their preference relation can be defined in first-order logic. We study expressiveness and definability results for a wide class of abstract preferential logics in the spirit of Universal Logic. The two main results of Chapter 2 are:

1. If the class of minimal models of a finite theory of an elementary prefer-

ential logic is axiomatizable, then it is finitely axiomatizable.

2. If a symbol is defined in the class of minimal models of a finite theory of an elementary preferential logic, then this class can be expressed by the finite theory plus an explicit definition for the symbol.

In the other two parts of this work, we deal with the expressiveness of languages over finite models.

1.2 Finite Models

Finite Model Theory was developed in the last decades motivated by problems in Computer Science, specially Database Theory and Complexity Theory [Lib04, Grä02, EF95]. One of the first results in this area is Trakhtenbrot's theorem [EFT94b]. It states that finite satisfiability of first-order sentences is not semidecidable. Methods of Classical Model Theory strongly rely on the fact that models can be infinite and results like compactness and Löwenheim-Skolem do not hold any more and thus cannot be applied when one works with finite models only. This led to the development of new techniques and concepts to deal with the problems which arise when we consider finite models. These techniques are heavily based on games, combinatorics and computability.

In the 1970's, a surprising relationship between logic and computational complexity was established. Fagin [Fag74a] showed that the class of queries computable in nondeterministic polynomial time correspond to the class of queries expressible in existential second-order logic. Previously, Büchi showed that the class of regular languages corresponds to the class of queries on strings expressible in monadic second-order logic [Büc60]. These two results are important because they give a machine-independent characterization of complexity classes. After Fagin's result, researchers started to look for the characterization of other complexity classes using logical languages. It gave raise to the field of Descriptive Complexity [Imm99], whose main purpose is to describe complexity classes by means of expressiveness of logical languages, with the hope that results and techniques from each side can be applied in the other.

In Chapters 3 and 4, we present some results on Finite Model Theory and Descriptive Complexity. In Chapter 3, we study the expressibility of problems in the polynomial hierarchy through sequences of formulas of hybrid logic [ABM01]. Hybrid logic extends usual modal logics with terms which refer to

states in the underlying Kripke semantics. Modal logics are particularly suitable to talk about relational structures, specially graphs [BDRV02]. Hybrid languages have been introduced to deal with some lack of expressiveness in tense logic [Bla06]. Since then, further applications have been investigated and more operators were introduced to deal with hybrid terms [Gor94, Gor96]. In [BS09], Benevides and Schechter use formulas of hybrid logic to express some graph properties in NP like being hamiltonian. For each n they show a formula ψ_n which expresses such graph properties for graphs with n vertices. They suggest to investigate whether other graph properties in NP can be expressed in hybrid logic. We show how graph problems in the Polynomial Hierarchy (PH) can be expressed by sequences of formulas of hybrid logic. We are particularly interested in how the size of formulas grows in this sequence. The main results of this chapter are:

1. The existence of formulas $\psi^n_{\mathcal{G}}$ of full hybrid logic which express a graph property $\mathcal{G}$ for graphs of size n for each graph property $\mathcal{G}$ in PH.

2. The size of such $\psi^n_{\mathcal{G}}$ formulas is polynomial in n.

3. We give an alternative proof of the NP-hardness of a fragment of hybrid logic different from the one presented in [tCF05].

In Chapter 4, we investigate a restriction of second-order logic where quantified relation variables range over relations of bounded-degree. In [GO98], Grandjean and Olive define the complexity class NLIN of problems solved in nondeterministic linear time on a registers machine. They show that sets of functions in NLIN correspond exactly to those definable in the existential fragment of a restriction of second-order logic where second-order quantifiers range over unary functions only using formulas whose first-order part has only one variable which is universally quantified. In [DLS98], Schwentick et al. investigate several restrictions of existential second-order quantification and divide them in four classes and separate them with respect to expressive power. In particular, they separate existential quantification of unary functions from existential quantification of binary relations of bounded-degree, that is, relations whose underlying Gaifman graph has degree bounded by a constant. Based on these results, we consider the logic BDSO obtained when both existential and universal quantifiers are considered. We also consider the logic BDSO(TC) obtained by introducing the second-order transitive closure operator on relations of bounded-degree. The main results of this chapter are:

1. BDSO captures linear time on an alternating RAM with constant number of alternations.

2. BDSO(TC) captures linear space on a nondeterministic RAM.

In the next section, we introduce some notation an basic definitions.

1.3 Preliminaries

The basic logic notation for classical logic used throughout this text follows that in [EFT94b]. For instance, a symbol set S is a set of predicate, function and constant symbols. We say that two symbol sets S and S' are similar iff there is a similarity function from S to S', that is, a bijection f from S to S' which maps each n-ary predicate symbol (n-ary function symbol, constant symbol) in S to an n-ary predicate symbol (n-ary function symbol, constant symbol) in S'. We write $f : S \sim S'$ to say that S is similar to S' and f is a similarity function from S to S'. A formula written using the symbols in S is an S-formula. An S-structure is a pair $\mathfrak{A} = (A, \sigma)$, where A is a set, the domain of $\mathfrak{A}$ and σ is a map which associates an n-ary relation $\sigma(P) = P^{\mathfrak{A}} \subset A^n$ to each n-ary relation symbol $P \in S$, an n-ary function $\sigma(f) = f^{\mathfrak{A}} : A^n \to A$ to each n-ary function symbol $f \in S$ and an element $\sigma(c) = c^{\mathfrak{A}} \in A$ to each constant symbol $c \in S$. We use Fraktur capital letters, such as $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \ldots$, to denote structures and the corresponding Roman capital letters $A, B, C, \ldots$ for their domains. If $f : S \sim S'$ and $\mathfrak{A} = (A, \sigma)$ is an S-structure, we write $\mathfrak{A}_f$ to refer to the similar S'-structure $\mathfrak{A}_f = (A, \sigma \circ f)$. We write $\mathfrak{A} \cong \mathfrak{B}$ meaning that $\mathfrak{A}$ is isomorphic to $\mathfrak{B}$. The cardinality $|\mathfrak{A}|$ of a structure $\mathfrak{A}$ is the cardinality of its domain.

Given a symbol set $S = \{R_1, \ldots, R_n, f_1, \ldots, f_m, c_1, \ldots, c_k\}$, we usually denote an S-structure $\mathfrak{A}$ as $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_n^{\mathfrak{A}}, f_1^{\mathfrak{A}}, \ldots, f_m^{\mathfrak{A}}, c_1^{\mathfrak{A}}, \ldots, c_k^{\mathfrak{A}})$, where $\sigma(R_i) = R_i^{\mathfrak{A}}$, $1 \leq i \leq n$, $\sigma(f_i) = f_i^{\mathfrak{A}}$, $1 \leq i \leq m$ and $\sigma(c_i) = c_i^{\mathfrak{A}}$, $1 \leq i \leq k$. We say that two S-structures $\mathfrak{A}$ and $\mathfrak{B}$ agree on $S' \subseteq S$ iff $s^{\mathfrak{A}} = s^{\mathfrak{B}}$, for each $s \in S'$. Given $S' \subset S$ and an S-structure $\mathfrak{A}$, the S'-reduct of $\mathfrak{A}$ is the S'-structure $\mathfrak{A}|_{S'}$ with the same domain of $\mathfrak{A}$ and such that $s^{\mathfrak{A}|_{S'}} = s^{\mathfrak{A}}$ for each $s \in S'$. We call $\mathfrak{A}$ an expansion of $\mathfrak{A}|_{S'}$.

An S-structure $\mathfrak{A}'$ is a substructure of the S-structure $\mathfrak{A}$ if $A' \subseteq A$ and $s^{\mathfrak{A}'}$ is the restriction of $s^{\mathfrak{A}}$ to A' for each relation or function symbol s in S and $c^{\mathfrak{A}} = c^{\mathfrak{A}'}$ for each constant symbol c in S. We call $\mathfrak{A}$ an extension of $\mathfrak{A}'$. We introduce the notation $\mathfrak{A}|^{A'}_{S'}$, where $\mathfrak{A}$ is an S-structure, S' is a subset of S and A' is a subset of the domain A of $\mathfrak{A}$, to denote the substructure of

domain A' of the S'-reduct $\mathfrak{A}|_{S'}$ of $\mathfrak{A}$. [BdFV01] calls $\mathfrak{A}$ an exp*e*nsion (an expansion of an extension) of $\mathfrak{A}|^{A'}_{S'}$. We call $\mathfrak{A}|^{A'}_{S'}$ a *subduct* of $\mathfrak{A}$, that is, a substructure of a reduct, if it exists[1]. Two S-structures $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic iff there is a bijection $h : A \to B$ such that, for each k-ary relations symbol $R \in S$ and all $a_1, \ldots, a_k \in A$, $(a_1, \ldots, a_k) \in R^{\mathfrak{A}}$ iff $(h(a_1), \ldots, h(a_k)) \in R^{\mathfrak{B}}$, for each k-ary function symbol $f \in S$, and all $a_1, \ldots, a_k \in A$, $f^{\mathfrak{A}}(a_1, \ldots, a_k) = f^{\mathfrak{A}}(h(a_1), \ldots, h(a_k))$, and each constant symbol $c \in S$, $c^{\mathfrak{A}} = c^{\mathfrak{B}}$.

We define below the first- and second-order logics.

Definition 1.1 (Language of First-Order Logic) *Let S be a symbol set. An S-*term *is the least set which contains constant symbols in S, first-order variables and function symbols in S applied to S-terms. For example, if f is a ternary function symbol in S, c and c' constant symbols and x a first-order variable, then $fcxc'$ is an S-term.*

An atomic first-order S-formula *is a first-order relation symbol in S applied to terms. For example, if R is a binary relation symbol in S and t_1 and t_2 are S-terms, then Rt_1t_2 is an atomic first-order S-formula.*

The set of first-order S-formulas *is the least set which contains the atomic formulas and such that, if x is a first-order variable, α and β are S-formulas, then $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$, $(\alpha \leftrightarrow \beta)$, $\exists x\alpha$ and $\forall x\alpha$ are S-formulas.*

We now introduce the semantics of first-order logic. To do this, we need the concept of first-order interpretation.

Definition 1.2 (First-Order Interpretation) *A* first-order S-interpretation *(or simply an* S-interpretation*) is a $\mathfrak{I} = (\mathfrak{A}, \beta)$ where $\mathfrak{A}$ is an S-structure and β is an assignment of first-order variables that maps first-order variables into elements in the domain A of $\mathfrak{A}$. Given an element $a \in A$ and a variable x, we define the assignment $\beta\frac{a}{x}$ such that $\beta\frac{a}{x}(x') = \beta(x')$ if $x' \neq x$ and $\beta\frac{a}{x}(x') = a$ if $x = x'$. We define the interpretation $\mathfrak{I}\frac{a}{x} = (\mathfrak{A}, \beta\frac{a}{x})$.*

The semantics of first-order logic is defined as follows:

Definition 1.3 (Semantics of First-Order Logic) *Let $\mathfrak{I} = (\mathfrak{A}, \beta)$ be an S-interpretation. We define $\mathfrak{I}(t) \in A$ for each S-term t as:*

- $\mathfrak{I}(x) = \beta(v)$ *for each first-order variable x;*

[1] It is not difficult to see that for some structures $\mathfrak{A}$ not any a subset of the domain of $\mathfrak{A}$ is the domain of a substructure (and in the same sense, of a subduct) of $\mathfrak{A}$ if there are constant symbols or function in the symbol set.

- $\mathfrak{I}(c) = c^{\mathfrak{A}}$ *for each constant* c *in* S*;*
- $\mathfrak{I}(ft_1 \dots f_n) = f^{\mathfrak{A}}(\mathfrak{I}(t_1), \dots, \mathfrak{I}(t_n))$*, for each* n*-ary function symbol* f *in* S *and* S*-terms* $t_1, \dots, t_n$*.*

The satisfiability relation between S*-interpretations and* S*-formulas is defined as:*

- $\mathfrak{I} \models Rt_1 \dots t_n$ *iff* $(\mathfrak{I}(t_1), \dots, \mathfrak{I}(t_n)) \in R^{\mathfrak{A}}$*, for each* n*-ary relation symbol* R *and* S*-terms* $t_1, \dots, t_n$*;*
- $\mathfrak{I} \models \neg\alpha$ *iff* $\mathfrak{I} \not\models \alpha$*;*
- $\mathfrak{I} \models (\alpha \wedge \beta)$ *iff* $\mathfrak{I} \models \alpha$ *and* $\models \beta$*;*
- $\mathfrak{I} \models (\alpha \vee \beta)$ *iff* $\mathfrak{I} \models \alpha$ *or* $\models \beta$*;*
- $\mathfrak{I} \models (\alpha \rightarrow \beta)$ *iff, if* $\mathfrak{I} \models \alpha$ *then* $\models \beta$*;*
- $\mathfrak{I} \models (\alpha \leftrightarrow \beta)$ *iff* $\mathfrak{I} \models \alpha$ *iff* $\models \beta$*;*
- $\mathfrak{I} \models \exists x\alpha$ *iff there is an* $a \in A$ *such that* $\mathfrak{I}\frac{a}{x} \models \alpha$*;*
- $\mathfrak{I} \models \forall x\alpha$ *iff for all* $a \in A$ *we have* $\mathfrak{I}\frac{a}{x} \models \alpha$*.*

If α *has no free variables and* $\mathfrak{A}$ *is a structure, then* $\mathfrak{A} \models \alpha$ *iff there is an interpretation* $\mathfrak{I} = (\mathfrak{A}, \beta)$ *such that* $\mathfrak{I} \models \alpha$*.*

Let $\overline{x} = x_1, \dots, x_r$ be a tuple of variables. The quantifiers $\exists^{=c}\overline{x}$ (exists exactly c tuples), $\exists^{\leq c}\overline{x}$ (exists at most c tuples) and $\exists^{\geq c}\overline{x}$ (exists at least c tuples) can be defined in first-order logic. Let $\overline{x_i} = x_{i,1} \dots x_{i,r}$. We have:

$$\exists^{\geq c}\overline{x}\phi(\overline{x}) \equiv \exists\overline{x_1} \dots \overline{x_c}(\bigwedge_{1 \leq i < j \leq c} \overline{x_i} \neq \overline{x_j} \wedge \bigwedge_{1 \leq c \leq i} \phi(\overline{x_i})),$$

where $\exists\overline{x_i} = \exists x_{i,1} \dots \exists x_{i,r}$ and $\overline{x_i} \neq \overline{x_j} = \bigvee_{1 \leq k \leq r}(x_{i,k} \neq x_{j,k})$,

$$\exists^{\leq c}\overline{x}\phi(\overline{x}) \equiv \neg\exists^{\geq c+1}\overline{x}\phi(\overline{x})$$

and

$$\exists^{=c}\overline{x}\phi(\overline{x}) \equiv \exists^{\geq c}\overline{x}\phi(\overline{x}) \wedge \neg\exists^{\geq c+1}\overline{x}\phi(\overline{x}).$$

Second-order logic extends first-order logic in the following way:

Definition 1.4 (Language of Second-Order Logic) *Let S be a symbol set. Besides first-order variables, the alphabet of second-order logic has relation variables of all possible arities. An* atomic second-order S-formula *is either a first-order atomic S-formula or a relational variable applied to S-terms. For example, if X is a ternary relation variable and t_1, t_2 and t_3 are S-terms, then $Xt_1t_2t_3$ is a second-order atomic S-formula.*[2]

The set of second-order S-formulas *is the least set which contains the atomic formulas and such that, if x is a first-order variable, X is a second-order variable and α and β are S-formulas, then $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$, $(\alpha \leftrightarrow \beta)$, $\exists x\alpha$, $\forall x\alpha$, $\exists X\alpha$, $\forall X\alpha$ are S-formulas.*

We now introduce the semantics of second-order logic. To do this, we need the concept of second-order interpretation.

Definition 1.5 (Second-Order Interpretation) *A* second-order S-interpretation *is a pair $\mathfrak{I} = (\mathfrak{A}, \beta)$ where $\mathfrak{A}$ is an S-structure and β is an assignment of first- and second-order variables that maps first-order variables into elements in the domain A of $\mathfrak{A}$ and r-ary relation variables into r-ary relations on A. Given a r-ary relation variable X and a r-ary relation $\mathbf{X}$ on A, we define the assignment $\beta\frac{\mathbf{X}}{X}$ as $\beta\frac{\mathbf{X}}{X}(X') = \beta(X')$ if $X' \neq X$ and $\beta\frac{\mathbf{X}}{X}(X') = \mathbf{X}$ if $X = X'$. We define the interpretation $\mathfrak{I}\frac{\mathbf{X}}{X} = (\mathfrak{A}, \beta\frac{\mathbf{X}}{X})$.*

The semantics of first-order logic is defined as follows:

Definition 1.6 (Semantics of Second-Order Logic) *Let $\mathfrak{I} = (\mathfrak{A}, \beta)$ be an S-interpretation, X be a r-ary relation variable and $t_1, \ldots, t_r$ be S-terms. We extend the satisfiability relation of first-order logic to second-order logic as follows:*

- $\mathfrak{I} \models Xt_1 \ldots t_r$ *iff* $(\mathfrak{I}(t_1), \ldots, \mathfrak{I}(t_r)) \in \beta(X)$;
- $\mathfrak{I} \models \exists X\alpha$ *iff there is an* $\mathbf{X} \in A^r$ *such that* $\mathfrak{I}\frac{\mathbf{X}}{X} \models \alpha$;
- $\mathfrak{I} \models \forall X\alpha$ *iff for all* $\mathbf{X} \in A^r$ *we have* $\mathfrak{I}\frac{\mathbf{X}}{X} \models \alpha$.

Now we define the hierarchies Π, Σ and Δ for first- and second-order formulas.

[2] We often regard a second-order atomic formula as a first-order atomic formula too, in a extended symbol set where X is not regarded as a relation variable but a relation symbol. That is because, since in this formula there is no second-order quantification involved, it has essentially a first-order character.

Definition 1.7 (Π_n^0, Σ_n^0 and Δ_n^0) *We define the sets Π_n^0, Σ_n^0 and Δ_n^0 of first-order formulas as:*

- $\Pi_0^0 = \Sigma_0^0 = \Delta_0^0 =$ *the set of quantifier-free formulas;*
- $\Pi_{n+1}^0 =$ *the set of formulas of the form* $\forall x_1 \ldots \forall x_m \phi$ *where* $\phi \in \Sigma_n^0$*;*
- $\Sigma_{n+1}^0 =$ *the set of formulas of the form* $\exists x_1 \ldots \exists x_m \phi$ *where* $\phi \in \Pi_n^0$*;*
- $\Delta_n^0 = \Pi_n^0 \cap \Sigma_n^0$

Similarly for second-order logic we have:

Definition 1.8 (Π_n^1, Σ_n^1 and Δ_n^1) *We define the sets Π_n^1, Σ_n^1 and Δ_n^1 of first-order formulas as:*

- $\Pi_0^1 = \Sigma_0^1 = \Delta_0^1 =$ *the set of first-order formulas;*
- $\Pi_{n+1}^1 =$ *the set of formulas of the form* $\forall X_1 \ldots \forall X_m \phi$ *where* $\phi \in \Sigma_n^1$*;*
- $\Sigma_{n+1}^1 =$ *the set of formulas of the form* $\exists X_1 \ldots \exists X_m \phi$ *where* $\phi \in \Pi_n^1$*;*
- $\Delta_n^1 = \Pi_n^1 \cap \Sigma_n^1$

We usually use these sets to refer to the quantifier prefix of a formula. For instance a Π_2^0 formula is a formula with a quantifier prefix composed by two blocks: a universal quantifier block followed by an existential quantifier block. A $\Sigma_1^1\Pi_1^0$ formula is a formula with a block of existential second-order quantifiers followed by a formula in Π_1^0.

1.3.1 Queries and Complexity Classes

We will now introduce some notation and definitions used in finite model theory and descriptive complexity. We follow the notation in [Lib04, EF95, Imm99].

Complexity classes are defined based on computation models. Many complexity classes are defined on Turing machines and its variations. A Turing machine is a quadruple $M = (K, \Sigma, \delta, q_0)$, where K is a finite set of states, $q_0 \in K$ is the initial state, Σ is a finite set of symbols (the alphabet of M). Σ has two special symbols $\sqcup$, the blank symbol, and $\triangleright$, the first symbol. δ is the transition function, a function in

$$K \times \Sigma \to (K \cup \{\text{“}yes\text{”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}.$$

A configuration of the machine is a triple (q, w, i) where q is the current state, w is a string in Σ^* and $0 < i \leq |w|$ is the position of the symbol in w currently scanned. The machine operates by performing the transitions specified by δ, writing a symbol in the current position, changing its state and moving the head. It starts with a string in the first positions of the tape and we say that it accepts the string if it reaches the accepting state "*yes*" at some point of the computation.

Complexity classes are defined based on computation models and the resources used to perform the computation [Pap03]. Usually, complexity classes are defined as classes of languages, that is, sets of sets of strings on some finite alphabet, since most common complexity classes are defined on Turing machines. For example, the class DSPACE$[f(n)]$ is the class of languages accepted by a deterministic Turing machine using space $O(f(n))$, where n is the size of the input. The class NSPACE$[f(n)]$ is the class of languages accepted by a nondeterministic Turing machine using space $O(f(n))$ and the class ASPACE$[f(n), g(n)]$ is the class of languages accepted by an alternating Turing machine using space $O(f(n))$ and allowing $O(g(n))$ alternations. Similar classes can be defined for the resource time.

In finite model theory, interpretations are finite structures whose domain are natural numbers, that is, if $\mathfrak{A}$ has cardinality n, then its domain is $n = \{0, \ldots, n-1\}$. We denote by STRUC$[S]$ the class of finite S-structures. Beside this, to each symbol set S there is a subset L^S of the language whose elements are called S-sentences. The satisfiability relation has the additional property that if $\mathfrak{A} \models \phi$, for an S-sentence ϕ, then $\mathfrak{A}$ is an S-structure. When talking about finite model theory, the logics considered will have the properties described in Definition 2.7. As the interpretations considered are finite structures, we usually omit the set of interpretations $\mathbb{M}$ when we consider a logic $\mathcal{L} = (L, \models)$.

We now introduce the concept of boolean query, which are decision problems on structures.

Definition 1.9 (Boolean Query) *A boolean query is a function*

$$f : STRUC[S] \to \{0, 1\}$$

for some symbol set S such that $f(\mathfrak{A}) = f(\mathfrak{B})$ if $\mathfrak{A} \cong \mathfrak{B}$. We usually regard a query f as the set

$$\mathcal{A} = \{\mathfrak{A} \in STRUC[S] | f(\mathfrak{A}) = 1\}.$$

Examples of queries are the set of hamiltonian graphs and the set of finite abelian groups. Given a logic $\mathcal{L} = (L, \models)$, an S-sentence $\phi \in L^S$ defines a query

$$\mathcal{A} = \{\mathfrak{A} \in \text{STRUC}[S] | \mathfrak{A} \models \phi\}.$$

We are interested in the complexity of the decision problem defined by a sentence of some logic $\mathcal{L}$. To do this, we need to precisely define what means a decision problem on structures to belong to a complexity class. Such definition depends on the computation model on which the complexity class is defined. That is because we need to code finite structures as inputs to such complexity classes and such code may differ according to the computation model. Turing machines have strings as inputs while register machines have sequences of natural numbers as inputs. As an example, we will show how to code structures as strings.

Let $S = \{R_1, \ldots, R_l, c_1, \ldots, c_m\}$ be a relational symbol set and

$$\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_l^{\mathfrak{A}}, c_1^{\mathfrak{A}}, \ldots, c_r^{\mathfrak{A}})$$

an S-structure of cardinality $|A| = n$. To each k-ary relation $\mathbf{R}$ on A we associate a binary string $\text{bin}(\mathbb{R})$ of size n^k in the following way. Let $\overline{a} \in A^k$ be the m-th element in the lexicographic ordering of A^k induced by the natural ordering of $A = \{0, \ldots, n-1\}$. Then $\text{bin}(\mathbf{R})$ has a 1 in position m iff $\overline{a} \in \mathbf{R}$. To each element $0 \leq a \leq n-1 \in A$ we define $\text{bin}(a)$ as the binary representation of a. We then code the structure $\mathfrak{A}$ as

$$\text{bin}(\mathfrak{A}) = \text{bin}(R_1^{\mathfrak{A}}) \ldots \text{bin}(R_l^{\mathfrak{A}}) \text{bin}(c_1^{\mathfrak{A}}) \ldots \text{bin}(c_r^{\mathfrak{A}}).$$

Let $\mathcal{A}$ be a boolean query on S-structures. It gives rise to a language $L_{\mathcal{A}} = \{\text{bin}(\mathfrak{A}) \in \{0,1\}^* | \mathfrak{A} \in \mathcal{A}\}$. Let $\mathcal{C}$ be a complexity class. We say that a query $\mathcal{A}$ belongs to $\mathcal{C}$ ($\mathcal{A} \in \mathcal{C}$) iff $L_{\mathcal{A}}$ is in $\mathcal{C}$.

Definition 1.10 ($\mathcal{L}$ captures $\mathcal{C}$) *A logic $\mathcal{L}$ captures a complexity class $\mathcal{C}$ iff each query expressible in $\mathcal{L}$ is in $\mathcal{C}$ and each query in $\mathcal{C}$ is expressible in $\mathcal{L}$.*

The notion of completeness for complexity classes depends on the concept of reduction.

Definition 1.11 (Many-One Reduction) *Let S and S' be symbol sets and $\mathcal{A} \subseteq \text{STRUC}[S]$ and $\mathcal{B} \subseteq \text{STRUC}[S']$ be two queries, a many-one reduction from*

$\mathcal{A}$ to $\mathcal{B}$ is a function

$$f : STRUC[S] \rightarrow STRUC[S']$$

such that

$$\mathfrak{A} \in \mathcal{A} \text{ iff } f(\mathfrak{B}) \in \mathcal{B}.$$

Definition 1.12 *A query $\mathcal{A}$ is* hard for a class $\mathcal{C}$ with respect to reductions in $\mathcal{R}$ *iff for each query $\mathcal{B} \in \mathcal{C}$ there is a reduction from $\mathcal{B}$ to $\mathcal{A}$ in $\mathcal{R}$. $\mathcal{A}$ is* complete *for a class $\mathcal{C}$ with respect to a class of reductions $\mathcal{R}$ iff $\mathcal{A}$ is hard for $\mathcal{C}$ with respect to reductions in $\mathcal{R}$ and $\mathcal{A} \in \mathcal{C}$. A complexity class $\mathcal{C}$ is* closed under reductions in $\mathcal{R}$ *iff, whenever a query $\mathcal{A}$ can be reduced to a query $\mathcal{B} \in \mathcal{C}$, then $\mathcal{A} \in \mathcal{C}$.*

A well known result of descriptive complexity is the correspondence between the polynomial hierarchy and the alternation hierarchy of second-order logic (with respect to finite models) [Imm99, Fag74a].

There are several ways to define the polynomial hierarchy, for example using alternating Turing machines [Imm99]. In this chapter we assume the definition presented in [Pap03], which uses Turing machines with oracles to define PH.

A Turing machine with an oracle is a machine that has the special ability of guessing some specific questions. When a Turing machine has an oracle for a decision problem B, during its execution it can ask for the oracle if some instance of problem B is positive or negative. This is made in constant time, regardless of the size of the instance. We use the notation M^B to define a Turing machine with an oracle for a problem B. In a similar way, we define $\mathcal{C}^{\mathcal{B}}$, where $\mathcal{C}$ and $\mathcal{B}$ are complexity classes, as the class of problems solved by a Turing machine in $\mathcal{C}$ with an oracle in $\mathcal{B}$.

Definition 1.13 (Polynomial Hierarchy) *Consider the following sequence of complexity classes. First, $\Delta_0^p = \Sigma_0^p = \Pi_0^p = P$, the class of problems solvable in deterministic polynomial time, and, for all $i \geq 0$,*

1. $\Delta_{i+1}^p = P^{\Sigma_i^p}$
2. $\Sigma_{i+1}^p = NP^{\Sigma_i^p}$
3. $\Pi_{i+1}^p = coNP^{\Sigma_i^p}$.

We define the Polynomial Time Hierarchy as the class $PH = \bigcup_{i \geq 0} \Sigma_i^p$.

1.3.2 Fagin's Theorem

To illustrate the kind of result in which we are interested in Descriptive Complexity, we will sketch the proof of Fagin's Theorem. The reader familiar with this result may skip to the next chapter. The idea of the proof is to give a suitable codification of the run of a Turing Machine as a relation on the domain of a structure and to give a formula that checks whether the run is an accepting one.

Without loss of generality, let $\mathcal{A}$ be a query on graphs, that is, a subset of STRUC[S] where $S = \{E\}$ and E is a binary relation symbol. Suppose $\mathcal{A} \in$ NP. It means that the language $L_{\mathcal{A}}$ is in NP, that is, there is a nondeterministic Turing machine M that accepts $L_{\mathcal{A}}$ in nondeterministic polynomial time. It means that there is a polynomial $n^{k'}$ such that an accepting run of M has at most $n^{k'}$ steps, where n is the size of the input. Let $\mathfrak{A}$ be a graph of cardinality m. Note that, since the cardinality of $\mathfrak{A}$ is m, then the length of $bin(\mathfrak{A}) = m^2 + m \leq m^3$ for $m \geq 2$. We will write a formula that checks whether $bin(\mathfrak{A})$ is accepted by M. Let $<$ be an order relation on the domain of $\mathfrak{A}$. Using the order $<$ we can define a lexicographic order $<^l$ on l-tuples. We define $<^1 = <$ and $a_1 \dots a_l <^l b_1 \dots b_l$ iff $a_1 < b_1$ or $a_1 = b_1$ and $a_2 \dots a_l <^{l-1} b_2 \dots b_l$.

We will code the configurations of the Turing machine M as tuples. Let $k = 3k'$. Since the machine runs in $n^{k'}$ steps where n is the size of the binary representation of the structure, and $n \leq m^3$, we will use a tuple $\overline{t} = t_1 \dots t_{m^{3k'}} = t_1 \dots t_{m^k}$ of elements of A to represent a timestamp. We will also use a tuple $\overline{p} = p_1 \dots p_{m^k}$ to represent a position in the tape, and a tuple $\overline{h} = h_1 \dots h_{m^k}$ to represent the position of the head on the tape. Let r be the number of states of M.

We will use r elements $q_1, \dots, q_r$ of A to represent the states of M. Let us suppose that the alphabet of the machine consists of $0, 1$ and b, for blank. We will use 3 elements of A to represent these symbols (for the sake of simplicity, lets denote these elements $0, 1$ and b). And we will also use 3 elements to represent the moves of the head l, r and s for left, right and the head to stay in the same position. (To avoid a notation overload, we will use below these symbols both as elements of A and as constants that refer to these elements. It is clear from the context to which one we refer.) We may assume that A is sufficiently large so that there are elements enough to represent states and symbols.

Now, we will use a $2m^k + 1$-ary relation T to represent the tape at each given time. That is, a tuple $\overline{t}\overline{s}c \in T$ iff at time $\overline{t}$ the symbol at position $\overline{p}$ is c, where

$c \in \{0, 1, b\}$.

We will use a $2m^k$-ary relation H to represent the position of the head at time $\overline{t}$. That is the tuple $\overline{t}\overline{p}$ is in H iff the head of the machine is in the position $\overline{p}$ at time $\overline{t}$.

We will use an $m^k + 1$-ary relation Q to represent the state of the machine at a given time. For example, a tuple $\overline{t}q_i \in Q$ iff the state of the machine at time $\overline{t}$ is q_i.

We will use an $m^k + 1$-ary relation D to represent the move made by the head to reach the current position. For example, a tuple $\overline{t}l \in D$ iff the head moved to the left in the transition from the previous configuration to the current configuration at time $\overline{t}$.

Now we will write a formula in the symbol set

$$\{<, E, T, H, Q, D, 0, 1, b, l, r, s, q_1, \ldots, q_r, \text{“}yes\text{”}\}$$

which checks whether the relations T, D, Q and H code a run on the machine M. Suppose that constant symbols in the symbol set above are interpreted by different elements. Let $A = \{0, \ldots, m-1\}$ and suppose, without loss of generality, that $<$ is the natural order on A, and that q_r is the accepting state. We can write a formula ϕ_0 which describes the initial configuration of M. Let $bin(\mathfrak{A})(i)$ be the symbol in the position i of $bin(\mathfrak{A})$. Let $\overline{0}_l = 0 \ldots 0$ be an l-tuple. We define ϕ_0 as:

$$\begin{aligned}
\phi_0 \quad = \quad & \forall xy(Exy \to T\overline{0}_{m^k}\overline{0}_{m^k-2}xy1) \wedge \\
& \forall xy(\neg Exy \to T\overline{0}_{m^k}\overline{0}_{m^k-2}xy0) \wedge \\
& \forall x_1 \ldots \forall x_{m^k}(x_1 \ldots x_{m^k-2} \neq \overline{0}_{m^k-2} \to R\overline{0}_{m^k}x_1 \ldots x_{m^k}b) \wedge \\
& H\overline{0}_{m^k}\overline{0}_{m^k} \wedge \\
& Q\overline{0}_{m^k}q_0.
\end{aligned}$$

The formula above says that, in the first m^2 positions of the tape at time zero we have the binary representation of $\mathfrak{A}$ (recall that $\mathfrak{A}$ is an $\{E\}$-structure) and that there are blanks in the remaining positions. It also says that the head starts at the beginning of the tape and that the initial state is q_0.

We now should write a formula that, given a configuration of the machine at time $\overline{t}$ says what is the configuration of the machine after the next step.

Let $\bar{t}, \overline{t+1}, \bar{p}, \overline{p+1}, \overline{p-1}$ be m^k-tuples of variables. The first two are intended to represent a point in the time, and the last three are intended to represent a position in the tape. We can write a first-order formula $succ_{m^k}(\bar{t}, \overline{t+1})$ which says that $\overline{t+1}$ (resp. $\overline{p+1}$) is the successor of $\bar{t}$ (resp. $\bar{p}$) with respect to the lexicographic ordering on m^k-tuples induced by $<$ (if $\overline{t+1}$ represent the least element with respect to $<^{m^k}$, then $\bar{t}$ and $\overline{t+1}$ represent the same element). The following formula says that $\bar{t}$ and $\overline{t+1}$ (resp. $\overline{p-1}$, $\bar{p}$ and $\overline{p+1}$) represent consecutive points in time (resp. consecutive places on the tape):

$$\theta = succ_{m^k}(\bar{t}, \overline{t+1}) \wedge succ_{m^k}(\bar{p}, \overline{p+1}) \wedge succ_{m^k}(\overline{p-1}, \bar{p}).$$

We can suppose that for each pair symbol-state there are exactly 2 transitions (since the machine is nondeterministic). For each pair of transitions $\alpha = (s, q, d', s', q')$ and $\alpha' = (s, q, d'', s'', q'')$ we have a formula that controls the tape, the state of the machine and the direction to which the head moved, in the case one of these transitions is chosen:

$$\begin{aligned} TQD_{\alpha\alpha'} \quad = \quad & Q\bar{t}q \wedge H\bar{t}\,\bar{p} \wedge T\bar{t}\,\bar{p}s \rightarrow \\ & \big((T\overline{t+1}\,\bar{p}s' \wedge Q\overline{t+1}q' \wedge D\overline{t+1}d')\vee \\ & (T\overline{t+1}\,\bar{p}s'' \wedge Q\overline{t+1}q'' \wedge D\overline{t+1}d'')\big) \end{aligned}$$

For each such pair of transitions we need the formula above. For the entire transition table we have:

$$TQD = \bigwedge_{\alpha,\alpha' \in \delta} (TQD_{\alpha\alpha'})$$

The following formula controls the move of the head:

$$Head = D\bar{t}r \rightarrow H\bar{t}\,\overline{p+1} \wedge D\bar{t}l \rightarrow H\bar{t}\,\overline{p-1} \wedge D\bar{t}s \rightarrow H\bar{t}\,\bar{p}$$

The following formula checks whether the run is an accepting run:

$$acc = \exists \bar{t} Q\bar{t}\text{“}yes\text{”}.$$

Let $ord(<)$ be a first-order formula that says that $<$ is a linear order. Now, we just need to make a conjunction of the formulas defined above and

existentially quantifying de new symbols introduced in the proof, namely $\{<, T, H, Q, D, 0, 1, b, l, r, s, q_1, \ldots, q_r, \text{“}yes\text{”}\}$, just the symbol E lasts unbounded by a quantifier:

$$\exists < \exists T \exists H \exists Q \exists D \exists 0 \exists 1 \exists b \exists l \exists r \exists s \exists \overline{q} \exists \text{“}yes\text{”} \forall \overline{t} \forall \overline{t+1} \forall \overline{p-1} \forall \overline{p} \forall \overline{p+1} ($$

$$ord(<) \wedge \theta \wedge \phi_0 \wedge TQD \wedge Head \wedge acc)$$

This show that any query computable by a nondeterministic Turing machine in polynomial time can be expressed by an existential second-order sentence. For the converse, i.e., to show that we can evaluate a second-order existential sentence on a structure, it is sufficient to observe that we can use a nondeterministic Turing machine to guess the values of the quantified variables and check the first-order part of the formula in polynomial time, since the size of the relations are polynomial.

Theorem 1.1 (Fagin's Theorem) *$\exists SO = NP$.*

Fagin's Theorem can be generalized to the Polynomial Hierarchy. In particular, we have:

Theorem 1.2 ([Imm99]) *Let $\mathcal{G}$ be a graph property in the polynomial hierarchy. Then there is a second-order sentence ϕ (not necessarily existential) in the language of graphs such that $G \in \mathcal{G}$ iff $G \models \phi$.*

In the following chapter, we will present the first results of this work, namely on the expressiveness of nonmonotonic logics using the approach of Abstract Model Theory.

Chapter 2

Abstract Preferential Logics

In this chapter, we introduce expressible preferential logics, whose preference relations can be defined by some abstract logic. Our approach follows that of Abstract Model Theory. We show that some well-known nonmonotonic logics are preferential. We prove that they are elementary, which means that their preference relation can be defined in first-order logic. We study expressiveness and definability results for wide classes of abstract preferential logics in the spirit of Universal Logic. We present a collapse result for expressible preferential logics. We prove that, for a class of expressible preferential logics, if the class of minimal models of a finite set of sentences is Δ-$\mathcal{L}$-expressible, then it is $\mathcal{L}$-expressible, that is, such class of models can be finitely axiomatized in $\mathcal{L}$. Using this result, we show that under certain conditions one can axiomatize the class $\mathbb{C}$ of minimal models of a finite set of sentences where some symbol P is defined using that set and an explicit definition for this symbol. The results obtained generalize some results presented in [FM11b] and were published in [FM11a].

2.1 Introduction

In practical situations, people reason and act without complete or sufficient knowledge about the situation that they are dealing with. Sometimes, there is no way or it is too much expensive to obtain all the necessary information in order to be secure about our conclusions. However, in such cases, we may find ourselves in a position in which it is mandatory to take some action or make some inferences. For instance, a pilot which need to abort landing the airplane

for some technical problem just discovered will immediately go-around instead of waiting for the confirmation that this action will not lead to a crash with another aircraft. Reasoning under such circumstances is, thus, required. Any system intended to describe or simulate practical reasoning must be able to deal with the lack of information. Actually, even in daily activities we assume beliefs which we take as certain, but for which there is no logical, deductive justification [Hum07]. However, although we cannot deduce from earlier facts that to put ours hands on the fire will burn them, no mentally healthy person will do this and think that nothing will go wrong.

The lack of information and the need for drawing conclusions force us to go beyond of which can be deduced from our partial, current knowledge. In real life, we make use of general or uncertain knowledge, such as that "birds generally flies" and that penguins do not, to guide us in the task of making such assumptions. However, due to the character of uncertainty of such sort of assumption, it may be confronted with new, reliable information and be refuted. Hence, besides the ability of handling the lack of information and the use of general and uncertain knowledge, another feature of a system that models practical reasoning is to be able to backtrack and drop some conclusions previously inferred from uncertain assumptions in order to stay consistent with the new information obtained.

In order to formalize practical reasoning under partial knowledge, classical logic is not appropriate. In fact, classical logic is deductive and, as a deductive approach to reasoning, classical logic cannot go beyond and infer more than what is already known. As we argued above, in practical situations one must conclude more than what can be deduced, that is, a logical system for practical reasoning must allow non-deductive inferences. Another feature of classical reasoning is the monotonicity property: the addition of new premises does not invalidate previous inferences. On the contrary, logical systems for practical reasoning must be nonmonotonic, since conclusions taken under partial knowledge may be defeated. Hence, classical logic is not suited to properly deal with practical reasoning.

In the 1980s, with the increasing interest in Artificial Intelligence, some nonmonotonic logical systems where proposed to formalize practical reasoning, such as Reiter's Default logic [Rei80], Doyle and McDermott's Nonmonotonic logic [MD80] and McCarthy's Circumscription [McC80, McC86]. Our focus here is in the latter approach since we are interested in minimal models, as we explain below.

McCarthy's *Predicate Circumscription* is one of the most studied logical approaches to nonmonotonic reasoning (see [Lif94] for a good introduction and extensive bibliography). In [McC80], McCarthy introduces Circumscription to deal with the *Qualification Problem*. The Qualification Problem is the problem of describing or qualifying the necessary conditions to take an action or safely infer some information in a given situation. As argued in [McC80], it is practically impossible to deal with the huge amount of constraints or conditions necessary for the success of an action. In practice, people disregard many possible obstacles or assumptions contrary to some conclusion just because they are unknown, there is no evidence for it, or they usually do not happen. Usually, we concentrate on the relevant evidences drawing conclusions even in the absence of information about conditions of success. We also suppose, under the lack of knowledge about the consequences of some action, that things not directly involved in an action will stand as they were before the action was taken. McCarthy also argued that if, on the one hand, we must go beyond what can be deduced from the known information, on the other hand we should avoid unreasonable assumptions. That is the case, for instance, of the winged horse solution for McCarthy's Missionaries and Cannibals puzzle [McC80]. General or default knowledge is used as an heuristics, a guide to reasoning.

The intuitive idea of Predicate Circumscription is to consider that the objects which have some property are only those necessary to satisfy the problem description. In [McC80], Predicate Circumscription was introduced as a first-order formula schema and, in [McC86], as a second-order formula, with an additional extension called *Formula Circumscription*. Here, we will call them first-order Circumscription and second-order Circumscription, respectively. Predicate Circumscription works minimizing the extent of some relation in the problem description. A variant called *Parallel Circumscription* works minimizing a tuple of relations at the same time. In a model of a circumscribed theory, the circumscribed relations are intended to have their extents as minimal as possible and satisfying the theory. Such models are called *minimal models*.

A general approach to nonmonotonic logic was given by Gabbay [Gab85]. He gave a characterization of nonmonotonic consequence relation similar to the characterization of deductive consequence given by Tarski, which can be seen as axioms for logical systems in the following sense.

A logic, or logical system, is usually regarded as a pair $\mathcal{L} = (L, \vdash)$ where $\vdash \subseteq \wp(L) \times L$ and $\Gamma \vdash \phi$ is intended to mean that ϕ can be inferred by Γ. Tarski defined a deductive logical system as one in which $\vdash$ has reflexivity, transitivity

and monotonicity, namely:

(Reflexivity) $\Gamma \vdash \phi$, for each $\phi \in \Gamma$.

(Transitivity) If $\Gamma \cup \{\phi\} \vdash \psi$ and $\Gamma \cup \{\psi\} \vdash \theta$ then $\Gamma \cup \{\phi\} \vdash \theta$.

(Monotonicity) If $\Gamma \vdash \phi$ then $\Gamma \cup \Delta \vdash \phi$.

Gabbay was interested in defining the properties a nonmonotonic consequence relation should satisfy [Gab85]. Akin to Tarski's characterization of deductive consequence relation, Gabbay defines reflexivity, transitivity and cautious monotonicity (see [Gab85, KLM90][1]) as the properties a "good" nonmonotonic logic should have. Cautious monotonicity is defined as follows, where $\mathrel{|\!\sim}$ represents a nonmonotonic inference relation:

(Cautious Monotonicity) If $\Gamma \mathrel{|\!\sim} \alpha$ and $\Gamma \mathrel{|\!\sim} \phi$ then $\Gamma \cup \{\alpha\} \mathrel{|\!\sim} \phi$.

Another approach to nonmonotonic reasoning is Shoham's preferential models. Shoham studied nonmonotonic logics from the semantical point of view. Shoham introduced the notion of preferred model by considering an order (a strict partial order) relation over the models used in the semantics of some logic [Sho87]. Using the preference relation, a new nonmonotonic consequence relation is defined. This approach generalizes Circumscription since in principle any pre-order on structures can be regarded. Gabbay's and Shoham's approaches were further investigated in [KLM90], which introduced the cumulative models, a generalization of Shoham's preferential models, and showed that such cumulative models can be used to give semantics to propositional logics with nonmonotonic consequence relation satisfying Gabbay's properties.

We are interested in the study of preferential logical systems from the semantical standpoint, that is, through the study of preferential models. Here, we will be specially concerned with preferential semantics in the sense of Shoham. We begin with a triple $\mathcal{L} = (L, \mathbb{M}, \models)$ where L is a set (of sentences), $\mathbb{M}$ is a class of interpretations for L and $\models$ is a satisfiability relation between elements in $\mathbb{M}$ and L. We call $\mathcal{L}$ an abstract logic [BF85, CK73, GMV07]. In order to obtain a preference semantics for L, we need only a preference relation $\prec$ between the elements of $\mathbb{M}$. The quadruple $\mathcal{L}' = (L, \mathbb{M}, \models, \prec)$ where $\mathcal{L} = (L, \mathbb{M}, \models)$ is an abstract logic and $\prec$ is a preferential relation between the elements of $\mathbb{M}$

[1] Gabbay uses the term "restricted monotonicity" instead of "cautious monotonicity" due to [KLM90].

defines what we call an abstract preferential logic. Additionally, if $\prec$ can be expressed by some preferential logic, then $\mathcal{L}'$ is an expressible preferential logic. The expressiveness of an abstract preferential logic can be examined from two perspectives: the classes of structures in $\mathbb{M}$ which can be expressed by sentences in L with respect to $\models$, and the classes of structures which are minimal models of sentences in L with respect to $\prec$. Also, from an abstract preferential logic, we can define two possibly different consequence relations: One defined using all models in $\mathbb{M}$ and another using only minimal models with respect to $\prec$. The last one can be nonmonotonic.

We are particularly interested in analysing the expressiveness and definability properties of abstract preferential logics with expressible preference relation. Our focus is in their expressive power with respect to minimal models and its relationship with expressible classes of (not necessarily minimal) models (see Definition 2.11). In this work, we aim to give general proofs which hold for any expressible preferential logic in large classes, in the spirit of Universal Logic [Bez07, Bez06].

Universal Logic is intended to be a general theory of logics. One of the goals of Universal Logic is to generalize important logical properties, like compactness, Lowenheim-Skolem, etc., by establishing necessary and sufficient conditions for logics to hold these properties. Some of the results presented here generalize others presented in [FM11b] for the particular case of Circumscription to wide classes of expressible preferential logics. To achieve this, we will follow the approach of Abstract Model Theory (see for instance [BF85, GMV07]).

This chapter is divided as follows: In Section 2.2, we will introduce the basic concepts regarding Abstract Model Theory, preference relations and minimal models. In Section 2.3, we will give some examples of known logical systems which can be characterized by expressible preferential logics. We will show that McCarthy's Circumscription and Reiter's normal default rules can be seen as elementary preferential logics, which means that they are expressible preferential logics whose preference relations can be expressed in first-order logic. In Sections 2.4 and 2.5 we will present our main contributions. In Section 2.4, we will give an example of the expressive power of an expressible preferential logic with respect to minimal models. We will use a well known preferential logic, namely McCarthy's Circumscription, to give an example of preferential logic $\mathcal{L}$ in which there are classes of minimal models of finite sets of sentences which cannot be expressed, in the sense of Definition 2.11, even by infinite sets of sentences in $\mathcal{L}$. After that, we will show that, if an abstract preferential logic $\mathcal{L}$ has some key

properties, whenever the class of minimal models of a finite theory in $\mathcal{L}$ coincides with the class of models (minimal or not) of an infinite set of sentences in $\mathcal{L}$, then it coincides with the class of models (minimal or not) of a finite one. That is, those properties guarantee that the class of minimal models of a finite theory is finitely axiomatizable in $\mathcal{L}$ or it is not even axiomatizable by infinite theories in $\mathcal{L}$. As a consequence, there is a gap in the possible classes of minimal models of sentences in $\mathcal{L}$, as depicted in Figure 2.1. In Section 2.5, we will give an application of the results in Section 2.4 to the definability theory. We will consider the cases where some relation symbol, say R, is defined in the class $\mathbb{C}$ of minimal models of some finite theory T in $\mathcal{L}$. The only restriction is that we need that the models of T where R is empty, if any, are all minimal. We will show how the results of Section 2.4 permit us to axiomatize $\mathbb{C}$ in terms of the initial theory T and an explicit definition for the defined relation symbol.

2.2 Expressible Preferential Logics

In this section, we will present the definitions upon which our theory will be developed. We will recall the concept of abstract logic [BF85, CK73, GMV07] and introduce new concepts to deal with preference relations. We will avoid, as much as possible, to say what are the sentences of our logical systems since we want that our results do hold for a variety of logics. Such logics will be characterized by the properties held by their satisfaction relations. Even without making the structure of our sentences explicit, we will be able to describe the semantical properties of the sentences with respect to the satisfiability relation, a central basic concept in this work.

We begin with the definition of logical system used here.

Definition 2.1 (Language, Consequence and Logical System) *We consider that a* language L *is a set whose elements are called* sentences. *A* consequence relation $\vdash$ *is a subset of* $\wp(L) \times L$. *A* logical system *(or* logical structure [Bez07]*) is a pair* $\mathcal{L} = (L, \vdash)$.

Although we have used the terms "language" and "sentence" in the definition above, we do not make any linguistic or syntactical assumption about these things. That is, L can be any set.

Many logical systems studied in logic have semantic characterizations. In Abstract Model Theory, the central concept is that of abstract logic. Logical

systems are studied in Abstract Model Theory through the concept of abstract logic [BF85, CK73, GMV07]. Abstract logics are used, for instance, in the proof of Lindström Theorems [Lin69, EFT94a][2].

Definition 2.2 (Abstract Logic) *An* abstract logic *is a triple* $(L, \mathbb{M}, \models)$ *with a language* L*, a class* $\mathbb{M}$ *whose elements are called* interpretations for L *or* L-interpretations*, and a satisfiability relation* $\models$ *between* L*-interpretations and sentences in* L*, that is, a subset of* $\mathbb{M} \times L$*. If* Γ *is a set of sentences and* $\mathfrak{I}$ *is an interpretation, then* $\mathfrak{I} \models \Gamma$ *iff* $\mathfrak{I} \models \phi$ *for each* ϕ *in* Γ*.*

Abstract logics are essential to our work, mainly the abstract preferential logics introduced below. From this point of view, the main relation is the satisfiability relation, from which many properties of the considered logical systems can be defined.

From an abstract logic, we can define a standard consequence relation.

Definition 2.3 (Standard Consequence Relation) *Let* $\mathcal{L} = (L, \mathbb{M}, \models)$ *be an abstract logic. We define a* standard consequence relation $\vdash_{\mathcal{L}}$ *as*

$$\Gamma \vdash_{\mathcal{L}} \phi \text{ iff, for all } \mathfrak{I} \in \mathbb{M}, \text{ if } \mathfrak{I} \models \Gamma \text{ then } \mathfrak{I} \models \phi.$$

We omit the subscript $\mathcal{L}$ *when it is clear from the context.*

The symbol $\vdash$ is commonly used to represent consequence relations syntactically defined, that is, by means of a deductive system. Here, this symbol is used to denote any sort of consequence relation, as in Definition 2.1.

Note that due to the definition of $\mathfrak{I} \models \Gamma$ given by Definition 2.2, the standard consequence relation defined in Definition 2.3 above is deductive in the sense of Tarski.

In order to investigate preferential logics following the approach of Abstract Model Theory, we will extend the concept of abstract logic.

Definition 2.4 (Abstract Preferential Logic) *An* abstract preferential logic *is a quadruple* $\mathcal{L} = (L, \mathbb{M}, \models, \prec)$ *where* $(L, \mathbb{M}, \models)$ *is an abstract logic and* $\prec$ *is a binary relation between elements in* $\mathbb{M}$*.*

The binary relation $\prec$ on $\mathbb{M}$ represents the preference between the interpretations in $\mathbb{M}$. Our concept of abstract preferential logic generalizes Shoham's

[2] In [EFT94a], abstract logics are simply called *logical systems*, a term used here with another meaning as presented in Definition 2.1.

concept of preferential logic since we allow any sort of binary relation between interpretations as a preference relation, while Shoham's definition considered only strict partial orders [Sho87].

Sometimes, we can select the "most preferred" among some (or all) interpretations in $\mathbb{M}$. Such interpretations correspond to the concept of minimal elements of a binary relation.

Definition 2.5 (Minimal Element and Minimal Models) *Let $\prec$ be a binary relation on a class $\mathbb{C}$. Let $\mathbb{C}'$ be contained in or equal to $\mathbb{C}$. An element $\mathfrak{C}$ in $\mathbb{C}'$ is a* $\prec$-minimal element *of $\mathbb{C}'$ iff there is no element $\mathfrak{B}$ in $\mathbb{C}'$ such that $\mathfrak{B} \prec \mathfrak{C}$ but $\mathfrak{C} \not\prec \mathfrak{B}$.*

Let $(L, \mathbb{M}, \models, \prec)$ be an abstract preferential logic and let Γ be a set of elements of L. We say that $\mathfrak{I}$ in $\mathbb{M}$ is a $\prec$-minimal model *of Γ iff $\mathfrak{I}$ is a $\prec$-minimal element of $\{\mathfrak{I} \in \mathbb{M} | \mathfrak{I} \models \Gamma\}$.*

From an abstract preferential logic, we can define a (possibly nonmonotonic) consequence relation and, hence, a (possibly nonmonotonic) logical system.

Definition 2.6 (Preferential Consequence and Logical System) *Let $\mathcal{L} = (L, \mathbb{M}, \models, \prec)$ be an abstract preferential logic. We define a* preferential consequence relation $\mid\!\sim_{\mathcal{L}}$ *between sets of elements in L and elements in L as*

$$\Gamma \mid\!\sim_{\mathcal{L}} \phi \text{ iff, for each } \prec\text{-minimal model } \mathfrak{I} \text{ of } \Gamma,\ \mathfrak{I} \models \phi.$$

We call the structure $\mathcal{L}' = (L, \mid\!\sim_{\mathcal{L}})$ a preferential logical system.

If $\prec$ is a strict partial order, the consequence relation $\mid\!\sim_{\mathcal{L}}$ in the definition above is exactly the kind of nonmonotonic consequence relation considered in [Sho87]. It must be noted that from an abstract preferential logic $\mathcal{L} = (L, \mathbb{M}, \models, \prec)$ we can define two consequence relations: a preferential consequence relation, as presented in Definition 2.6 above, and a standard one, defined as in Definition 2.3 with respect to the abstract logic $(L, \mathbb{M}, \models)$.

As one can see, the concepts defined above do not depend on the structure of L or on what are the elements in $\mathbb{M}$. Here, we will be concerned with abstract preferential logics which have structures as the interpretations for their languages, that is, such that the class $\mathbb{M}$ is a class of relational structures.

Since structures will be the basic interpretations for us, the following assumptions for languages and satisfiability relations will be useful. Some of the properties below are used in [EFT94a, Chapter XIII].

Definition 2.7 (Languages and Satisfiability Revisited) *A language (intended to be interpreted by structures) is a set L of sentences such that, for each symbol set S, L^S is a subset of L whose elements are called the S-sentences of L and such that*

- *if $S_0 \subseteq S_1$, then $L^{S_0} \subseteq L^{S_1}$.*

A satisfiability relation for L is a binary relation $\models$ between S-structures and sentences in L^S with the following properties:

- *(Isomorphism Property) if $\mathfrak{A} \models \phi$ and $\mathfrak{B} \cong \mathfrak{A}$ then $\mathfrak{B} \models \phi$;*
- *(Reduct Property) if $\phi \in L^{S_0}$, $\mathfrak{A}$ is an S-structure and $S_0 \subseteq S$, then $\mathfrak{A} \models \phi$ iff $\mathfrak{A}|_{S_0} \models \phi$;*
- *(Symbol Invariance Property) if $f : S \sim S'$ and $\phi \in L^S$ then there is a $\phi[S' \backslash S] \in L^{S'}$ such that, if $\mathfrak{A} = (A, \sigma)$ is an S-structure, then $\mathfrak{A} \models \phi$ iff $\mathfrak{A}_f \models \phi[S' \backslash S]$.*

Hereafter, the languages and satisfaction relations used will have the properties in Definition 2.7. We call $Mod_S^{\mathcal{L}}(\Gamma)$ the class of S-structures which are models of the sentences in the set Γ of S-sentences. Given a class of S-structures $\mathbb{C}$, we call $\mathrm{Th}_S^{\mathcal{L}}(\mathbb{C})$ the set of S-sentences in $\mathcal{L}$ satisfied by every S-structure in $\mathbb{C}$.

Now, we will introduce the central concept of this chapter: $\mathcal{L}$-expressible preference logics defined over an $\mathcal{L}$-expressible preference relation. Intuitively, an $\mathcal{L}$-expressible preference relation is a collection of pairs of structures defined by an $\mathcal{L}$-sentence in a suitable way. Given a symbol set S, we define the relation $\prec^{\phi(S,S')}$ through a defining sentence $\phi(S, S')$. S' is a symbol set similar to S and S and S' are disjoint. To be precise, if

$$S = \{R_1, \ldots, R_n, f_1, \ldots, f_m, c_1, \ldots, c_k\},$$

then

$$S' = \{R'_1, \ldots, R'_n, f'_1, \ldots, f'_m, c'_1, \ldots, c'_k\},$$

if $s \in S$ has arity n then so is s', and $S \cap S' = \emptyset$. We will also allow two new unary symbols A and A' to appear in $\phi(S, S')$. The sentence $\phi(S, S')$ is in fact an $S \cup S' \cup \{A, A'\}$-sentence. The idea is that from an $S \cup S' \cup \{A, A'\}$-structure $\mathfrak{A}''$ we can define two structures, namely, a substructure $\mathfrak{A}$ of the S-reduct of

$\mathfrak{A}''$ and a substructure $\mathfrak{A}'$ of the S'-reduct of $\mathfrak{A}''$. The domains of $\mathfrak{A}$ and $\mathfrak{A}'$ are defined by the interpretations $A^{\mathfrak{A}''}$ and $A'^{\mathfrak{A}''}$ of the monadic predicate symbols A and A' in the $S \cup S' \cup \{A, A'\}$-structure $\mathfrak{A}''$. It can be easily seen that $\mathfrak{A}''$ is a common expension of $\mathfrak{A}$ and $\mathfrak{A}'$. As it was previously remarked, not always a subset of the domain of a structure is the domain of a substructure or a subduct. In this way, we will consider only those pairs which are subducts of an $S \cup S' \cup \{A, A'\}$-structure. In particular, if S is composed only by relation symbols, then the subducts are always well defined.

Definition 2.8 (Expressible Preference Relation) *Let S and S' be symbol sets such that $f : S \sim S'$, for some f, and $\mathcal{L}$ be an abstract logic as defined above. Let $\phi(S, S')$ be an $S \cup S' \cup \{A, A'\}$-sentence of $\mathcal{L}$ such that, if $\mathfrak{A}'' \models \phi(S, S')$ then there is $\mathfrak{A} = \mathfrak{A}''|_S^{A^{\mathfrak{A}''}}$ and $\mathfrak{A}'$ such that $\mathfrak{A}'_f = \mathfrak{A}''|_{S'}^{A'^{\mathfrak{A}''}}$. We define the relation $\prec^{\phi(S,S')}$ between S-structures as*

$$\mathfrak{A} \prec^{\phi(S,S')} \mathfrak{A}'$$

iff

there is $\mathfrak{A}''$ such that $\mathfrak{A}'' \models \phi(S, S')$ and $\mathfrak{A} = \mathfrak{A}''|_S^{A^{\mathfrak{A}''}}$ and $\mathfrak{A}'_f = \mathfrak{A}''|_{S'}^{A'^{\mathfrak{A}''}}$.

*$\prec$ is an $\mathcal{L}$-*expressible preference relation *iff $\prec = \prec^{\phi(S,S')}$ for some sentence $\phi(S, S')$ in $\mathcal{L}$. If $\mathcal{L}$ is first-order logic, we call $\prec^{\phi(S,S')}$ an* elementary preference relation.

Note that the relation $\prec^{\phi(S,S')}$ may not be symmetric. The structure $\mathfrak{A}''$ can be thought of as comprising two S-structures $\mathfrak{A}$ and $\mathfrak{A}'$, but one of them is represented in $\mathfrak{A}$ using symbols in S'. Then we can refer to the relations of $\mathfrak{A}'$ in $\mathfrak{A}$ using the symbols in S and to the relations of $\mathfrak{A}''$ in $\mathfrak{A}$ using the symbols in S'.

Definition 2.9 (Expressible Preferential Logic) *We call an abstract preferential logic $\mathcal{L}' = (L', \mathbb{M}', \models', \prec)$ an* $\mathcal{L}$-expressible abstract preferential logic *(*$\mathcal{L}$-expressible preferential logic*, for short) iff $\prec$ is an $\mathcal{L}$-expressible preference relation. We omit $\mathcal{L}$- when $\mathcal{L}$ is clear from the context or when we want to say that a preference relation or an abstract preferential logic is $\mathcal{L}$-expressible for some $\mathcal{L}$. If $\prec$ is an elementary preference relation, then $\mathcal{L}'$ is an* elementary preferential logic*.*

In the next section, we will give some examples of known nonmonotonic logics which can be presented as abstract preferential logics and we will show that they are in fact elementary preferential logics.

2.3 Examples of Elementary Preferential Logics

In this section, we shall exhibit two examples of nonmonotonic logics which can be presented as expressible preferential logics: McCarthy Predicate Circumscription and Reiter's Normal Default Logic. We will show that they are *elementary preferential logics*, that is, abstract preferential logics $\mathcal{L} = (L, \mathbb{M}, \models, \prec)$ where the preference relation $\prec$ is FO-expressible (see Definition 2.9).

Example 2.1 (Predicate Circumscription) Consider the following known problem. Suppose we want to axiomatize the facts that: i) generally birds fly, ii) penguins are birds which do not fly and iii) Tweety is a bird. The last two can be easily described by the following first-order sentences in the symbol set $\{bird, penguin, fly, Tweety\}$:

$$\forall x(penguin(x) \rightarrow bird(x)) \wedge \forall x(penguin(x) \rightarrow \neg fly(x)), \tag{2.1}$$

$$bird(Tweety). \tag{2.2}$$

The proposition i), however, involves the concept of "generally fly." This is a vague information, since it says something about the class of birds, but does not say how we can apply it to particular birds. In such a situation, for instance, one would prefer to infer, on the absence of information about Tweety being a penguin or not, that Tweety is a "general bird" and that it is plausible that it flies. The way Circumscription deals with this problem is through the so-called *abnormality theories*. A new predicate, say *abnormal*, is introduced in the theory in order to deal with the exceptions to the general information, in this case that birds generally fly. For instance, i) is represented by the following sentence:

$$\forall x(bird(x) \wedge \neg abnormal(x) \rightarrow fly(x)) \tag{2.3}$$

and, to the sentence above, it should be added the following *circumscription policy*:

$$\mathbf{circ}\ abnormal\ \mathbf{var} fly. \tag{2.4}$$

Let T be the set composed by the sentences (2.1), (2.2) and (2.3) above. As a set of first-order sentences, T has some models where $fly(Tweety)$ holds and some where it does not hold. The circumscriptive policy (2.4) indicates that we are

interested in those models of T which have the least possible interpretation for the *abnormality* predicate symbol which satisfies the sentences in T, and that the interpretation of fly is allowed to vary in order to get a smaller interpretation for *abnormality* (see [Lif94]). In this case, the least possible interpretation for *abnormal* in T is the empty set, but in this case the interpretation of fly must include $Tweety$. Hence, from T and the circumscriptive policy (2.4), we infer $fly(Tweety)$ by circumscription. One can see that, if a new information is added that Tweety is a penguin, which corresponds to the sentence

$$penguin(Tweety),$$

the least possible interpretation for *abnormal* is no longer the empty set and hence we cannot infer $fly(Tweety)$ any more.

In general, an abnormality theory is a pair $\langle T, C\rangle$ where T is a finite set of first-order sentences in a symbol set $S \cup \{abnormal_1, abnormal_2, \dots\}$ and C is a set of circumscription policies. For the sake of simplicity, we will consider C a singleton.

McCarthy [McC86] defined Circumscription in the following way. Let S be a symbol set, and let P and $\overline{Z} = Z_1, \dots, Z_n$ be relation symbols. Given the circumscription policy **circ** P **var** $\overline{Z}$ we define the consequence relation $\vdash_{Circ}^{P;\overline{Z}}$ between finite sets $T(P, \overline{Z})$ of $S \cup \{P, \overline{Z}\}$-sentences and $S \cup \{P, \overline{Z}\}$-sentences as[3] (see [McC86]):

$$\begin{gathered} T(P,\overline{Z}) \vdash_{Circ}^{P;\overline{Z}} \phi \\ \text{iff} \\ T(P,\overline{Z}) \wedge \forall P' \forall \overline{Z'}(P' \subsetneq P \rightarrow \neg T(P', \overline{Z'})) \vdash_{SO} \phi, \end{gathered} \tag{2.5}$$

where $\vdash_{SO}$ is the second-order logic consequence relation. The consequence relation $\vdash_{Circ}^{P;\overline{Z}}$ corresponds to the circumscription policy **circ** P **var** $\overline{Z}$ applied to $T(P, \overline{Z})$. Hence, Circumscription can be understood as the class of logical systems $(L, \vdash_{Circ}^{P;\overline{Z}})$ where L is the language of first-order logic and $\vdash_{Circ}^{P;\overline{Z}}$ is as defined above.

The second-order sentence

$$Circ[T(P,\overline{Z}); P; \overline{Z})] = T(P,\overline{Z}) \wedge \forall P' \forall \overline{Z'}(P' \subsetneq P \rightarrow \neg T(P', \overline{Z'})) \tag{2.6}$$

[3] In the symbol $\vdash_{Circ}^{P;\overline{Z}}$, we use a semicolon to distinguish from the circumscribed predicate P and the varied predicates $\overline{Z}$. We use comma to separate elements in a tuple.

in (2.5) means that there is no P' strictly included in P which satisfies T, no matter what is the interpretation given to $\overline{Z'}$.

The consequence relation $\vdash_{Circ}^{P;\overline{Z}}$ has a well known characterization as a preferential consequence relation. That is, instead of characterizing Circumscription as the class of logical systems like $(L, \vdash_{Circ}^{P;\overline{Z}})$, we may characterize Circumscription as the class of abstract preferential logics like $\mathcal{L}_{Circ} = (L, \mathbb{M}_{P;\overline{Z}}, \models, \leq^{P;\overline{Z}})$, where L is the set of first-order sentences, $\mathbb{M}_{P;\overline{Z}}$ is the class of relational structures on the symbol set $S \cup \{P, \overline{Z}\}$ for each S, and $\models$ is the first-order satisfiability relation restricted to $\mathbb{M}_{P;\overline{Z}}$ and L. The preference relation $\leq^{P;\overline{Z}}$ between $S \cup \{P; \overline{Z}\}$-structures in $\mathbb{M}_{P;\overline{Z}}$ with the same domain and which agree in $S - \{P \cup \overline{Z}\}$ is defined as:

$$\mathfrak{A} \leq^{P;\overline{Z}} \mathfrak{B} \text{ iff } P^{\mathfrak{A}} \subseteq P^{\mathfrak{B}}. \tag{2.7}$$

When $\overline{Z}$ is empty, we write $Circ[T; P]$ and $\leq^P$ for $Circ[T; P; \emptyset]$ and $\leq^{P;\emptyset}$, respectively.

We get the following well known characterization of Circumscription's consequence relation (see, for example, [Lif94]):

Lemma 2.1 $\Gamma \vdash_{Circ}^{P;\overline{Z}} \phi$ *if and only if each model of* Γ *which is minimal with respect to* $\leq^{P;\overline{Z}}$ *is also a model of* ϕ.

The Lemma 2.1 above can be restated in the following equivalent form, according to Definition 2.6, characterizing Circumscription as a preferential logical system:

Corollary 2.1 (Circumscription as an Abstract Preferential Logic) *If*

$$\mathcal{L}_{Circ} = (L, \mathbb{M}_{P;\overline{Z}}, \models, \leq^{P;\overline{Z}}),$$

then

$$\Gamma \mathrel{|\!\sim}_{\mathcal{L}_{Circ}} \phi \text{ iff } \Gamma \vdash_{Circ}^{P;\overline{Z}} \phi.$$

Circumscription, as a logical approach to nonmonotonic inference, can be seen as the class of those abstract preferential logics $\mathcal{L}_{Circ}$ of Corollary 2.1.

Now, we will show that $\leq^{P;\overline{Z}}$ is an elementary preference relation.

Lemma 2.2 (Circumscription as an Elementary Preferential Logic) *The relation* $\leq^{P;\overline{Z}}$ *defined in (2.7) is an elementary preference relation.*

Proof. It is sufficient to show an $S \cup S' \cup \{A, A'\}$-sentence $\phi(S, S')$ of first-order logic, with $\{P, \overline{Z}\} \subseteq S$ and $\{P', \overline{Z'}\} \subseteq S'$, such that

$$\leq^{P;\overline{Z}} = \prec^{\phi(S,S')} . \tag{2.8}$$

For the sake of simplicity, let us restrict ourselves to the case where S is relational. Consider the following sentence:

$$\phi(S, S') = A = A' \wedge (\bigwedge_{s \in S - \{P, \overline{Z}\}} (s = s')) \wedge P \subseteq P'. \tag{2.9}$$

By $s = s'$ in (2.9) we mean a first-order sentence which states that s and s' have the same extension and by $P \subseteq P'$ a first-order sentence which states that the extension of P is contained in the extension of P'. It follows that in any model $\mathfrak{A}$ of $\phi(S, S')$, $A^{\mathfrak{A}} = A'^{\mathfrak{A}}$, $s^{\mathfrak{A}} = s'^{\mathfrak{A}}$ for each $s \in S$, and $P^{\mathfrak{A}} \subset P'^{\mathfrak{A}}$. It is easy to see that

$$\mathfrak{A}|_S^{A^{\mathfrak{A}}} \prec^{\phi(S,S')} \mathfrak{A}|_{S'}^{A'^{\mathfrak{A}}} \text{ iff } \mathfrak{A} \models \phi(S, S') \text{ iff } \mathfrak{A}|_S^{A^{\mathfrak{A}}} \leq^{P;\overline{Z}} \mathfrak{A}|_{S'}^{A'^{\mathfrak{A}}}.$$

It follows that (2.8) holds. ∎

In the next example, we show how Reiter's closed normal default logic without precondition can be presented as an elementary preferential logic.

Example 2.2(Closed Normal Defaults without Preconditions) In [Rei80], Reiter defines the *Default Logic*. A *default rule* (*default* for short) is an expression like

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma},$$

where $\alpha, \beta_1, \ldots, \beta_m$ and γ are first-order S-formulas for some symbol set S, α is the *precondition*, $\beta_1, \ldots, \beta_n$ are the *justifications* and γ is the *conclusion*. The informal meaning of such default is: if we have α and, for each $1 \leq i \leq m$, β_i is consistent with which we have so far, then we can get γ. A *default theory* is a pair $\langle W, D \rangle$ where W is a set of sentences and D is a set of default rules. Associated to a default theory, there is a set of *default extensions*, which are theories that correspond to various, incompatible scenarios, see Definition 2.10 below.

The logical consequences of a default theory is the set of sentences which be-

long to every extension[4]. We can think that any set D of defaults defines a consequence relation $\models_D$ such that

$$W \models_D \phi \text{ iff } \phi \text{ is in all extensions of } \langle W, D\rangle.$$

Hence, Reiter's default logic can be seen as the class of logical systems $(L, \models_D)$ where D is a set of defaults.

A default is said to be *closed* if $\alpha, \beta_1, \ldots, \beta_m, \gamma$ are closed formulas, that is, formulas without free variables. A default theory $\langle W, D\rangle$ is said to be *closed* if each default in D is closed. We precisely define default extension for closed default theories below.

Definition 2.10 (Default Extension) *Let $\langle W, D\rangle$ be a default theory and, for each set of sentences T, let $Th_S^{FO}(T)$ be the set of first-order S-sentences which are logical consequences of T. An extension E of $\langle W, D\rangle$ is a minimal set of first-order sentences such that:*

1. *$W \subseteq E$,*
2. *$Th_S^{FO}(E) = E$ and*
3. *if $E \models \alpha$, $\neg\beta_i \notin E$ for $1 \leq i \leq m$ and $\frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma} \in D$, then $\gamma \in E$.*

A default is *normal* if $m = 1$ and $\beta_1 = \gamma$. Thus normal defaults have the following shape:

$$\frac{\alpha : \beta}{\beta}.$$

We will restrict ourselves to defaults without precondition, that is, defaults of the form:

$$\frac{: \beta}{\beta}.$$

We will show that for finite sets D of closed normal defaults without precondition, the relation $\models_D$ is a preferential consequence relation defined through an elementary preference relation.

In [Poo94], Poole give a proof of the following fact about closed normal default theories without precondition (the version below is slightly different but can be easily obtained from that in [Poo94]):

[4]Here, we will follow the so-called skeptical approach to default reasoning, see [Rei80].

Theorem 2.1 *If all defaults in D are closed, normal and without precondition then the following are equivalent:*

- $W \models \phi$.
- *ϕ holds in all models of W which are minimal with respect to the ordering $\leq^D$ defined as $\mathfrak{A} \leq^D \mathfrak{B}$ iff*

$$\{\frac{:d}{d} \in D | \mathfrak{A} \models \neg d\} \subseteq \{\frac{:d}{d} \in D | \mathfrak{B} \models \neg d\}.$$

We immediately get:

Theorem 2.2 (Default Logic as an Abstract Preferential Logic) *If*

$$\mathcal{L}_D = (L, \mathbb{M}_S, \models, \leq^D),$$

then

$$W \mid\sim_{\mathcal{L}_D} \phi \text{ iff } W \models_D \phi.$$

Now we will show that the $\models_D$ relation is an elementary preferential consequence relation. Let $H = \{\beta | \frac{:\beta}{\beta} \in D\}$ be the set of conclusions of defaults in D. For each set F of formulas, let *co*-$F = \{\neg\alpha | \alpha \in F\}$.

Lemma 2.3 (Default Logic as an Elementary Preferential Logic) *If D is a finite set of normal closed default rules, then $\leq^D$ is an elementary preferential relation.*

Proof. It is sufficient to show an $S \cup S' \cup \{A, A'\}$-sentence $\phi(S, S')$ of first-order logic such that $\leq^D = \prec^{\phi(S,S')}$. Again, for the sake of notational simplicity, let us suppose that the vocabularies are relational. For each $\Gamma \in \wp(H)$, let Γ^A be the set of formulas obtained by the relativization of formulas in Γ to A and $\overline{\Gamma}^A$ the set of formulas obtained by the relativization of formulas in *co*-$(H - \Gamma)$ to A. Note that Γ^A and $\overline{\Gamma}^A$ are sets of $S \cup \{A\}$-sentences. We write $\Gamma^A(S)$ and $\overline{\Gamma}^A(S)$ to stress this fact. Let $\bigwedge(\Gamma^A(S) \cup \overline{\Gamma}^A(S))$ be the conjunction of the formulas in $\Gamma^A(S)$ and $\overline{\Gamma}^A(S)$ which, since H is finite, is an FO-formula. Let $\bigwedge(\Gamma^{A'}(S') \cup \overline{\Gamma}^{A'}(S')$ be obtained by replacing the symbols in $S \cup \{A\}$ which occur in $\bigwedge(\Gamma^A(S) \cup \overline{\Gamma}^A(S))$ with the corresponding in $S' \cup \{A'\}$. Now consider

the formula

$$\phi(S,S') = \bigvee_{\Gamma,\Gamma' \in \wp(H);\Gamma \supseteq \Gamma'} \left(\bigwedge (\Gamma^A(S) \cup \overline{\Gamma}^A(S)) \wedge \bigwedge (\Gamma'^{A'}(S') \cup \overline{\Gamma'}^{A'}(S') \right). \tag{2.10}$$

Let S_e be a symbol set containing S and let $\mathfrak{B}$ and $\mathfrak{B}'$ be S_e-structures in $\mathbb{M}$. If $\mathfrak{B} \prec^{\phi(S,S')} \mathfrak{B}'$, then there is an $S_e \cup S'_e \cup \{A, A'\}$-structure $\mathfrak{A}''$, with $\mathfrak{A}''^A = B$ and $\mathfrak{A}''^{A'} = B'$, which is a common exp*e*nsion to $\mathfrak{B}$ and $\mathfrak{B}'$ such that $\mathfrak{A}''|^B_S = \mathfrak{B}$ and $\mathfrak{A}''|^{B'}_{S'} = \mathfrak{B}'$ and $\mathfrak{A}''$ is a model of $\phi(S,S')$. By definition of $\phi(S,S')$, it follows that

$$\{d \in D | \mathfrak{B} \models \neg d\} \subseteq \{d \in D | \mathfrak{B}' \models \neg d\}.$$

Hence $\mathfrak{B} \leq^D \mathfrak{B}'$. Now suppose that $\mathfrak{B} \leq^D \mathfrak{B}'$. Then

$$\{d \in D | \mathfrak{B} \models \neg d\} \subseteq \{d \in D | \mathfrak{B}' \models \neg d\}.$$

Now, let $\mathfrak{A}''$ be an $S \cup S' \cup \{A, A'\}$-structure which is a common exp*e*nsion to $\mathfrak{B}$ and $\mathfrak{B}'$ such that $\mathfrak{A}''|^B_S = \mathfrak{B}$ and $\mathfrak{A}''|^{B'}_{S'} = \mathfrak{B}'$. It can be easily shown that $\mathfrak{A}'' \models \phi(S,S')$, hence $\mathfrak{B} \prec^{\phi(S,S')} \mathfrak{B}'$. It follows that $\leq^D = \prec^{\phi(S,S')}$, hence $\leq^D$ is elementary. ∎

In this section, we gave two examples of nonmonotonic logics which can be seen as elementary preferential logics. In the next section, we will investigate the expressive power of expressible preferential logics.

2.4 Expressiveness

The expressive power of a logic is its capability to express classes of structures or to distinguish between classes of structures. Our aim in this section is to investigate the expressive power of expressible preferential logics with expressible preference relation with respect to minimal models, that is, given an abstract preferential logic $\mathcal{L} = (L, \mathbb{M}, \models, \prec)$ where $\prec$ is $\mathcal{L}'$-expressible for some $\mathcal{L}' = (L', \mathbb{M}', \models')$, we want to investigate which classes of $\prec$-minimal models can be expressed by sentences in L. We will investigate the case where $\mathcal{L}' = (L, \mathbb{M}, \models)$. We will see that the definability results of Section 2.5 will appear as applications of the results obtained in this section.

Our first aim in this section is to point out the fact that there are elementary preferential logics which lack the Downward Löwenheim-Skolem property with

respect to minimal models. The Downward Löwenheim-Skolem property for an abstract logic $\mathcal{L}$ which admits structures as interpretations or models for its sentences states that, whenever a sentence in $\mathcal{L}$ has a model, then it has a countable one (for first-order logic see [CK73]). This shows that there are expressible preferential logics $\mathcal{L} = (L, \mathbb{M}, \models, \prec)$ with $\mathcal{L}'$-expressible $\prec$ and $\mathcal{L}' = (L, \mathbb{M}, \models)$, where $(L, \mathbb{M}, \models)$ has the Löwenheim-Skolem Property but there are sentences in L whose class of $\prec$-minimal models does not contain a countable model. We will see the case of Predicate Circumscription, which can be seen as an elementary preferential logic (see Section 2.3).

In [Sch87], Schlipf studied decidability questions regarding Circumscription such as to decide whether a first-order formula has a countable minimal model or not. Among other things, Schlipf showed that some formulas which have minimal models do not have countable $\leq^{P,\overline{Z}}$-minimal models [Sch87, Example 2.6]. It means that an analogue to the Downward Löwenheim-Skolem Theorem does not hold for Predicate Circumscription.

Theorem 2.3 (Schlipf, [Sch87]) *There is a first-order formula $\phi(P)$ which has only uncountable P-minimal models.*

Here, P-minimality means minimality with respect to $\leq^P$. Schlipf's proof is based on the existence of "ω_1-like" models of Peano Arithmetics with order, that is, models of cardinality $\aleph_1$ but such that every element has only countable many predecessor with respect to the order relation. In [FM11b], we gave an alternative proof of this fact based on other set theoretical assumption different from those in [Sch87]. Our proof is based on the fact that any two countable dense linear orders without endpoints are isomorphic (see [HJ99]). We then use Circumscription to express a class of structures which have a complete dense linear order without endpoints as substructures. Since there are dense linear orders which are not complete, e.g. the rationals, such substructures cannot be countable. It follows that the models of such circumscription are uncountable.

The failure of Downward Löwenheim-Skolem for Circumscription is particularly important in our characterization of the expressive power of expressible preferential logics since it shows a gap in the expressive power caused by considering the minimal models as explained below.

The semantical approach to preferential logical systems provides a uniform way, through the use of preferential semantics, to compare the expressive power of the deductive and nonmonotonic logical systems by comparing expressible

classes of models and expressible classes of minimal models. For instance, Theorem 2.3 says that we can build elementary preferential logics which are more expressive than first-order logic. In the following, we will investigate the relation between abstract logics $\mathcal{L}$ and expressible preferential logics with $\mathcal{L}$-expressible preferential relations.

Definition 2.11 ($\mathcal{L}$-Expressibility) *Let $\mathcal{L} = (L, \mathbb{M}, \models)$ be an abstract logic. We say that a class $\mathbb{C}$ of structures is $\mathcal{L}$-expressible iff $\mathbb{C}$ is the class of models of some finite set of sentences in L. We say that $\mathbb{C}$ is Δ-$\mathcal{L}$-expressible iff $\mathbb{C}$ is the class of models of some set of sentences in $\mathcal{L}$. If $\mathcal{L}$ is first-order logic, we use the terms elementary and Δ-elementary for FO-expressible and Δ-FO-expressible, respectively.*

As a corollary of Theorem 2.3, we have:

Corollary 2.2 *There are elementary preferential logics $\mathcal{L}$ and classes of minimal models of finite sets of sentences in $\mathcal{L}$, w.r.t. some $\mathcal{L}$-expressible preference relation $\prec$, which are not even Δ-$\mathcal{L}$-expressible.*

It is well known that the collection of elementary classes is strictly included in the collection of Δ-elementary classes. Our next aim is to investigate to what extent the expressive power of preferential logical systems with respect to minimal models is strong enough to distinguish between $\mathcal{L}$-expressible and Δ-$\mathcal{L}$-expressible classes in the following subtle way: Is there an abstract logic $(L, \mathbb{M}, \models)$ and an $\mathcal{L}$-expressible preference relation $\prec$ such that for some finite theory Γ in L the class of $\prec$-minimal elements is Δ-$\mathcal{L}$-expressible but not $\mathcal{L}$-expressible? We address this problem for a class of abstract logics and give a negative answer to this question with respect to this class in Theorem 2.4. First, we will need the following definitions.

Definition 2.12 (Defined Relation) *Let $\mathcal{L} = (L, \mathbb{M}, \models)$ be an abstract logic and let $S \cup \{a_1, \ldots, a_n\}$ be a symbol set. Let $\phi \in L$ be an $S \cup \{a_1, \ldots, a_n\}$-sentence of L and $\mathfrak{A}$ an S-structure. We say that the sentence ϕ defines an n-ary relation $\phi^{a_1,\ldots,a_n,\mathfrak{A}}$ on A as*

$$\phi^{a_1,\ldots,a_n,\mathfrak{A}} = \{(\mathbf{a}_1, \ldots, \mathbf{a}_n) \in A^n | (\mathfrak{A}, \mathbf{a}_1, \ldots, \mathbf{a}_n) \models \phi\}.$$

*We call ϕ an n-*ary relation definiens *(only* definiens *for short). We also use the term "defined relation" applied to such sentence ϕ as reference to the relation it defines on the domain of some structure.*

We will use the following definition to abbreviate some conditions on abstract logics we will need.

Definition 2.13 (Good Logic) *An abstract logic $\mathcal{L} = (L, \mathbb{M}, \models)$, where $\mathbb{M}$ is a class of structures, which is at least as expressive as first-order logic is* good *iff it has the following properties:*

- *(Compactness) for all set Γ of L-sentences, Γ has a model iff any finite subset of Γ has a model (we say that $\mathcal{L}$ is* compact*);*
- *(Partial Relativization) for each L^S sentence ψ and each monadic relation* definiens *ϕ in $L^{S\cup\{a\}}$, there is a sentence ψ^ϕ such that, if $\mathfrak{A}|^{\phi^{a,\mathfrak{A}}}$ is defined, then*
$$\mathfrak{A} \models \psi^\phi \text{ iff } \mathfrak{A}|^{\phi^{a,\mathfrak{A}}} \models \psi.$$
We call ψ^ϕ the relativization of ψ to ϕ *and say that $\mathcal{L}$ has* partial relativization*;*
- *(Boolean Negation) for each L^S sentence ϕ there is a sentence $\neg\phi$ such that*
$$\mathfrak{A} \models \neg\phi \text{ iff } \mathfrak{A} \not\models \phi.$$

The following is the main theorem of this section.

Theorem 2.4 *Let $\mathcal{L}' = (\mathcal{L}, \prec)$ be an $\mathcal{L}$-expressible preferential logic and the abstract logic $\mathcal{L} = (L, \mathbb{M}, \models)$ be a good logic. Let T be a finite set of L-sentences in the symbol set S and $\mathbb{C}$ the class of $\prec$-minimal models of T. If $\mathbb{C}$ is Δ-$\mathcal{L}$-expressible, then $\mathbb{C}$ is $\mathcal{L}$-expressible.*

Proof. Let $\Theta = Th^{\mathcal{L}}_S(\mathbb{C})$ be the $\mathcal{L}$-theory of $\mathbb{C}$, that is, the set of $\mathcal{L}$-sentences which are satisfied by all structures in $\mathbb{C}$. Since $\mathbb{C}$ is Δ-$\mathcal{L}$-expressible, we have

$$Mod^{\mathcal{L}}_S(Th^{\mathcal{L}}_S(\mathbb{C})) = \mathbb{C}.$$

Consider also $T' = T[S'\backslash S]$ for some symbol set S' similar to S (see the symbol invariance property at Definition 2.7). Since $\mathcal{L}$ is at least as expressible as first-order logic, it has sentences δ and δ' equivalent to the first-order atomic sentences $A(c)$ and $A'(c)$ for some constant symbol c, that is, in a $S\cup S'\cup\{A, A'\}$-structure $\mathfrak{B}$, $\delta^{c,\mathfrak{B}} = A^{\mathfrak{B}}$ and $\delta'^{c,\mathfrak{B}} = A'^{\mathfrak{B}}$. Since $\mathcal{L}$ is good, it has partial relativization Θ^δ for the sentences in Θ to the monadic relation *definiens* δ, and

$T'^{\delta'}$ for T' to δ'. As $\prec$ is an $\mathcal{L}$-expressible preferential relation, then there is an $S \cup S' \cup \{A, A'\}$-sentence $\phi(S, S')$ of $\mathcal{L}$ such that $\prec = \prec^{\phi(S,S')}$. Now, consider the set

$$\Phi = \Theta^{\delta} \cup T'^{\delta'} \cup \{\phi(S', S), \neg\phi(S, S'), \psi_S^A, \psi_{S'}^{A'}\}$$

of L-sentences where $\phi(S', S)$ is obtained by simultaneously interchanging the symbols $S \cup \{A\}$ and $S' \cup \{A'\}$ in $\phi(S, S')$, and the sentences ψ_S^A and $\psi_{S'}^{A'}$ are such that, if $\mathfrak{B} \models \psi_S^A$ (respectively $\psi_{S'}^{A'}$) then $\mathfrak{B}|_S^{A^{\mathfrak{B}}}$ (respectively $\mathfrak{B}|_{S'}^{A'^{\mathfrak{B}}}$) is defined, that is, $\mathfrak{B}|_S^{A^{\mathfrak{B}}}$ is an S-structure (respectively $\mathfrak{B}|_{S'}^{A'^{\mathfrak{B}}}$ is an S'-structure). The sentences ψ_S^A and $\psi_{S'}^{A'}$ exist since $\mathcal{L}$ is at least as expressible as first-order logic. Note that these sentences are necessary only if we have function symbols in S.

Suppose Φ has a model $\mathfrak{B}''$ (remember that Φ is an $\mathcal{L}$-theory). As $\mathfrak{B}'' \models \{\psi_S^A, \psi_{S'}^{A'}\}$, we have that $\mathfrak{B}''|_S^{A^{\mathfrak{B}''}}$ and $\mathfrak{B}''|_{S'}^{A'^{\mathfrak{B}''}}$ are defined. Since $\mathfrak{B}'' \models \Theta^{\delta}$, it follows that $\mathfrak{B}''|_{S\cup A} \models \Theta^{\delta}$. Hence, by the partial relativization property, $\mathfrak{B}''|_{S\cup A}^{A^{\mathfrak{B}''}} \models \Theta$. As A does not appear in Θ, then $\mathfrak{B}''|_S^{A^{\mathfrak{B}''}} \models \Theta$. It follows that $\mathfrak{B}''|_S^{A^{\mathfrak{B}''}} \in \mathbb{C}$. That is, $\mathfrak{B}''|_S^{A^{\mathfrak{B}''}}$ is a $\prec$-minimal model of T. Since $\mathfrak{B}'' \models \phi(S', S)$, by definition of $\prec^{\phi(S,S')}$ we have

$$\mathfrak{B}''|_{S'}^{A'^{\mathfrak{B}''}} \prec^{\phi(S,S')} \mathfrak{B}''|_S^{A^{\mathfrak{B}''}}.$$

Similarly, as $\mathfrak{B}'' \models \neg\phi(S, S')$, it follows that

$$\mathfrak{B}''|_S^{A^{\mathfrak{B}''}} \not\prec^{\phi(S,S')} \mathfrak{B}''|_{S'}^{A'^{\mathfrak{B}''}}.$$

But this contradicts the fact that $\mathfrak{B}''|_S^{A^{\mathfrak{B}''}}$ is a $\prec$-minimal model of T. Hence, the set Φ is not satisfiable. Since Φ is an inconsistent set of $\mathcal{L}$ formulas and $\mathcal{L}$ is good and, hence, compact, there is a finite subset Φ_f which is also inconsistent. Let $\Theta'_f = \Phi_f \cap \Theta^{\delta}$ and let Θ_f be the finite subset of Θ whose relativization Θ_f^{δ} to δ is equal to Θ'_f, and consider the set $\Theta_f \cup T$. Obviously, any model of Θ is a model of $\Theta_f \cup T$. Suppose that $\Theta_f \cup T$ has a model $\mathfrak{B}$ not in $\mathbb{C}$. Then $\mathfrak{B}$ is not a $\prec$-minimal model of T which means that, by definition of $\prec$-minimal model, there is a model $\mathfrak{B}'$ of T such that $\mathfrak{B}' \prec \mathfrak{B}$ but $\mathfrak{B} \not\prec \mathfrak{B}'$. Let $\mathfrak{A}''$ be an $S \cup S' \cup \{A, A'\}$-structure which is a common exp*e*nsion to both $\mathfrak{B}$ and $\mathfrak{B}'$, and such that

$$\mathfrak{B} = \mathfrak{B}''|_S^{A^{\mathfrak{B}''}} \text{ and } \mathfrak{B}' = \mathfrak{B}''|_{S'}^{A'^{\mathfrak{B}''}}.$$

It follows that

$$\mathfrak{B}'' \models \Theta_f^{\delta} \cup T'^{\delta'} \cup \{\phi(S', S), \neg\phi(S, S'), \psi_S^A, \psi_{S'}^{A'}\} \supseteq \Phi_f$$

which contradicts the fact that Φ_f is inconsistent. Hence, any model of $\Theta_f \cup T$ is also a model of Θ. Thus, we have that

$$Mod_S^{\mathcal{L}}(\Theta_f \cup T) = \mathbb{C}.$$

It follows that $\mathbb{C}$ is $\mathcal{L}$-expressible. ∎

The results in this section show an interesting behaviour in the expressiveness of expressible preferential logics. Schlipf's Theorem (Theorem 2.3 above) shows that there are expressible preferential logics which have the Löwenheim-Skolem Property with respect to general models, but do not have this properties with respect to minimal models. Moreover, it shows that there are finite sets of sentences of some preferential logical systems $\mathcal{L}$ whose class of minimal models is not even Δ-$\mathcal{L}$-expressible (see Corollary 2.2). Theorem 2.4, however, shows that, whenever the class of minimal models of a finite set of sentences in a good logic $\mathcal{L}$ is Δ-$\mathcal{L}$-expressible, then it is in fact $\mathcal{L}$-expressible. That is, for good logics, there is no class of minimal models of finite sets of sentences which is strictly Δ-$\mathcal{L}$-expressible. Figure 2.1 below represents this situation.

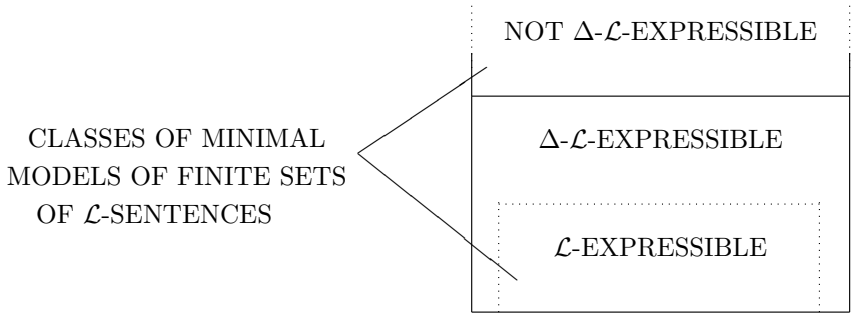


Figure 2.1: A gap in the expressiveness of (some) $\mathcal{L}$-expressible preferential logics, for good logics $\mathcal{L}$, with respect to classes of minimal models of finite sets of sentences.

It is important to note that the requirement of $\mathcal{L}$ in Theorem 2.4 above to be a good logic is too much strong. In fact, we only need that: i) the sen-

tence $\phi(S, S')$ which define the preference relation $\prec$ has a Boolean complement $\neg\phi(S, S')$, ii) it can express the sentences ψ_S^A and $\psi_{S'}^{A'}$ which guarantee that the subducts $\mathfrak{A}''|_S^{A^{\mathfrak{A}''}}$ and $\mathfrak{A}''|_{S'}^{A'^{\mathfrak{A}''}}$ are well defined, iii) it is capable of making relativizations for monadic predicate symbols of the language, and iv) it is compact. Hence, the theorem above also applies to abstract logics which are less expressive than first-order logic.

In the next section, we will show how to use Theorem 2.4 to obtain definability results about expressible preferential logics.

2.5 Definability

In Section 2.4, we introduced the concepts of defined relation and *definiens* of a relation. In this section, we will be concerned with definability properties of expressible preferential logics. We will need the following concepts of definability theory in order to prove a theorem about definable relations in classes of minimal models.

Definition 2.14 (P-defined Class of Structures) *Let $S \cup \{P\}$ be a symbol set where P is an n-ary relation symbol and let $\mathbb{C}$ be a class of $S\cup\{P\}$-structures. We say that $\mathbb{C}$ is P-defined iff for each $(\mathfrak{A}, \mathbf{P})$ and $(\mathfrak{A}, \mathbf{P}')$ in $\mathbb{C}$, $\mathbf{P} = \mathbf{P}'$.*

Definition 2.15 (Implicit Definition) *Let ϕ be an $S \cup \{P\}$-sentence of $\mathcal{L}$. We say that ϕ implicitly defines P iff the class $Mod^{\mathcal{L}}_{S\cup\{P\}}(\phi)$ is P-defined.*

The concept of explicit definition is language-dependent. Intuitively, an explicit definition is an expression like

$$Exp1 \equiv Exp2,$$

where $Exp1$, called *definiendum*, is an expression of the language that represents a symbol of the symbol set, $\equiv$ is an expression of the language which states, in some sense, the equivalence between the expressions at the left- and right-hand side, and $Exp2$, called *definiens*, is an expression which in some sense defines an object which can be used to interpret the *definiendum* and such that neither the *definiendum* nor any other expression which involves the symbol that represents the *definiendum* occurs in $Exp2$. For instance, in first-order logic, an expression like

$$\forall\overline{x}(P(\overline{x}) \leftrightarrow \psi(\overline{x}))$$

is an explicit definition for P, provided that P does not occur in $\psi(\overline{x})$. In Definability Theory, Padoa's method [Pad00] for first-order logic, consists in showing that an explicit definition for a symbol, say P, cannot be entailed from a theory T by exhibiting two models of T which differ on the interpretation of P but agree on the interpretation of the other symbols. That is the same of saying that, if a theory logically implies an explicit definition, then such theory implicitly defines the symbol represented by the *definiendum*. Beth's Definability Theorem for first-order logic [Bet53] is the converse. Beth showed that if a first-order set of sentences implicitly defines a symbol of the language, then such set entails some explicit definition for that symbol. As a consequence, implicitly defined symbols can be eliminated from first-order theories.

As we argued above, the concept of explicit definition is language dependent, even if we have a general shape for it. We can ask: how can we state Padoa's and Beth's Theorems without determine the syntax of our languages?

Although we cannot define what an explicit definition is for an arbitrary abstract logic, we can define what a *definiens* can be. First of all, a *definiens* cannot contain the defined symbol. If we restrict ourselves to relation symbols, we have the following.

Definition 2.16 (*Definiens* for Relation Symbols) *Let P be an n-ary relation symbol and $\mathcal{L} = (L, \mathbb{M}, \models)$ an abstract logic. A* definiens for P in $\mathcal{L}$ *is a sentence $\phi^{\overline{c}}$ in $L^{S\cup\{\overline{c}\}}$ for some S and $\overline{c} = \{c_1, \ldots, c_n\}$ such that $P \notin S$.*

Now, we can state analogues of Padoa's and Beth's properties for abstract logics.

Definition 2.17 (Padoa's Property) *An abstract logic $\mathcal{L} = (L, \mathbb{M}, \models)$ has the* Padoa's Property *iff, for each symbol set S and each S-theory T, if there is $\phi^{\overline{c}} \in L^{(S\cup\{\overline{c}\})-\{P\}}$ such that, for each S-structure $\mathfrak{A}$ which is a model of T, $\phi^{\overline{c},\mathfrak{A}} = P^{\mathfrak{A}}$, then $Mod^{\mathcal{L}}_S(T)$ is P-defined.*

Definition 2.18 (Beth's Property) *An abstract logic $\mathcal{L} = (L, \mathbb{M}, \models)$ has the* Beth's Property *iff, for each symbol set S and each S-theory T, if $Mod^{\mathcal{L}}_S(T)$ is P-defined, then there is an explicit definition $\phi^{\overline{c}} \in L^{(S\cup\{\overline{c}\})-\{P\}}$ such that, for each S-structure $\mathfrak{A}$ which is a model of T, $\phi^{\overline{c},\mathfrak{A}} = P^{\mathfrak{A}}$.*

The following definition makes precise the notion of replacing a predicate symbol with a defined relation.

Definition 2.19 (Replacement) *Let $\mathcal{L}$ be an abstract logic and $S \cup \{P, \overline{c}\}$ a symbol set. Let $\phi^{\overline{c}}$ be a sentence in $L^{S' \cup \{c_1,\ldots,c_n\}}$ where $S' \subseteq S \cup \{P\}$. We say that $\mathcal{L}$ has (relation) replacement iff, for each sentence $\psi(P) \in L^{S''}$ where $S'' \subseteq S \cup \{P\}$, there is a sentence $\psi(\phi^{\overline{c}}) \in L^{S' \cup S''}$ such that, for each $S \cup \{P\}$-structure $(\mathfrak{A}, \mathbf{P})$,*

$$(\mathfrak{A}, \mathbf{P}) \models \psi(\phi^{\overline{c}}) \text{ iff } (\mathfrak{A}, \phi^{\overline{c},(\mathfrak{A},\mathbf{P})}) \models \psi(P),$$

where $\phi^{\overline{c},(\mathfrak{A},\mathbf{P})}$ is the relation defined by $\phi^{\overline{c}}$ on $(\mathfrak{A}, \mathbf{P})$ (see Definition 2.12). We call $\psi(\phi^{\overline{c}})$ the sentence obtained from $\psi(P)$ by replacing P with $\phi^{\overline{c}}$.

Note that we can replace a predicate symbol, say P, for a defined relation ψ where the symbol P may occur.

Now, we deal with the following problem: Let $\mathcal{L}$ be a good logic which has Padoa's and Beth's Properties and replacement. Let T be a finite subset of L such that the class $\mathbb{C}$ of $\prec$-minimal models of T is P-defined for some $P \in S$. Suppose that $\mathbb{C}$ is Δ-$\mathcal{L}$-expressible and that there is a *definiens* $\phi^{c_1,\ldots,c_n}$ for P in L. When is it the case that

$$(\mathfrak{A}, \mathbf{P}) \models Th^{\mathcal{L}}(\mathbb{C})$$

iff,

$$(\mathfrak{A}, \mathbf{P}) \models T \text{ and, for all } \overline{\mathbf{a}} \in A^n, (\overline{\mathbf{a}}) \in \mathbf{P} \text{ iff } (\mathfrak{A}, \mathbf{P}, \overline{\mathbf{a}}) \models \phi^{c_1,\ldots,c_n}?$$

Or, if $\mathcal{L}$ has propositional biconditional,

$$Th^{\mathcal{L}}(\mathbb{C}) \equiv T \cup \{P(c_1, \ldots, c_n) \leftrightarrow \phi^{c_1,\ldots,c_n}\}?$$

If we obtain the equivalence above, what we have done is to represent a finite axiomatization of $Mod_S^{\mathcal{L}}(\mathbb{C})$, which is the class of $\prec$-minimal models of T, in terms of T and an explicit definition for P.

The following theorem follows as a consequence of Theorem 2.4.

Theorem 2.5 *Let $\mathcal{L} = (L, \mathbb{M}, \models)$ be a good logic which has both Padoa's and Beth's properties, replacement and propositional conjunction*[5]*. Let $\prec$ be an $\mathcal{L}$-expressible preference relation. Let T be a finite $\mathcal{L}$-theory whose class $\mathbb{C}$ of $\prec$ minimal models is Δ-$\mathcal{L}$-expressible. If $\mathbb{C}$ is P-defined and the models $\mathfrak{A}$ of T*

[5]It means that for each two sentences ϕ and ψ in $\mathcal{L}$, there is a third sentence, which we denote by $\phi \wedge \psi$, whose class of models is the intersection of the models of ϕ and ψ.

for which $P^{\mathfrak{A}} = \emptyset$*, if any, are* $\prec$*-minimal, then there is an explicit definition* $\phi'^{c_1,\ldots,c_n}$ *such that*

$$(\mathfrak{A}, \mathbf{P}) \models Th^{\mathcal{L}}(\mathbb{C})$$

iff,

$$(\mathfrak{A}, \mathbf{P}) \models T \text{ and, for all } \overline{\mathbf{a}} \in A^n, (\overline{\mathbf{a}}) \in \mathbf{P} \text{ iff } (\mathfrak{A}, \mathbf{P}, \overline{\mathbf{a}}) \models \phi^{c_1,\ldots,c_n}.$$

Proof. Let P be an n-ary predicate symbol. By Theorem 2.4 and the fact that $\mathcal{L}$ has propositional conjunction, there exists an $\mathcal{L}$-sentence $\gamma(P) = \Theta_f \wedge \bigwedge T$ such that

$$\mathbb{C} = Mod^{\mathcal{L}}_{S\cup\{P\}}(\gamma(P)).$$

As $\mathbb{C}$ is P-defined, by Beth's Property for $\mathcal{L}$, there is an explicit definition $\phi^{\overline{c}}$ such that, for each model $(\mathfrak{A}, \mathbf{P})$ of $\gamma(P)$,

$$\phi^{\overline{c},(\mathfrak{A},\mathbf{P})} = \phi^{\overline{c},\mathfrak{A}} = \mathbf{P}. \tag{2.11}$$

Let $\gamma' = \gamma(\phi^{\overline{c}})$ be the S-sentence obtained from $\gamma(P)$ by replacing P with $\phi^{\overline{c}}$ in $\gamma(P)$. Let $\mathfrak{A}$ be a model of γ' and $\mathbf{P} = \phi^{\overline{c},\mathfrak{A}}$. Then, since $\mathcal{L}$ has replacement, $(\mathfrak{A}, \mathbf{P}) \models \gamma(P)$. On the other hand, if $(\mathfrak{A}, \mathbf{P}) \models \gamma(P)$, then, by (2.11), we get $\mathbf{P} = \phi^{\overline{c},\mathfrak{A}}$ and, by replacement, we get $\mathfrak{A} \models \gamma'$. It follows that

$$(\mathfrak{A}, \mathbf{P}) \models \gamma(P) \quad \text{iff} \quad \mathfrak{A} \models \gamma' \text{ and } \mathbf{P} = \phi^{\overline{c},\mathfrak{A}}. \tag{2.12}$$

Let $\phi'^{\overline{c}} = \gamma' \wedge \phi^{\overline{c}}$. Let $(\mathfrak{A}, \mathbf{P})$ be a model of $\gamma(P)$. Since $\gamma(P) = \Theta_f \wedge \bigwedge T$, we have that

$$(\mathfrak{A}, \mathbf{P}) \models \bigwedge T. \tag{2.13}$$

By (2.11), we have that $\phi^{\overline{c},(\mathfrak{A},\mathbf{P})} = \mathbf{P}$. Let $\overline{\mathbf{a}} \in A^n$ a tuple of elements in A. We have that

$$\overline{\mathbf{a}} \in \mathbf{P} \text{ iff } (\mathfrak{A}, \overline{\mathbf{a}}) \models \phi^{\overline{c}}, \tag{2.14}$$

and, by (2.12),

$$(\mathfrak{A}, \overline{\mathbf{a}}) \models \phi^{\overline{c}} \text{ iff } (\mathfrak{A}, \overline{\mathbf{a}}) \models \gamma' \wedge \phi^{\overline{c}}. \tag{2.15}$$

From (2.14) and (2.15), we get

$$\overline{\mathbf{a}} \in \mathbf{P} \text{ iff } (\mathfrak{A}, \overline{\mathbf{a}}) \models \phi'^{\overline{c}}. \tag{2.16}$$

By (2.13) and (2.16), we have, for each $\overline{\mathbf{a}}$,

$$\overline{\mathbf{a}} \in \mathbf{P} \text{ iff } (\mathfrak{A}, \mathbf{P}, \overline{\mathbf{a}}) \models \bigwedge T \wedge \phi'^{\overline{c}}. \tag{2.17}$$

As (2.17) holds for any model $(\mathfrak{A}, \mathbf{P})$ of $\gamma(P)$, it follows that

$$\text{if } (\mathfrak{A}, \mathbf{P}) \models Th^{\mathcal{L}}(\mathbb{C}),$$

$$\text{then,} \tag{2.18}$$

$$(\mathfrak{A}, \mathbf{P}) \models T \text{ and, for all } \overline{\mathbf{a}} \in A^n, \overline{\mathbf{a}} \in P^{\mathfrak{A}} \text{ iff } (\mathfrak{A}, \mathbf{P}, \overline{\mathbf{a}}) \models \phi'^{\overline{c}}.$$

Now, suppose that

$$(\mathfrak{A}, \mathbf{P}) \models T \text{ and, for all } \overline{\mathbf{a}} \in A^n, \overline{\mathbf{a}} \in P^{\mathfrak{A}} \text{ iff } (\mathfrak{A}, \mathbf{P}, \overline{\mathbf{a}}) \models \phi'^{\overline{c}}. \tag{2.19}$$

If $\mathbf{P} = \emptyset$, then $(\mathfrak{A}, \mathbf{P})$ is a $\prec$-minimal model of T by hypothesis. If $\mathbf{P} \neq \emptyset$, then there is $\overline{a}$ in A^n such that $\overline{a} \in \mathbf{P}$. But then $(\mathfrak{A}, \mathbf{P}) \models T$, $(\mathfrak{A}, \mathbf{P}, \overline{a}) \models \phi'^{\overline{c}}$ and hence $(\mathfrak{A}, \mathbf{P}, \overline{a}) \models \gamma' \wedge \phi^{\overline{c}}$. Therefore $(\mathfrak{A}, \mathbf{P}) \models \gamma'$ and, as P does not occur in γ',

$$\mathfrak{A} \models \gamma'. \tag{2.20}$$

By (2.19), we have that $\mathbf{P} = \phi'^{\overline{c}, \mathfrak{A}}$. But as $\mathfrak{A} \models \gamma'$, it follows that

$$\mathbf{P} = \phi^{\overline{c}, \mathfrak{A}}. \tag{2.21}$$

From (2.20), (2.21) and (2.12), we have

$$(\mathfrak{A}, \mathbf{P}) \models \gamma(P),$$

and, hence, we have established that if

$$(\mathfrak{A}, \mathbf{P}) \models T \text{ and, for all } \overline{\mathbf{a}} \in A^n, \overline{\mathbf{a}} \in P^{\mathfrak{A}} \text{ iff } (\mathfrak{A}, \mathbf{P}, \overline{\mathbf{a}}) \models \phi'^{\overline{c}}.$$

$$\text{then,} \tag{2.22}$$

$$(\mathfrak{A}, \mathbf{P}) \models Th^{\mathcal{L}}(\mathbb{C}).$$

By (2.18) and (2.22) we have proved the theorem. ∎

Corollary 2.3 *Let $\mathcal{L} = (L, \mathbb{M}, \models)$, $\prec$, T and $\mathbb{C}$ be as in Theorem 2.5 above.*

Moreover, suppose $\mathcal{L}$ has propositional biconditional[6]*. Let $\mathcal{L}' = (L, \mathbb{M}, \models, \prec)$ be an abstract preferential logic. Then, for each $\alpha \in L^{S-\{P,c_1,\ldots,c_n\}}$,*

$$\Gamma \mathrel{|\!\sim_{\mathcal{L}'}} \alpha \quad \textit{iff} \quad T \cup \{P(c_1,\ldots,c_n) \leftrightarrow \phi^{c_1,\ldots,c_n}\} \vdash_{\mathcal{L}} \alpha,$$

where $\phi^{c_1,\ldots,c_n} \in L^{(S\cup\{c_1,\ldots,c_n\})-P}$ for some symbol set S.

2.6 Conclusions

In this chapter, we introduced abstract preferential logics, an extension of the concept of abstract logic to deal with nonmonotonic logic based on preference relations. We presented some expressiveness results about a class of abstract preferential logics, namely the elementary preferential logics. These logics are characterized by having a preference relation between interpretations which can be specified by a first-order formula. In particular, we showed that, whenever the class of minimal models of a finite theory from an elementary preferential logic based on a good logic is the class of models (minimal or not) of a theory in this logic, then it is the class of models (minimal or not) of a finite one. We also presented a definability results for elementary preferential logics. For instance, we showed the existence of an explicit definition when some theory of an elementary preferential logic implicitly defines a symbol in its class of minimal models.

The next two chapters are dedicated to the study of Finite Model Theory. In the next chapter, we will investigate the expressibility of problems in the polynomial hierarchy by sequences of formulas of hybrid logic.

[6]It means that for each two sentences ϕ and ψ in $\mathcal{L}$ there is a third sentence, which we denote by $\phi \leftrightarrow \psi$, whose models are the elements in $\mathbb{M}$ which satisfy both ϕ and ψ or do not satisfy ϕ nor ψ.

Chapter 3

PH Graph Properties in Hybrid Logic

In this chapter, we show that for each property of graphs $\mathcal{G}$ in NP there is a sequence $\phi_1, \phi_2, \ldots$ of formulas of the full hybrid logic which are satisfied exactly by the frames in $\mathcal{G}$. Moreover, the size of ϕ_n is bounded by a polynomial. We also show that the same holds for each graph property in the polynomial hierarchy. These results lead to the definition of syntactically defined fragments of hybrid logic whose model checking problem is complete for each degree in the polynomial hierarchy. The results presented here were published in [FFB$^+$11].

3.1 Introduction

The use of graphs as a mathematical abstraction of objects and structures makes it one of the most used concepts in computer science. Plenty of problems people want to solve using computers have their inputs modelled by graphs, and such problems commonly involve evaluating some graph property. To mention a well-known example, deciding whether a map can be coloured with a certain number of colors equals to a similar problem on planar graphs [HAK89, RSST97]. The applications of graphs in computer science do not restrict to model the input of problems. Graphs can be used in the theoretical framework in which some branches of computer science are formalized. This is the case, for example, in distributed systems, in which the model of computation is built up on a graph [Bar96, Lyn96]. Again, properties of graphs can be exploited in order to obtain

results about such models of computation.

In the last few decades, modal logics have attracted the attention of computer scientists working with logic and computation [BDRV02]. Among the reasons is the fact that modal logics often have interesting computer theoretical properties, like decidability [Var96, Grä01]. This is due to a lack of expressive power in comparison with other logics such as first-order logics and its extensions. Many modal logics present also good logical properties, like interpolation, definability, etc. Research in modal logic include augmenting the expressive power of the logic using resources as fixed-point operators [BS07] or hybrid languages [AtC07, ABM01]. Modal logics are particularly suitable to deal with graphs because standard semantics of most modal logics are built up from structures called *frames*, which are essentially graphs.

In [BS09], hybrid logics are used to express graph properties, like being connected, hamiltonian or eulerian. Several hybrid logics and fragments were studied to define graph properties through the concept of *validity in a frame* (see Definition 3.5 below). Some graph properties, like being hamiltonian, require a high expressive power and cannot be expressed by a single sentence in traditional hybrid logics. There are, however, sentences ϕ_n which can express such properties for frames of size n.

We are interested in expressing graph properties in NP, and more generally in PH, using hybrid logics (HL). The hybrid modal logics that we studied have low expressive power in comparison, for example, to second-order logic, hence we do not aim to associate to each graph property a single formula. Instead, we present, to each graph property a sequence of hybrid sentences ϕ_1, ϕ_2, ..., such that a graph of size n has the desired property iff ϕ_n is valid in the graph, regarded as a frame. In Section 3.2, we define the hybrid logic which we will study and define a prenex form for such logic. In Section 3.3, we show that, for any graph property, there is a sequence of sentences ϕ_1, ϕ_2, ... of the fragment of hybrid logic with nominals and the @ operator such that a graph of size n has the property iff ϕ_n is valid in the corresponding frame. However, the size of ϕ_n obtained is exponential on n. In Section 3.4, we show that, for graph properties in NP, and more generally in the polynomial hierarchy PH, there is such a sequence, but the size of the sentences is bounded by a polynomial on n. In Section 3.5, we show that, in general, the global modality cannot be disregarded. We also show how to obtain the results of the previous section for the fragment of HL without the global modality E and without nominals, provided that graphs are connected and with loops. In Section 3.6, we show

fragments of HL whose model-checking problem is complete for each degree of the polynomial hierarchy based on the results of the other sections. This gives an alternative proof for the NP-hardness of the model-checking problem for the fragment FHL$\setminus \downarrow \Box \downarrow$ of full hybrid logic given in [tCF05].

3.2 Hybrid Logic

In this section, we present the hybrid logic and the fragments which we will use. Hybrid modal logics extends classic modal logics by adding nominals and state variables to the language. Nominals and state variables behave like propositional atoms which are true in exactly one world. Other extensions include the operators $\downarrow$ (binder) and @. The $\downarrow$ allows one to assign the current state to a variable state. This can be used to keep a record of the visited states. The @ operator allows one to evaluate a formula in the state assigned to a certain nominal or state variable.

Definition 3.1 *The alphabet of the* hybrid logic (HL) with the $\downarrow$ binder *is a hybrid language consisting of a set Φ of countably many proposition symbols $p_1, p_2, \ldots$, a set $\mathcal{L}$ of countably many nominals $i_1, i_2, \ldots$, a set $\mathcal{S}$ of countably many state-variables $x_1, x_2, \ldots$, such that Φ, $\mathcal{L}$ and $\mathcal{S}$ are disjoint, the boolean connectives $\neg$ and $\wedge$ and the modal operators $@_i$, for each nominal i, $@_x$, for each state-variable x, $\Diamond$, $\Diamond^{-1}$, E and $\downarrow$. The language L_{FHL} of the (Full) Hybrid Logic can be defined by the following BNF rule:*

$$\alpha := \top \mid p \mid t \mid \neg\alpha \mid \alpha \wedge \alpha \mid \Diamond\alpha \mid \Diamond^{-1}\alpha \mid E\alpha \mid @_t\alpha \mid \downarrow x.\alpha.$$

For each $C \subseteq \{@, \downarrow, \Diamond^{-1}, E\}$, we define HL($C$) to be the corresponding fragment. In particular, we define FHL = HL($@, \downarrow, \Diamond^{-1}, E$). We also use HL($C$)$\setminus$NOM and HL($C$)$\setminus$PROP to refer to the fragments of HL(C) without nominals and propositional symbols respectively.

The standard boolean abbreviations $\rightarrow$, $\leftrightarrow$, $\vee$ and $\bot$ can be used with the standard meaning as well as the abbreviations of the dual modal operators: $A\phi = \neg E\neg\phi$, $\Box\phi = \neg\Diamond\neg\phi$ and $\Box^{-1}\phi = \neg\Diamond^{-1}\neg\phi$.

Formulas of hybrid modal logics are evaluated in *hybrid Kripke structures* (or *hybrid models*). These structures are built using *frames*.

Definition 3.2 *A* frame *is a graph $\mathcal{F} = (W, R)$, where W is a non-empty set (finite or not) of vertices and R is a binary relation over W, i.e., $R \subseteq W \times W$.*

Definition 3.3 *A* (hybrid) model *for the hybrid logic is a pair* $\mathcal{M} = (\mathcal{F}, \mathbf{V})$*, where* $\mathcal{F}$ *is a frame and* $\mathbf{V} : \Phi \cup \mathcal{L} \mapsto \mathcal{P}(W)$ *is a valuation function mapping proposition symbols into subsets of* W *and nominals into singleton subsets of* W*, i.e, if* i *is a nominal then* $\mathbf{V}(i) = \{v\}$ *for some* $v \in W$*.*

In order to deal with the state-variables, we need to introduce the notion of *assignments*.

Definition 3.4 *An* assignment *is a function* g *that maps state-variables to vertices of the model, i.e.,* $g : \mathcal{S} \mapsto W$*. We use the notation*

$$g' = g\frac{x_1 \dots x_n}{v_1 \dots v_n}$$

to denote an assignment g' *such that* $g'(x) = g(x)$ *if* $x \notin \{x_1, \dots, x_n\}$ *and* $g'(x_i) = v_i$*, otherwise.*

The semantical notion of satisfaction is defined as follows:

Definition 3.5 *Let* $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ *be a model. The notion of* satisfaction *of a formula* φ *in a model* $\mathcal{M}$ *at a vertex* v *under an assignment* g*, notation* $\mathcal{M}, g, v \Vdash \varphi$*, can be inductively defined as follows:*

$\mathcal{M}, g, v \Vdash p$ *iff* $v \in \mathbf{V}(p)$*;*
$\mathcal{M}, g, v \Vdash \top$ *always;*
$\mathcal{M}, g, v \Vdash \neg\varphi$ *iff* $\mathcal{M}, g, v \nVdash \varphi$*;*
$\mathcal{M}, g, v \Vdash \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, g, v \Vdash \varphi_1$ *and* $\mathcal{M}, g, v \Vdash \varphi_2$*;*
$\mathcal{M}, g, v \Vdash \Diamond\varphi$ *iff there is a* $w \in W$ *such that* vRw *and* $\mathcal{M}, g, w \Vdash \varphi$*;*
$\mathcal{M}, g, v \Vdash \Diamond^{-1}\varphi$ *iff there is a* $w \in W$ *such that* wRv *and* $\mathcal{M}, g, w \Vdash \varphi$*;*
$\mathcal{M}, g, v \Vdash E\varphi$ *iff there is a* $w \in W$ *such that* $\mathcal{M}, g, w \Vdash \varphi$*;*
$\mathcal{M}, g, v \Vdash i$ *iff* $v \in \mathbf{V}(i)$*;*
$\mathcal{M}, g, v \Vdash @_i\varphi$ *iff* $\mathcal{M}, g, d_i \Vdash \varphi$*, where* $\{d_i\} = \mathbf{V}(i)$*;*
$\mathcal{M}, g, v \Vdash x$ *iff* $g(x) = v$*;*
$\mathcal{M}, g, v \Vdash @_x\varphi$ *iff* $\mathcal{M}, g, d \Vdash \varphi$*, where* $d = g(x)$*;*
$\mathcal{M}, g, v \Vdash\downarrow x.\varphi$ *iff* $\mathcal{M}, g\frac{x}{v}, v \Vdash \varphi$*.*

For each nominal i, the formula $@_i\varphi$ means that if $\mathbf{V}(i) = \{v\}$ then φ is satisfied at v. If $\mathcal{M}, g, v \Vdash \varphi$ for every vertex v, we say that φ is *globally true* in the model $\mathcal{M}$ under the assignment g ($\mathcal{M}, g \Vdash \varphi$) and if φ is globally true in all models $\mathcal{M}$ of a frame $\mathcal{F}$ for all possible assignments, we say that φ is *valid* in $\mathcal{F}$ ($\mathcal{F} \Vdash \varphi$).

In the following, we define a prenex form for formulas in FHL and show that any formula in FHL has an equivalent in prenex form. We use this form to define classes of formulas whose model-checking problem is complete for the degrees of the polynomial hierarchy (see Section 3.6).

Definition 3.6 (Prenex Form) *A formula ϕ in FHL is in* prenex form *iff $\phi = q_1 \dots q_n \psi$ where each q_i is A, E, $\Box$, $\Diamond$, $\Box^{-1}$, $\Diamond^{-1}$ or $\downarrow x$. for some x and ψ has no occurrence of $\downarrow$ and modalities in ψ occur only in front of atoms.*

First-order logic language of graphs (first-order language on the symbol set $\{E\}$, E a binary relation[1]) can be translated into the full hybrid logic and the full hybrid logic can be translated into the first-order logic language of graphs as well, and both translations preserve truth [AtC07]. That is, full hybrid logic has the same expressive power as first-order logic. Using the *hybrid translation* from first-order logic to full hybrid logic and the *standard translation* from full hybrid logic back to first-order logic, we can prove a prenex normal form theorem for hybrid logic. We present below the hybrid translation HT and the standard translation ST. Let E be a binary relation symbol and for each propositional symbol p in the modal hybrid language let P be a monadic relation symbol. Let $\Psi = \{P | p \in \Phi\} \cup \{c_i | i \in \mathcal{L}\}$. Let $L^{\{E\} \cup \Psi}$ the first-order language on the symbol set $\{E\} \cup \Psi$ and x a first-order variable. The functions $ST_x : FHL \to L^{\{E\} \cup \Psi}$ and $HT : L^{\{E\} \cup \Psi} \to FHL$ are defined as follows (see [AtC07]):

$$\begin{aligned}
HT(E(x,y)) &= @_x \Diamond y \\
HT(P(x)) &= @_x p \\
HT(x = y) &= @_x y \\
HT(\neg\phi) &= \neg HT(\phi) \\
HT(\phi \wedge \psi) &= HT(\phi) \wedge HT(\psi) \\
HT(\exists x \phi) &= E \downarrow x . HT(\phi)
\end{aligned}$$

[1] We use the symbol E to represent both the global existential modality and the binary relation symbol of the graph language. It is clear from the context which one is the case.

$$
\begin{aligned}
ST_x(s) &= x = s \\
ST_x(p) &= P(x) \\
ST_x(\neg\phi) &= \neg ST_x(\phi) \\
ST_x(\phi \wedge \psi) &= ST_x(\phi) \wedge ST_x(\psi) \\
ST_x(\Diamond\phi) &= \exists y(R(x,y) \wedge ST_y(\phi)) \\
ST_x(\Diamond^{-1}\phi) &= \exists y(R(y,x) \wedge ST_y(\phi)) \\
ST_x(E\phi) &= \exists y(ST_y(\phi)) \\
ST_x(@_s\phi) &= ST_s(\phi) \\
ST_x(\downarrow z.\phi) &= \exists z(z = x \wedge ST_x(\phi))
\end{aligned}
$$

Given a hybrid model $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ we can associate a first-order structure $\mathcal{M}' = (\mathcal{F}, \{\mathbf{P}|p \in \Phi\}, \{\mathbf{i}|i \in \mathcal{L}\})$ where $\mathbf{P} = \mathbf{V}(p)$ and $\mathbf{i} = g(i)$. In the following, we will use the same symbol to refer both to the hybrid model and the corresponding first-order structure.

Lemma 3.1 ([tCF05, ABM01]) *Let α be a formula in FHL where the variable x does not occur and ϕ a formula of first-order logic. Let $\mathcal{M}$ be a model, g an assignment of variables and w a state in $\mathcal{M}$.*

- $\mathcal{M}, g, w \Vdash \alpha$ *iff* $(\mathcal{M}, g\frac{x}{w}) \models ST_x(\alpha)$.
- $(\mathcal{M}, g) \models \phi$ *iff* $\mathcal{M}, g \Vdash HT(\phi)$.

It follows from Lemma 3.1 that the modality $\Diamond^{-1}$ can be eliminated by translating a hybrid formula into FO using the standard translation and back to FHL using the hybrid translation. Hence, we do not need to consider it in the proofs below.

In the following, we will show a series of lemmas about hybrid logic that will be used in the following sections.

Using the standard and hybrid translation, the definition of the satisfaction relation and Lemma 3.1, we have the following equivalence in hybrid modal logic.

Lemma 3.2 *The following equivalences hold in FHL:*

1. $(A\alpha \wedge \beta) \equiv \downarrow x.A(\alpha \wedge @_x\beta)$;

2. $(E\alpha \wedge \beta) \equiv \downarrow x.E(\alpha \wedge @_x\beta)$;

3. $(A\alpha \vee \beta) \equiv \downarrow x.A(\alpha \vee @_x\beta)$;

4. $(E\alpha \vee \beta) \equiv \downarrow x.E(\alpha \vee @_x\beta)$;

5. $\neg \downarrow x.\phi \equiv \downarrow x.\neg\phi$;

6. $@_xE\phi \equiv E\phi$;

7. $@_xA\phi \equiv A\phi$;

8. $((\downarrow x.\alpha) \wedge \beta) \equiv \downarrow x.(\alpha \wedge \beta)$, *x not occurring in* β;

9. $((\downarrow x.\alpha) \vee \beta) \equiv \downarrow x.(\alpha \vee \beta)$, *x not occurring in* β;

10. $(\Diamond\alpha \wedge \beta) \equiv \downarrow x.E \downarrow y.[(@_x\Diamond y \wedge HT(ST_y(\alpha))) \wedge HT(ST_x(\beta))]$, *x and y not occurring in* α *or* β;

11. $(\Box\alpha \wedge \beta) \equiv \downarrow x.A \downarrow y.[(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \wedge HT(ST_x(\beta))]$, *x and y not occurring in* α *or* β;

12. $(\Diamond\alpha \vee \beta) \equiv \downarrow x.E \downarrow y.[(@_x\Diamond y \wedge HT(ST_y(\alpha))) \vee HT(ST_x(\beta))]$, *x and y not occurring in* α *or* β;

13. $(\Box\alpha \vee \beta) \equiv \downarrow x.A \downarrow y.[(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \vee HT(ST_x(\beta))]$, *x and y not occurring in* α *or* β.

Proof. The first nine equivalences can be easily proved by directly applying the definition of the satisfiability relation. Let $\mathcal{M}$ be a model, g an assignment and v a vertex in $\mathcal{M}$. We have:

1. $\mathcal{M}, g, v \Vdash (A\alpha \wedge \beta)$ iff $\mathcal{M}, g, v \Vdash A\alpha$ and $\mathcal{M}, g, v \Vdash \beta$ iff, for all $v' \in V$, $\mathcal{M}, g, v' \Vdash \alpha$ and $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff, for all $v' \in V$, $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ and $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff, for all $v' \in V$, $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha \wedge @_x\beta$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash A(\alpha \wedge @_x\beta)$ iff $\mathcal{M}, g, v \Vdash \downarrow x.A(\alpha \wedge @_x\beta)$.

2. $\mathcal{M}, g, v \Vdash (E\alpha \wedge \beta)$ iff $\mathcal{M}, g, v \Vdash E\alpha$ and $\mathcal{M}, g, v \Vdash \beta$ iff exists $v' \in V$ such that $\mathcal{M}, g, v' \Vdash \alpha$ and $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff exists $v' \in V$ such that $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ and $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff exists $v' \in V$ such that $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha \wedge @_x\beta$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash E(\alpha \wedge @_x\beta)$ iff $\mathcal{M}, g, v \Vdash \downarrow x.E(\alpha \wedge @_x\beta)$.

3. $\mathcal{M}, g, v \Vdash (A\alpha \vee \beta)$ iff $\mathcal{M}, g, v \Vdash A\alpha$ or $\mathcal{M}, g, v \Vdash \beta$ iff, for all $v' \in V$, $\mathcal{M}, g, v' \Vdash \alpha$ or $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff, for all $v' \in V$, $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ or $\mathcal{M}, g\frac{v}{x}, v' \Vdash$

$@_x\beta$ iff, for all $v' \in V$, $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha \vee @_x\beta$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash A(\alpha \vee @_x\beta)$ iff $\mathcal{M}, g, v \Vdash\downarrow x.A(\alpha \vee @_x\beta)$.

4. $\mathcal{M}, g, v \Vdash (E\alpha \vee \beta)$ iff $\mathcal{M}, g, v \Vdash E\alpha$ or $\mathcal{M}, g, v \Vdash \beta$ iff exists $v' \in V$ such that $\mathcal{M}, g, v' \Vdash \alpha$ or $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff exists $v' \in V$ such that $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ or $\mathcal{M}, g\frac{v}{x}, v' \Vdash @_x\beta$ iff exists $v' \in V$ such that $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha \vee @_x\beta$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash E(\alpha \vee @_x\beta)$ iff $\mathcal{M}, g, v \Vdash\downarrow x.E(\alpha \vee @_x\beta)$.

5. $\mathcal{M}, g, v \Vdash \neg \downarrow x.\phi$ iff not $\mathcal{M}, g, v \Vdash\downarrow x.\phi$ iff not $\mathcal{M}, g\frac{v}{x}, v \Vdash \phi$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash \neg\phi$ iff $\mathcal{M}, g, v \Vdash\downarrow x.\neg\phi$.

6. $\mathcal{M}, g, v \Vdash @_xE\phi$ iff $\mathcal{M}, g, g(x) \Vdash E\phi$ iff exists $v' \in V$ such that $\mathcal{M}, g, v' \Vdash \phi$ iff $\mathcal{M}, g, v \Vdash E\phi$.

7. $\mathcal{M}, g, v \Vdash @_xA\phi$ iff $\mathcal{M}, g, g(x) \Vdash A\phi$ iff, for all $v' \in V$, $\mathcal{M}, g, v' \Vdash \phi$ iff $\mathcal{M}, g, v \Vdash A\phi$.

8. $\mathcal{M}, g, v \Vdash ((\downarrow x.\alpha) \wedge \beta)$ iff $\mathcal{M}, g, v \Vdash (\downarrow x.\alpha)$ and $\mathcal{M}, g, v \Vdash \beta$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash \alpha$ and $\mathcal{M}, g\frac{v}{x}, v \Vdash \beta$, as x does not occur in β, iff $\mathcal{M}, g, v \Vdash\downarrow x.(\alpha \wedge \beta)$.

9. $\mathcal{M}, g, v \Vdash ((\downarrow x.\alpha) \vee \beta)$ iff $\mathcal{M}, g, v \Vdash (\downarrow x.\alpha)$ or $\mathcal{M}, g, v \Vdash \beta$ iff $\mathcal{M}, g\frac{v}{x}, v \Vdash \alpha$ or $\mathcal{M}, g\frac{v}{x}, v \Vdash \beta$, as x does not occur in β, iff $\mathcal{M}, g, v \Vdash\downarrow x.(\alpha \vee \beta)$.

For the next four equivalences, we need Lemma 3.2.

10. $\mathcal{M}, g, v \Vdash (\Diamond\alpha \wedge \beta)$ iff $\mathcal{M}, g, v \Vdash \Diamond\alpha$ and $\mathcal{M}, g, v \Vdash \beta$ iff exists $v' \in V$ with vRv' such that $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ and $\mathcal{M}, g, v \Vdash \beta$, iff, by Lemma 3.2, exists $v' \in V$ with vRv' such that $\mathcal{M}, g\frac{v'}{y}, v' \Vdash HT(ST_y(\alpha))$ and $\mathcal{M}, g\frac{v}{x}, v \Vdash HT(ST_x(\beta))$, iff exists $v' \in V$ such that $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_x\Diamond y \wedge HT(ST_y(\alpha))$ and $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_xHT(ST_x(\beta))$, since y does not occur in α or β, iff exists $v' \in V$ such that $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash (@_x\Diamond y \wedge HT(ST_y(\alpha))) \wedge HT(ST_x(\beta))$, iff $\mathcal{M}, g\frac{v}{x}, v \Vdash E \downarrow y.(@_x\Diamond y \wedge HT(ST_y(\alpha))) \wedge HT(ST_x(\beta))$, iff $\mathcal{M}, g, v \Vdash\downarrow x.E \downarrow y.(@_x\Diamond y \wedge HT(ST_y(\alpha))) \wedge HT(ST_x(\beta))$.

11. $\mathcal{M}, g, v \Vdash (\Box\alpha \wedge \beta)$ iff $\mathcal{M}, g, v \Vdash \Diamond\alpha$ and $\mathcal{M}, g, v \Vdash \beta$ iff, for all $v' \in V$ with vRv', $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ and $\mathcal{M}, g, v \Vdash \beta$, iff, by Lemma 3.2, for all $v' \in V$ with vRv', $\mathcal{M}, g\frac{v'}{y}, v' \Vdash HT(ST_y(\alpha))$ and $\mathcal{M}, g\frac{v}{x}, v \Vdash HT(ST_x(\beta))$, iff, for all $v' \in V$, $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_x\Diamond y \rightarrow HT(ST_y(\alpha))$ and $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_xHT(ST_x(\beta))$, since y does not occur in α or β, iff, for all $v' \in V$, $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash (@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \wedge @_xHT(ST_x(\beta))$, iff $\mathcal{M}, g\frac{v}{x}, v \Vdash E \downarrow y.(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \wedge @_xHT(ST_x(\beta))$, iff $\mathcal{M}, g, v \Vdash\downarrow x.E \downarrow y.(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \wedge @_xHT(ST_x(\beta))$.

12. $\mathcal{M}, g, v \Vdash (\Diamond\alpha \vee \beta)$ iff $\mathcal{M}, g, v \Vdash \Diamond\alpha$ or $\mathcal{M}, g, v \Vdash \beta$ iff exists $v' \in V$ with vRv' such that $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ or $\mathcal{M}, g, v \Vdash \beta$, iff, by Lemma 3.2, exists $v' \in V$

with vRv' such that $\mathcal{M}, g\frac{v'}{y}, v' \Vdash HT(ST_y(\alpha))$ or $\mathcal{M}, g\frac{v}{x}, v \Vdash HT(ST_x(\beta))$, iff exists $v' \in V$ such that $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_x\Diamond y \wedge HT(ST_y(\alpha))$ or $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_x HT(ST_x(\beta))$, since y does not occur in α or β, iff exists $v' \in V$ such that $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash (@_x\Diamond y \wedge HT(ST_y(\alpha))) \vee @_x HT(ST_x(\beta))$, iff $\mathcal{M}, g\frac{v}{x}, v \Vdash E \downarrow y.(@_x\Diamond y \wedge HT(ST_y(\alpha))) \vee @_x HT(ST_x(\beta))$, iff $\mathcal{M}, g, v \Vdash\downarrow x.E \downarrow y.(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \vee HT(ST_x(\beta))$.

13. $\mathcal{M}, g, v \Vdash (\Box\alpha \vee \beta)$ iff $\mathcal{M}, g, v \Vdash \Diamond\alpha$ or $\mathcal{M}, g, v \Vdash \beta$ iff, for all $v' \in V$ with vRv', $\mathcal{M}, g\frac{v}{x}, v' \Vdash \alpha$ or $\mathcal{M}, g, v \Vdash \beta$, iff, by Lemma 3.2, for all $v' \in V$ with vRv', $\mathcal{M}, g\frac{v'}{y}, v' \Vdash HT(ST_y(\alpha))$ or $\mathcal{M}, g\frac{v}{x}, v \Vdash HT(ST_x(\beta))$, iff, for all $v' \in V$, $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_x\Diamond y \rightarrow HT(ST_y(\alpha))$ or $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash @_x HT(ST_x(\beta))$, since y does not occur in α or β, iff, for all $v' \in V$, $\mathcal{M}, g\frac{vv'}{xy}, v' \Vdash (@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \vee @_x HT(ST_x(\beta))$, iff $\mathcal{M}, g\frac{v}{x}, v \Vdash E \downarrow y.(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \vee @_x HT(ST_x(\beta))$, iff $\mathcal{M}, g, v \Vdash\downarrow x.E \downarrow y.(@_x\Diamond y \rightarrow HT(ST_y(\alpha))) \vee @_x HT(ST_x(\beta))$. ∎

By performing iterated applications of these equivalences, we can see that each formula in FHL can be put in the prenex form.

Lemma 3.3 *If $\phi \in$ FHL, then there is $\psi \in$ FHL in prenex form which is equivalent to ϕ.*

Proof. By induction on the number of modality occurrences in ϕ and applying the equivalences in Lemma 3.2. ∎

This prenex form can be strengthened with the following lemma:

Lemma 3.4 $\downarrow x. \downarrow y.\phi(x, y) \equiv\downarrow x.\phi(x, x)$*, if x does not occur bound in ϕ, where $\phi(x, x)$ is obtained from $\phi(x, y)$ by substituting free occurrences of y with x.*

Proof. It follows straightforward by induction on ϕ. ∎

Lemma 3.5 *If $\phi \in FHL$, then ϕ is equivalent to a formula in FHL in prenex form whose prefix has no consecutive applications of binders.*

We end this section with the definition of the hierarchies of hybrid formulas induced by the prefix in the prenex form. Based on Lemma 3.5, we define the following classes of formulas:

Definition 3.7 *We recursively define the classes of formulas σ^i and π^i in prenex form as:*

- $\sigma^0 = \pi^0 = \{\phi \in HL | \textit{modalities occur in } \phi \textit{ only in front of atoms}\}$;
- $\sigma^{i+1} = \{\phi \in HL | \phi = q_1 \dots q_n \psi, \psi \in \pi^i, q_j = \Diamond, E \textit{ or } \downarrow x., \textit{ for some } x\}$;
- $\pi^{i+1} = \{\phi \in HL | \phi = q_1 \dots q_n \psi, \psi \in \sigma^i, q_j = \Box, A \textit{ or } \downarrow x., \textit{ for some } x\}$.

We say that a formula is σ^i (resp. π^i) if it is equivalent to a formula in σ^i (resp. π^i).

From Lemma 3.3 it follows that each formula in HL is π^i or σ^i for some i.

In the following section, we talk about the expressibility of graph properties in hybrid logic with respect to frame definability.

3.3 Properties of Graphs in HL

In [BS09], it was shown that there is a formula ϕ_n of FHL such that a graph G of size n is Hamiltonian iff ϕ_n is valid in G. The main question which underlies this investigation is whether there is a sequence of formulas $(\phi_n)_{n \in N}$ for each graph property $\mathcal{G}$ in NP such that a graph G of size n is in $\mathcal{G}$ iff ϕ_n is globally true in G, regarded as a frame. Actually, we can show that such sequence exists for each graph property. This follows directly from the equivalence between FHL ans FO. W.r.t. frame definability, we can show the existence of such formulas in a very restrict fragment of FHL. Recall that a graph property is any set of graphs closed under isomorphisms.

Let $G = (V, E)$ be a graph of cardinality n. Let us consider that the set V of vertices coincides with the set $\{1, \dots, n\}$ of nominals. Consider the formula:

$$\psi_G = \bigwedge_{(i,j) \in E} @_i \Diamond j \wedge \bigwedge_{(i,j) \notin E} @_i \neg \Diamond j.$$

Let $\mathcal{G}$ be any property of graphs. We define the formulas

$$\psi^n_{\mathcal{G}} = \bigvee_{G \in \mathcal{G}, |G| = n} \psi_G \quad , \quad \theta^n = \bigwedge_{i,j \in \{1,\dots,n\}, i \neq j} @_i \neg j \quad \text{and} \quad \phi^n_{\mathcal{G}} = \theta^n \rightarrow \psi^n_{\mathcal{G}}.$$

Lemma 3.6 *Let G be a graph of cardinality n and $\mathcal{G}$ a property of graphs. Then $G \in \mathcal{G}$ iff $G \Vdash \phi^n_{\mathcal{G}}$.*

Proof. Let $G = (V, E)$ be a graph. Then $G \Vdash \phi^n_{\mathcal{G}}$ iff, for each valuation function $\mathbf{V}$ of the nominals we have, $(G, \mathbf{V}) \Vdash \phi^n_{\mathcal{G}}$. Let $\mathbf{V}$ be a valuation function. Suppose $(G, \mathbf{V}) \Vdash \phi^n_{\mathcal{G}}$. If $(G, \mathbf{V}) \Vdash \theta^n$, then $\mathbf{V}$ assigns to each nominal a different element (that is, the restriction of $\mathbf{V}$ to the set $\{1, \dots, n\}$ of nominals

is injective). In this case, $(G, \mathbf{V}) \Vdash \psi_{G'}$ for some $G' \in \mathcal{G}$. It follows that $(G, \mathbf{V}) \Vdash @_i \Diamond j$ iff $(\mathbf{V}(i), \mathbf{V}(j)) \in E$ iff $(i, j) \in E'$. Hence, the restriction of $\mathbf{V}$ to $\{1, \ldots, n\}$ is an isomorphism between G and G'. Then $G \in \mathcal{G}$.

Now, suppose $G \in \mathcal{G}$. Let $\mathbf{V}$ be a valuation function which is injective in the set $\{1, \ldots, n\}$ of nominals. It follows that $(G, \mathbf{V}) \Vdash \theta^n$ and $(G, \mathbf{V}) \Vdash @_i \Diamond j$ iff $(\mathbf{V}^{-1}(i), \mathbf{V}^{-1}(j)) \in E$. $\mathbf{V}$ induces a graph $G' = (V', E')$ isomorphic to G where $V' = V$ and $(i, j) \in E'$ iff $(\mathbf{V}^{-1}(i), \mathbf{V}^{-1}(j)) \in E'$. Hence, $(i, j) \in E'$ iff $(G, \mathbf{V}) \Vdash @_i \Diamond j$. It follows that $(G, \mathbf{V}) \Vdash \psi_{G'}$. Then $(G, \mathbf{V}) \Vdash \phi^n_{\mathcal{G}}$, for each valuation function $\mathbf{V}$. It follows that $G \Vdash \phi^n_{\mathcal{G}}$. ∎

Since there are 2^{n^2} graphs with vertices in $\{1, \ldots, n\}$, we have that the size of $\phi^n_{\mathcal{G}}$ is $O(2^{n^2})$ for any graph property $\mathcal{G}$. Obviously, there is no hope that such sequence of formulas will always be recursive. We can show, however, that for problems in the polynomial hierarchy such sequence is recursive and, moreover, there is a polynomial bound in the size of formulas.

3.4 Translation

In this section, we show that for each graph property $\mathcal{G}$ in the polynomial hierarchy there is a sequence $(\phi_n)_{n \in N}$ of formulas such that a graph G is in $\mathcal{G}$ iff $G \Vdash \phi_{|G|}$ and such that ϕ_n is bounded from above by a polynomial on n. We will use the well-known characterization of problems in PH and classes of finite models definable in second-order logic (SO) from descriptive complexity theory. To this end, we define a translation from formulas in SO to formulas in FHL which are equivalent with respect to frames of size n, for some $n \in N$. Such translation will give us formulas whose size is bounded by a polynomial on n. Moreover, the formulas obtained by the translation have no occurrence of propositional symbols, nominals or free state variables, which means that, for these formulas, the complexity of model-checking and frame-checking coincides. We use the well known definitions and concepts related to first- and second-order logic which can be found in most textbooks (see, for instance, [EFT94b]).

Definition 3.8 (Translation from FO to FHL) *Let ϕ be a first-order formula in the symbol set $S = \{E, R_1, \ldots, R_m\}$, E binary, n a natural number and f a function from the set of first-order variables into $\{1, \ldots, n\}$. Let $t, z_1, \ldots, z_n$ be state variables and for each $R \in \{R_1, \ldots, R_m\}$ of arity h, let $y^R_{j_1, \ldots, j_h}$ be a state variable, with $j_i \in \{1, \ldots, n\}$, $1 \leq i \leq h$. We define the function $tr^f_n : L^S_{FO} \to L_{FHL}$ as:*

- $tr_n^f(x_1 \equiv x_2) = @_{z_{f(x_1)}} z_{f(x_2)}$;
- $tr_n^f(E(x_1, x_2)) = @_{z_{f(x_1)}} \Diamond z_{f(x_2)}$;
- $tr_n^f(R(x_1, \ldots, x_k)) = @_t y^R_{f(x_1),\ldots,f(x_k)}$, *for each* $R \in \{R_1, \ldots, R_m\}$;
- $tr_n^f(\gamma \wedge \theta) = tr_n^f(\gamma) \wedge tr_n^f(\theta)$;
- $tr_n^f(\neg\gamma) = \neg tr_n^f(\gamma)$;
- $tr_n^f(\exists x\gamma) = \bigvee_{i=1}^n tr_n^{f\frac{x}{i}}(\gamma)$;
- $tr_n^f(\forall x\gamma) = \bigwedge_{i=1}^n tr_n^{f\frac{x}{i}}(\gamma)$.

In the translation above, t is intended to represent a state v such that, if $z^R_{j_1,\ldots,j_h}$ is assigned to t and $z_{j_1}, \ldots, z_{j_h}$ are assigned to $v_1, \ldots, v_h$, then $(v_1, \ldots, v_h)$ belongs to the interpretation of R. The function $f\frac{x}{i}$ maps x to i and y to $f(y)$ for $y \neq x$. The translation above only works for frames with more than one state but, since there are only two frames of size 1, we can state for each graph property which frames of size 1 belong to the property.

Note that if ϕ is a sentence, then $tr_n^f(\phi) = tr_n^{f'}(\phi)$. Hence we write $tr_n(\phi)$ instead of $tr_n^f(\phi)$ for a sentence ϕ.

Example 3.1 *We give an example of application of the translation above. Let* $\oplus(\phi, \psi, \theta)$ *be the ternary exclusive "or". Consider the following first-order sentence:*

$$\begin{aligned}\phi \;=\; & \forall x\big(\oplus(R(x), G(x), B(x))\big) \wedge \forall x \forall y\big((E(x,y) \wedge x \neq y) \rightarrow \\ & \neg((R(x) \wedge R(y)) \vee (G(x) \wedge G(y)) \vee (B(x) \wedge B(y)))\big).\end{aligned}$$

The sentence above says that each element belongs to one of the sets R, G *and* B, *each adjacent pair does not belong to the same set, and no element belongs to more than one set. This sentence is true iff the sets* R, G *and* B *forms a* 3*-colouring of a graph with edges in* E. *Below we translate* ϕ *into a formula of hybrid logic using the translation given above and setting* $n = 3$*:*

$$\begin{aligned}tr_n(\phi) \;=\; & \bigwedge_{i=1}^{3} \oplus\big(@_t y_i^R, @_t y_i^G, @_t y_i^B\big) \wedge \bigwedge_{i=1}^{3} \Bigg[\bigwedge_{j=1}^{3} \big((@_{z_i} \Diamond z_j \wedge \neg @_{z_i} z_j) \rightarrow \\ & \neg((@_t y_i^R \wedge @_t y_j^R) \vee (@_t y_i^G \wedge @_t y_j^G) \vee (@_t y_i^B \wedge @_t y_j^B))\big)\Bigg].\end{aligned}$$

Lemma 3.7 $tr_n(\phi)$ *has polynomial size in* n *for each fixed formula* ϕ, *that is,* $tr_n(\phi) \in O(n^k)$ *for some* $0 \leq k$.

Proof. By induction on ϕ one can see that $tr_n(\phi)$ is $O(n^k)$, where k is the quantifier rank of ϕ. ∎

Lemma 3.8 *Let $G = (V, E^G)$ be a graph of cardinality n, $\mathbf{R}_1, \ldots, \mathbf{R}_m$ relations on V with arities $r_1, \ldots, r_m$, g an assignment of state variables, β an assignment of first-order variables and f a function from the set of first-order variables to $\{1, \ldots, n\}$ such that:*

(i) g assigns to each variable z_i, $1 \leq i \leq n$, a different element in V;

(ii) $g(y^R_{i_1,\ldots,i_k}) = g(t)$ iff $(g(z_{i_1}), \ldots, g(z_{i_k})) \in \mathbf{R}$ for each $R \in \{R_1, \ldots, R_m\}$;

(iii) $\beta(x) = g(z_{f(x)})$ for each first-order variable x.

If ϕ is a first-order formula in the symbol set $\{E, R_1, \ldots, R_m\}$, then

$$(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi \text{ iff for all } w \in V,\ (G, g, w) \Vdash tr^f_n(\phi).$$

Proof. We proceed by induction on ϕ.

— ϕ is atomic: In this case $\phi = x \equiv y$, $\phi = E(x, y)$ or $\phi = R_i(x_1, \ldots, x_n)$. If $\phi = x \equiv y$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff $\beta(x) = \beta(y)$ iff, by (iii), $g(z_{f(x)}) = g(z_{f(y)})$ iff $(G, g, w) \Vdash @_{z_{f(x)}} z_{f(y)} = tr^f_n(\phi)$. If $\phi = E(x, y)$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff $(\beta(x), \beta(y)) \in E^G$ iff, by (iii), $(g(z_{f(x)}), g(z_{f(y)}) \in E^G$ iff $(G, g, w) \Vdash @_{z_{f(x_1)}} \Diamond z_{f(x_2)} = tr^f_n(\phi)$. If $\phi = R_i(x_1, \ldots, x_h)$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff $(\beta(x_1), \ldots, \beta(x_n)) \in \mathbf{R}_i$ iff, by (iii), $(g(z_{f(x_1)}), \ldots, g(z_{f(x_n)})) \in \mathbf{R}_i$ iff, by (ii), $g(y^R_{f(x_1),\ldots,f(x_k)}) = g(t)$ iff $(G, g, w) \Vdash @_t y^R_{f(x_1),\ldots,f(x_k)} = tr^f_n(\phi)$.

— $\phi = \gamma \wedge \theta$ or $\phi = \neg\gamma$: These cases follow directly from the definition of tr^f_n and the inductive hypothesis.

— $\phi = \exists x\gamma$: In this case, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff there is a $v \in V$ such that $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta^x_v) \models \gamma$. By (i), there is a j be such that $v = z_j$. Hence we have $\beta^x_v(y) = g(z_{f^x_j(y)})$ for each first-order variable y. By inductive hypothesis we have, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta^x_v) \models \gamma$ iff $(G, g, w) \Vdash tr^{f^x_j}_n(\gamma)$ iff $(G, g, w) \Vdash \bigvee_{i=1}^n tr^{f^x_i}_n(\gamma) = tr^f_n(\phi)$.

— $\phi = \forall x\gamma$: In this case, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff, for each $v \in V$, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta^x_v) \models \gamma$. By (i), for each $v \in V$ there is a j such that $v = z_j$. Hence we have $\beta^x_v(y) = g(z_{f^x_j(y)})$ for each first-order variable y. By inductive hypothesis we have, for each $v \in V$, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta^x_v) \models \gamma$ iff,

for each $j \in \{1,\ldots,n\}$, $(G,g,w) \Vdash tr_n^{f\frac{x}{j}}(\gamma)$ iff $(G,g,w) \Vdash \bigwedge_{i=1}^{n} tr_n^{f\frac{x}{i}}(\gamma) = tr_n^f(\phi)$.

∎

Definition 3.9 (Translation from SO to FHL) *Let X_i be a relation variable of arity r_i. Let $y_{\overline{a}}^{X_i}$, where $\overline{a} \in \{1,\ldots,n\}^{r_i}$ be a state variable (in particular we set $\overline{i} = i,\ldots,i$ for each $i \in \{1,\ldots,n\}$). Let $\phi = Q_1X_1\ldots Q_lX_l\psi$ be a second-order formula, $Q_i \in \{\forall,\exists\}$, where ψ is a first-order sentence. We define*

$$T^n(\phi) = \spadesuit_1 \downarrow y_{\overline{1}}^{X_1}.\ldots\spadesuit_1 \downarrow y_{\overline{n}}^{X_1}.\ldots\spadesuit_l \downarrow y_{\overline{1}}^{X_l}.\ldots\spadesuit_l \downarrow y_{\overline{n}}^{X_l} tr_n(\psi),$$

where $\spadesuit_i = E$ if $Q_i = \exists$ and A otherwise, $1 \leq i \leq l$.

Example 3.2 *Consider the sentence ϕ of Example 3.1. Let ψ be the following second-order sentence:*

$$\psi = \exists R \exists G \exists B(\phi).$$

The sentence ψ above states that there are three sets R, G and B which forms a 3-colouring of elements in the domain of a structure. Hence, ϕ is satisfied in a graph with edges in E iff such graph is 3-colourable. Deciding whether a graph is 3-colourable is a NP-complete problem [Pap03]. We apply the translation T^n for $n = 3$ below. Let

$$\hat{\mathbf{Q}} = E \downarrow y_1^R.E \downarrow y_2^R.E \downarrow y_3^R.E \downarrow y_1^G.E \downarrow y_2^G.E \downarrow y_3^G.E \downarrow y_1^B.E \downarrow y_2^B.E \downarrow y_3^B..$$

We have $T^3(\psi) = \hat{\mathbf{Q}} tr_3(\phi)$. That is,

$$\begin{aligned} T^3(\phi) \;=\; \hat{\mathbf{Q}}\Bigg(& \bigwedge_{i=1}^{3} \oplus\big(@_t y_i^R, @_t y_i^G, @_t y_i^B\big) \wedge \bigwedge_{i=1}^{3}\Bigg[\bigwedge_{j=1}^{3}\big((@_{z_i}\Diamond z_j \wedge \neg @_{z_i} z_j) \rightarrow \\ & \neg((@_t y_i^R \wedge @_t y_j^R) \vee (@_t y_i^G \wedge @_t y_j^G) \vee (@_t y_i^B \wedge @_t y_j^B))\big)\Bigg]\Bigg). \end{aligned}$$

Lemma 3.9 *Let $G = (V, E^G)$ be a graph of cardinality n, $\mathbf{R}_1,\ldots,\mathbf{R}_m$ relations on V with arities $r_1,\ldots,r_m$, g an assignment of state variables, β an assignment of first-order variables and f a function from the set of first-order variables to $\{1,\ldots,n\}$ such that:*

(i) g assigns to each variable z_i a different element in V;

(ii) $g(y^R_{i_1,\ldots,i_k}) = g(t)$ *iff* $(g(z_{i_1}),\ldots,g(z_{i_k})) \in \mathbf{R}$ *for each* $R \in \{R_1,\ldots,R_m\}$;

(iii) $\beta(x) = g(z_{f(x)})$ *for each first-order variable* x.

If $\phi = Q_1X_1\ldots Q_lX_l\psi$ *is a second-order formula in the symbol set*

$$\{E, R_1,\ldots,R_m\},$$

then $(G,\mathbf{R}_1,\ldots,\mathbf{R}_m,\beta) \models \phi$ *iff for all* $w \in V$, $(G,g,w) \models T^n(\phi)$.

Proof. Let $v \in V$ such that $v \neq g(t)$. For each $\mathbf{X}_1$ on V, let $v^{\mathbf{X}_1}_{i_1,\ldots,i_h} = g(t)$ if $(g(z_{i_1}),\ldots,g(z_{i_h}))$ and v otherwise. Then, for each $\mathbf{X}_1$ on V there is an assignment $g^{\mathbf{X}_1}$ defined as

$$g^{\mathbf{X}_1} = g\frac{y^{X_1}_{1,\ldots,1}\cdots y^{X_1}_{i_1,\ldots,i_h}\cdots y^{X_1}_{n,\ldots,n}}{v^{\mathbf{X}_1}_{1,\ldots,1}\cdots v^{\mathbf{X}_1}_{i_1,\ldots,i_h}\cdots v^{\mathbf{X}_1}_{n,\ldots,n}}.$$

On the contrary, given an assignment g' we can find $\mathbf{X}_1$ such that $g' = g^{\mathbf{X}_1}$.

The assignment $g^{\mathbf{X}_1}$ can be described as:

$$g^{\mathbf{X}_1}(s) = \begin{cases} g(t) & \text{, if } s = y^{X_1}_{i_1,\ldots,i_h} \text{ and } (i_1,\ldots,i_h) \in \mathbf{X}_1; \\ v & \text{, for some } v \neq g(t), \text{ if } s = y^{X_1}_{i_1,\ldots,i_h} \text{ and } (i_1,\ldots,i_h) \notin \mathbf{X}_1; \\ g(s) & \text{, otherwise;} \end{cases}$$

It follows that $g^{\mathbf{X}_1}$ and $\mathbf{X}_1$ satisfies (i) and (iii) and

(ii') $g^{\mathbf{X}_1}(y^R_{i_1,\ldots,i_k}) = g^{\mathbf{X}_1}(t)$ iff $(g^{\mathbf{X}_1}(z_{i_1}),\ldots,g^{\mathbf{X}_1}(z_{i_k})) \in \mathbf{R}$ for each

$$R \in \{R_1,\ldots,R_m,X_1\}.$$

Now, we proceed by induction on the size l of the prefix $Q_1X_1\ldots Q_lX_l$. If $l = 0$, then ϕ is first-order and the result follows immediately from Lemma 3.8. Suppose that $l > 0$. If $\phi = \exists X_1\ldots Q_lX_l\psi$, then

$$(G,\mathbf{R}_1,\ldots,\mathbf{R}_m,\beta) \models \phi$$

iff there is $\mathbf{X_1} \subseteq V^{r_1}$ such that

$$(G,\mathbf{R}_1,\ldots,\mathbf{R}_m,\mathbf{X}_1,\beta) \models Q_2X_2\ldots Q_lX_l\psi$$

iff, by the inductive hypothesis, there is $g^{\mathbf{X}_1}$ such that

$$(G, g^{\mathbf{X}_1}, w) \models T^n(Q_2X_2 \dots Q_lX_l\psi)$$

iff

$$(G, g\frac{y^{X_1}_{1,\dots,1} \dots y^{X_1}_{i_1,\dots,i_h} \dots y^{X_1}_{n,\dots,n}}{v^{\mathbf{X}_1}_{1,\dots,1} \dots v^{\mathbf{X}_1}_{i_1,\dots,i_h} \dots v^{\mathbf{X}_1}_{n,\dots,n}}, w) \models T^n(Q_2X_2 \dots Q_lX_l\psi)$$

iff

$$(G, g, w) \models E \downarrow y^{X_1}_{\overline{1}} \dots . E \downarrow y^{X_1}_{\overline{n}} . T^n(Q_2X_2 \dots Q_lX_l\psi) = T^n(\phi).$$

If $\phi = \forall X_1 \dots Q_lX_l\psi$, then

$$(G, \mathbf{R}_1, \dots, \mathbf{R}_m, \beta) \models \phi$$

iff, for all $\mathbf{X_1} \subseteq V^{r_1}$,

$$(G, \mathbf{R}_1, \dots, \mathbf{R}_m, \mathbf{X}_1, \beta) \models Q_2X_2 \dots Q_lX_l\psi$$

iff, by the inductive hypothesis, for all $g^{\mathbf{X}_1}$,

$$(G, g^{\mathbf{X}_1}, w) \models T^n(Q_2X_2 \dots Q_lX_l\psi)$$

iff, for all $v^{\mathbf{X}_1}_{1,\dots,1} \dots v^{\mathbf{X}_1}_{i_1,\dots,i_h} \dots v^{\mathbf{X}_1}_{n,\dots,n}$,

$$(G, g\frac{y^{X_1}_{1,\dots,1} \dots y^{X_1}_{i_1,\dots,i_h} \dots y^{X_1}_{n,\dots,n}}{v^{\mathbf{X}_1}_{1,\dots,1} \dots v^{\mathbf{X}_1}_{i_1,\dots,i_h} \dots v^{\mathbf{X}_1}_{n,\dots,n}}, w) \models T^n(Q_2X_2 \dots Q_lX_l\psi)$$

iff

$$(G, g, w) \models A \downarrow y^{X_1}_{\overline{1}} \dots . A \downarrow y^{X_1}_{\overline{n}} . T^n(Q_2X_2 \dots Q_lX_l\psi) = T^n(\phi).$$

∎

We have the following:

Theorem 3.1 *Let ϕ be a second-order S-sentence, $S = \{E\}$, and G a graph of cardinality n. Then $G \models \phi$ iff*

$$G \models \downarrow t . E \downarrow z_1 \dots . E \downarrow z_n . \left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \wedge T^n(\phi) \right).$$

Proof.

$$(G, g, w) \models\downarrow t.E \downarrow z_1 \ldots E \downarrow z_n. \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right]$$

iff

$$(G, g\frac{t}{w}, w) \models E \downarrow z_1 \ldots E \downarrow z_n. \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\phi) \right]$$

iff there are $v_1, \ldots, v_n \in V$ such that

$$(G, g\frac{tz_1 \ldots z_n}{wv_1 \ldots v_n}, w) \models \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\phi) \right]$$

iff there are $v_1 \neq \ldots \neq v_n \in V$ such that

$$(G, g\frac{tz_1 \ldots z_n}{wv_1 \ldots v_n}, w) \models T^n(\phi)$$

iff, by Lemma 3.9, $G \models \phi$. ∎

The following is the main theorem of this section:

Theorem 3.2 *Let $\mathcal{G}$ be a graph property in the polynomial hierarchy. Then there is a set of sentences $\Phi = \{\phi_1, \phi_2, \ldots\}$ of FHL$\backslash\{NOM, PROP\}$, such that:*

(1) $G \in \mathcal{G}$ iff $G \models \Phi$ iff $G \models \phi_{|G|}$, and

(2) ϕ_n is $O(n^k)$ for some constant k depending only on $\mathcal{G}$.

Proof. Let ψ be a second-order formula expressing $\mathcal{G}$. This formula exists by Fagin's Theorem. Let

$$\begin{aligned} \theta_n \quad = \quad & E \downarrow z_1 \ldots E \downarrow z_n. \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \right] \wedge \\ & A \downarrow z_1 \ldots A \downarrow z_n. \left[\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \rightarrow A \downarrow z. \left(\bigvee_{1 \leq i \leq n} @_{z_i} z \right) \right] \end{aligned} \quad (3.1)$$

The sentence θ_n says that there are exactly n vertices in the frame. We define

ϕ_n as:

$$\phi_n = \theta_n \rightarrow \downarrow t.E \downarrow z_1 \ldots . E \downarrow z_n . \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right].$$

Let $G \in \mathcal{G}$. Let g be any assignment of state variables and w be any point in G. If $G \not\models \theta_n$, then $G \models \phi_n$. It follows that $G \models \phi_n$ for each $n \neq |G|$. Hence, $G \models \Phi$ iff $G \models \phi_{|G|}$. Let $|G| = n$. Then $(G, g, w) \models \phi_n$ iff

$$(G, g, w) \models \downarrow t.E \downarrow z_1 \ldots . E \downarrow z_n . \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right]$$

iff, by Theorem 3.1, $G \in \mathcal{G}$. ∎

The following corollary will be used in the complexity results of Section 3.6.

Corollary 3.1 *If $\phi \in \exists SO$, the existential fragment of second-order logic, then T^n is in HL$(@, \downarrow, E) \backslash \{\downarrow \Box \downarrow, NOM, PROP\}$, that is, the fragment of HL$(@, \downarrow, E)$ without, nominals, propositional symbols and the patterns $\downarrow \Box \downarrow$ and $\downarrow A \downarrow$, that is, an $\downarrow$ inside the scope of a universal modality A or $\Box$, which in turn is inside the scope of other $\downarrow$.*

In the following section, we will show that we can discharge the global modalities if we consider connected frames with loops.

3.5 Connected Frames with Loops

Let FHL$\backslash\{E, NOM, PROP\}$ be the hybrid logic without the modality E and without nominals. One can see that an analogous to Theorem 3.2 does not hold for FHL$\backslash\{E, NOM, PROP\}$. Let us define the disjoint union of frames $G = (V, E)$ and $G' = (V', E')$ such that $V \cap V' = \emptyset$ as the frame $G'' = (V \cup V', E \cup E')$. Similarly, for models $\mathcal{M} = (G, \mathbf{V})$ and $\mathcal{M}' = (G', \mathbf{V}')$ the disjoint union is defined as $\mathcal{M}'' = (G'', \mathbf{V} \cup \mathbf{V}')$, where $(\mathbf{V} \cup \mathbf{V}')(p) = \mathbf{V}(p) \cup \mathbf{V}'(p)$.

Its is not difficult to show that:

Lemma 3.10 *Frame validity and model (global) satisfaction for sentences from FHL$\backslash\{E, NOM, PROP\}$ are invariant under disjoint union.*

Proof. Let $G = (V, E)$ and $G' = (V', E')$ be two frames such that $V \cap V'$, and let G'' be the disjoint union of G and G'. Let $\mathcal{M} = (G, \mathbf{V})$ and $\mathcal{M}' = (G', \mathbf{V}')$ and $\mathcal{M}''$ the corresponding disjoint union. Let ϕ be a formula of FHL$\backslash\{E, NOM, PROP\}$. It follows easily by induction on ϕ that,

- $\mathcal{M} \Vdash \phi$ iff $\mathcal{M}', g, v \Vdash \phi$ for all $v \in V$ and all assignment g which maps free state-variables occurring in ϕ in elements of V, and
- $\mathcal{M}' \Vdash \phi$ iff $\mathcal{M}', g, v' \Vdash \phi$ for all $v' \in V'$ and all assignment g which maps free state-variables in elements of V'.

Hence, if ϕ is a sentence, it follows that $\mathcal{M} \Vdash \phi$ and $\mathcal{M}' \Vdash \phi$ iff $\mathcal{M}'' \Vdash \phi$. ∎

It follows from Lemma 3.10 above that an analogous to Theorem 3.2 does not hold for FHL$\backslash\{E, NOM, PROP\}$.

Corollary 3.2 *There are graph properties in P for which there is no set $\Phi = \{\phi_1, \phi_2, \ldots\}$ from sentences in FHL$\backslash\{E, NOM, PROP\}$ which satisfies condition (1) from Theorem 3.2 above.*

Proof. Connectivity is one such a property. Suppose there is such set. Let G and G' be connected frames of size n, then G" be the disjoint union of G and G'. Then $G'' \nVdash \phi_{2n}$, hence $G'' \Vdash \neg\phi_{2n}$ since ϕ_{2n} has no propositional symbol, and by Lemma 3.10 we have that $G \Vdash \neg\phi_{2n}$ or $G' \Vdash \neg\phi_{2n}$, which contradicts condition (1). ∎

However, Theorem 3.2 still hold if we restrict ourselves to connected frames with loops. Consider the following translation from formulas in SO into formulas in FHL$\backslash\{E, NOM, PROP\}$:

Definition 3.10 *Let $\phi = Q_1 X_1 \ldots Q_l X_l \psi$ be a second-order formula where ψ is a first-order sentence. We define*

$$\hat{T}^n(\phi) = \spadesuit_1 \downarrow y_{\overline{1}}^{X_1} \ldots . \spadesuit_1 \downarrow y_{\overline{n}}^{X_1} \ldots . \spadesuit_l \downarrow y_{\overline{1}}^{X_l} \ldots . \spadesuit_l \downarrow y_{\overline{n}}^{X_l} tr_n(\psi),$$

where $\spadesuit_i = (\Diamond\Diamond^{-1})^n = \underbrace{\Diamond\Diamond^{-1} \ldots \Diamond\Diamond^{-1}}_{n}$ if $Q_i = E$ and $(\Box\Box^{-1})^n$ otherwise.

Lemma 3.11 *Let $G = (V, E^G)$ be a connected graph with loops on each vertex, $\mathbf{R}_1, \ldots, \mathbf{R}_m$ relations on V with arities $r_1, \ldots, r_m$, g an assignment of state variables, β an assignment of first-order variables and f a function from the set of first-order variables to $\{1, \ldots, n\}$ such that:*

(i) g assigns to each variable z_i a different element in V;

(ii) $g(y^R_{i_1,\ldots,i_k}) = g(t)$ iff $(g(z_{i_1}),\ldots,g(z_{i_k})) \in \mathbf{R}$ for each $R \in \{R_1,\ldots,R_m\}$;

(iii) $\beta(x) = g(z_{f(x)})$ for each first-order variable x;

If $\phi = Q_1X_1 \ldots Q_lX_l\psi$ is a second-order formula in the symbol set

$$\{E, R_1, \ldots, R_m\},$$

then

$$(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi \text{ iff for all } w \in V,\ (G, g, w) \models \hat{T}^n(\phi).$$

Proof. Analogous to the proof of Lemma 3.9 ∎

Theorem 3.3 *Let ϕ be a second-order sentence and G a connected graph of cardinality n with loops. Then $G \models \phi$ iff*

$$G \models\downarrow t.(\Diamond\Diamond^{-1})^n \downarrow z_1 \ldots (\Diamond\Diamond^{-1})^n \downarrow z_n. \left(\bigwedge_{1\le i<j\le n} @_{z_i}\neg z_j \wedge \hat{T}^n(\phi) \right).$$

Proof. Analogous to the proof of Theorem 3.1 ∎

Theorem 3.4 *Let $\mathcal{G}$ be a property of connected graphs with loops in the polynomial hierarchy. Then there is a set of sentences*

$$\Phi = \{\phi_1, \phi_2, \ldots\}$$

of FHL$\backslash\{E, NOM, PROP\}$, such that:
(1)for all connected G, $G \in \mathcal{G}$ iff $G \models \Phi$ iff $G \models \phi_{|G|}$, and
(2) ϕ_m is $O(n^k)$ for some constant k depending only on $\mathcal{G}$.

Proof. Let ψ be a second-order formula expressing $\mathcal{G}$. Let

$$\begin{aligned} \theta_n \quad = \quad & (\Box\Box^{-1})^n \downarrow z_1 \ldots (\Box\Box^{-1})^n \downarrow z_n.[\\ & \bigwedge_{1\le i<j\le n} @_{z_i}\neg z_j \rightarrow (\Box\Box^{-1})^n \downarrow z. \left(\bigvee_{1\le i\le n} @_{z_i} z \right)]. \end{aligned}$$

We define ϕ_n as:

$$\phi_n = \theta_n \to \downarrow t.(\Diamond\Diamond^{-1})^n \downarrow z_1 \ldots (\Diamond\Diamond^{-1})^n \downarrow z_n.[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \Bigg]. \tag{3.2}$$

$$\tag{3.3}$$

The remaining of the proof is similar to the proof of Theorem 1.2 ∎

In the following section, we will show that the prefix pattern of the prenex form is closely related with the degrees of the polynomial hierarchy.

3.6 Polynomial Hierarchy

In [tCF05], it is proved that the model checking for the FHL$\setminus \downarrow \Box \downarrow$ fragment is NP-complete. The translation given in Section 3.4 above can be used to produce hybrid formulas of polynomial size using formulas of second-order logic. This leads to a alternative proof that the model checking problem for the fragment FHL$\setminus \downarrow \Box \downarrow$ is hard for NP, since there is a polynomial reduction for any instance of a NP problem to the model checking of FHL$\setminus \downarrow \Box \downarrow$.

Theorem 3.5 ([tCF05]) *The model checking problem for $\sigma^1 \subseteq$ FHL$\setminus \downarrow \Box \downarrow$ is NP-hard.*

Proof. Let $\mathcal{G}$ be an NP-complete graph property. Let ϕ be an $\exists$SO sentence which express $\mathcal{G}$. By Fagin's Theorem [Fag74a], such sentence exists. Let G be a graph. Let $T^{|G|}(\phi)$ be as defined in Definition 3.9. It is easy to see that $T^{|G|}(\phi)$ can be constructed from ϕ in time polynomial in $|G|$. Now, $(G, T^{|G|}(\phi))$ is an instance of the model checking for FHL$\setminus \downarrow \Box \downarrow$. By Theorem 3.2, the model-checker returns *true* for $(G, T^{|G|}(\phi))$ iff $G \in \mathcal{G}$. Hence, the model checking for FHL$\setminus \downarrow \Box \downarrow$ is hard for NP. ∎

Actually, for each degree of the polynomial hierarchy, there is a syntactically defined fragment of HL whose model checking problem is hard.

Theorem 3.6 *The model checking problem for σ^i (resp. π^i) is Σ_i^p-hard (resp. Π_i^p-hard).*

Proof. Analogous to the proof of Theorem 3.5 above, since each graph property in Σ_i^p (resp. Π_i^p) can be expressed by a SO sentence in Σ_i^1 (resp. Π_i^1), and $T^n(\phi)$ can always be constructed in time polynomial in n, for a fixed $\phi \in$ SO. ∎

Also, the model checking for σ^i and π^i are in Σ_i^p and Π_i^p, respectively.

Theorem 3.7 *The model checking problem for σ^i (resp. π^i) is in Σ_i^p (resp. Π_i^p).*

Proof. We proceed by induction on i. For the sake of simplicity, we consider only the modalities $\Diamond$ and $\Box$ in the prefix, but the proof is analogous for E and A. In [tCF05], it is shown that the model checking for FHL$\setminus \downarrow \Box \downarrow$, which contains σ^1, is in NP. It follows that π^1 is in co-NP. Now, let $\overline{q}\phi$ be a sentence in σ^{i+1}, where ϕ is in π^i. It follows that $\overline{q}$ has the form

$$\overline{q} = \Diamond^{k_1} \downarrow x_1.\Diamond^{k_2} \downarrow x_2. \ldots \Diamond^{k_m} \downarrow x_m.\Diamond^{k_{m+1}}.$$

Let M be a finite model, g be an assignment of state variables and w a point in W. By inductive hypothesis, suppose that the model checking problem for π^i is in Π_i^p. We can use nondeterministic Turing machine to existentially guess values v_j for x_j among the points in W which are reachable in $\sum_{i=1}^{j} k_i$ steps from w, with respect to the accessibility relation R, in polynomial nondeterministic time, and we can existentially guess points w' reachable in $\sum_{i=1}^{m+1} k_i$ steps from w in polynomial nondeterministic time also. Finally, we can use an oracle for the model checking of π^i with the input

$$(M, g\frac{v_1 \ldots v_m}{x_1 \ldots x_m}, w'), \phi).$$

By inductive hypothesis, such an oracle is in Π_i^p. As the existential guesses initially performed can be made in (existential) nondeterministic polynomial time, the model checking for σ^{i+1} is in Σ_{i+1}^p.

The proof is analogous for the model checking of π^{i+1}. ∎

Corollary 3.3 *Let $\Phi = \{\phi_1, \phi_2, \ldots\}$ be such that each ϕ_i can be constructed in time polynomial on i and each ϕ_i is in π^j (resp. σ^j). Then the graph property $\mathcal{G}$ defined as:*

$$G \in \mathcal{G} \text{ iff } G \models \phi_{|G|}$$

is in Π_j^p (resp. Σ_j^p).

From Theorems 3.6 and 3.7 we have:

Corollary 3.4 *The model checking problem for σ^i (resp. π^i) is Σ_i^p-complete (resp. Π^i-complete).*

Corollary 3.5 *The frame checking problem for $\sigma^i \backslash PROP$ (resp. $\pi^i \backslash PROP$) is Σ_i^p-complete (resp. Π^i-complete).*

3.7 Conclusions

In this chapter we presented some results about the expressibility of properties in the polynomial hierarchy by sequences of formulas of hybrid logic. In [BS09] it is presented a sequence ϕ_1, ϕ_2, ... of HL that expresses the property that a graph is hamiltonian, in the sense that a graph of size n satisfies ϕ_n iff it is hamiltonian. We show that this can be done for every property in NP, actually any property in the polynomial hierarchy PH (Theorem 3.2). Beside this, the size of formulas is bounded by a polynomial on the size of the graph. This leads to an alternative proof of the NP-hardness (Theorem 3.5), different from the one presented in [tCF05]. If we do not use the global modalities E and A, we can no more express properties in PH, by sequences of formulas, actually, relative simple properties like connectivity cannot be expressed by sets of sentences in FHL$\{E, NOM, PROP\}$. However, if we consider only connected frames with loops, the result still holds (Theorem 3.4). We also defined the fragments π_i and σ_i of HL and showed that the model-checking problem for those fragments are complete problems for the corresponding degree of the polynomial hierarchy.

In the next chapter, we will introduce the bounded-degree second-order logic and study its descriptive complexity.

Chapter 4

Bounded-Degree SO

In this chapter we introduce the Bounded-Degree Second-Order Logic (BDSO), a restriction of second-order logic where second-order quantifiers range over bounded-degree relations, and study its descriptive complexity. Based on previous works from Olive and Grandjean [GO98] and Schwentick et al. [DLS98], we show that BDSO captures the class ALIN of queries computable in an alternating random access machine in linear time and a fixed number of alternations depending only on the query. We also extend BDSO with the transitive closure operator on relations of bounded degree and we show that it captures the classes of unary structures which can be accepted by a nondeterministic random access machine using linearly many registers provided that the values stored in each register is bounded by a linear function in the cardinality of the input structure.

4.1 Introduction

The relation between the expressive power of logics and complexity classes is one of the main issues of descriptive complexity, a branch of finite model theory and computational complexity. From the perspective of computational complexity, we are interested in the characterization of the expressive power of logics over finite models in terms of complexity classes. From the point of view of finite model theory, we are interested in logics whose languages are able to express all problems in a complexity class, as well as the kind of logical operators (e.g. quantifiers, fixed-point operators, etc.) these logical languages need to express those problems.

One of the first results in this field is Fagin's Theorem [Fag74b], which states that second-order logic captures the class NP. Each formula ϕ of second-order logic defines a class of finite structures which satisfies ϕ. Using a standard code of structures into strings, the class of models of ϕ gives rise to a language and the corresponding decision problem over strings. This construction give us a relation between sentences in a logical language, classes of structures and languages. A natural question arises, namely what is the computational complexity of the language defined by the formula ϕ? Fagin showed that each formula of second-order logic defines a language which can be accepted by a nondeterministic Turing machine in polynomial time and, conversely, if a class of structures corresponds to a language in NP, then it can be defined by a formula of SO. We say that SO captures NP. Before the result of Fagin, Büchi established the equivalence between regular languages and monadic second-order logic on strings. That is, a set of strings is regular iff it is the set of strings which are models of a monadic second-order sentence [Büc60].

Since the remarkable result of Fagin, many other capturing results were found, for example, the characterization of the class P on ordered structures with least fixed-point logic [Imm82, Var82], the equivalence between first-order logic with deterministic transitive closure operator FO(DTC) and deterministic logspace [Imm83], the equivalence between P and SO-Horn [Grä92], between PSPACE and partial fixed-point logic [AV91, Imm99], etc..

In [Lyn92], Lynch investigates the expressive power of restrictions of second-order logic. Lynch is concerned with the problem of finding good lower-bounds for NP-complete problems. He is intended to find the most restricted language where the problems of some complexity class can be expressed and then show a lower bound for certain problem by showing that it cannot be expressed in this fragment, and hence does not belong to the corresponding class [Lyn92]. In particular, Lynch shows that any language which can be decided in nondeterministic linear time $O(n)$ can be expressed by a sentence of existential monadic second-order logic with quantifier prefix $\Sigma_1^1\Pi_1^0$ (a block of second-order existential quantifiers followed by a block of universal first-order quantifiers) using the addition operator as a built-in operator [Lyn92]. In [GO98], Grandjean and Olive improve the result of Lynch by showing that even the class NLIN can be expressed in this fragment. The class NLIN is the class of problems which can be solved in linear time by a nondeterministic random access machine (we give more details in Section 4.2 below). Many NP- complete problems are known to belong to NLIN, but just a few are known to be in NTIME($O(n)$).

In this chapter, we investigate the expressive power of a restriction of second-order logic. The *bounded-degree second-order logic* BDSO is the restriction of SO where second-order quantifiers range over relations of bounded degree only. A relation has bounded degree if the corresponding Gaifman graph has bounded degree (see Section 4.3 below). We will show that BDSO captures the ALIN hierarchy of problems on unary structures which can be solved in a random access machine in alternating linear time. In Section 4.2 we present the random access machine model and the results of Lynch and Grandjean and Olive. In Section 4.3 we define the BDSO logic. In Section 4.4 we show the equivalence between BDSO and MSO(f), a restriction of second-order logic with quantification over unary functions only. This equivalence is the capturing result we are looking for. Some results in Section 4.4 where independently proved in [DLS98]. Finally, in Section 4.5 we extend BDSO with transitive closure operator on high-order relations on relations of bounded degree. We show that it captures the class of unary structures which can be recognized by an NRAM using linearly many registers provided that the values stored in the registers during a computation are bounded by a linear function in the cardinality of the input structure.

In this chapter, models are finite.

4.2 Unary Second-Order Logic

In this Section we will present some results about the expressive power of the unary fragment of second-order logic MSO(f). MSO(f) is the fragment of SO where second-order quantifiers are restricted to monadic (unary) function.

In this Chapter we use lower case Roman letters $f, g, \ldots$ to to represent functions and the correspond upper case Roman letters $F, G, \ldots$ for their graphs.

Definition 4.1 (Unary Second-Order Logic) *The logic* MSO(f) *extends first-order logic by allowing quantification of unary functions. That is, first-order formulas are* MSO(f) *formulas and, if* α *and* β *are* MSO(f) *formulas,* x *a first-order variable and* F *a binary relation variable, then* $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $\neg\alpha$, $\exists x\alpha$ *and* $\exists^{tf} F\alpha$ *are formulas of* MSO(f).

Let $\mathfrak{I}$ *be an interpretation and* $\exists^{tf} F\phi$ *a* MSO(f) *formula. We define the*

satisfaction relation between $\mathfrak{I}$ *and* $\exists^{tf} F\phi$ *as*

$$\mathfrak{I} \models \exists^{tf} F\phi$$

iff

exist $\mathbf{F} \subseteq A^2$ *which is the graph of a function and such that* $\mathfrak{I}\frac{\mathbf{F}}{F} \models \phi$.

The existential fragment $\exists\mathrm{MSO}(f)$ *of* $\mathrm{MSO}(f)$ *is composed of the formulas of the form* $\exists^{tf} F_1 \ldots \exists^{tf} F_n\phi$ *where* ϕ *is a first-order formula.*

$\mathrm{MSO}(f)$ is at least as expressive as MSO, monadic second-order logic. That is because monadic relations can be coded as its characteristic function. Let P be a monadic relation variable and $\psi = \exists P\phi(P)$ be an MSO formula. An interpretation $\mathfrak{I}$ satisfies ψ iff there is a monadic relation $\mathbf{P} \subseteq A$ such that $\mathfrak{I}\frac{\mathbf{P}}{P} \models \phi(P)$. Suppose that $\mathfrak{I}\frac{\mathbf{P}}{P} \models \phi(P)$ and $|A| > 1$. Let $\mathbf{F} \in A^2$ be the graph of a function such that $(a, a) \in \mathbf{F}$ iff $a \in \mathbf{P}$. It is not difficult to see that

$$\mathfrak{I}\frac{\mathbf{F}}{F} \models \phi(F\_),$$

where $\phi(F\_)$ is obtained from $\phi(P)$ by substituting[1] Ftt for Pt for each term t. Hence $\mathfrak{I} \models \exists^{tf} F\phi(F\_)$.

Similarly, let $\mathfrak{I} \models \exists^{tf} F\phi(F\_)$. Then there is the graph of a function $\mathbf{F} \subseteq A^2$ such that $\mathfrak{I}\frac{\mathbf{F}}{F} \models \phi(F\_)$. Let $\mathbf{P} \subseteq A$ such that $a \in \mathbf{P}$ iff $(a, a) \in \mathbf{F}$. Then

$$\mathfrak{I}\frac{\mathbf{P}}{P} \models \phi(P),$$

and hence $\mathfrak{I} \models \exists P\phi(P)$. It follows that

$$\exists P\phi(P) \equiv \exists^{tf} F\phi(F\_).$$

As pointed out in [DLS98], it can be shown that $\mathrm{MSO}(f)$ is strictly more expressive than MSO. That is because the EVEN query can be expressed in $\mathrm{MSO}(f)$ but not in MSO (see [Lib04]), as we will see below.

Definition 4.2 (EVEN) *Let* S *be a symbol set. The* $\mathrm{EVEN}(S)$ *is the set of* S*-structures whose cardinality is even. We write* EVEN *instead of* $\mathrm{EVEN}(\emptyset)$.

[1] In this chapter, substitutions of subformulas should be carried out with the appropriate renaming of bound variables in order to avoid erroneous bound of variables in the introduced subformulas.

In order to show that EVEN can be expressed in MSO(f), let us first show that it is possible to simulate quantification over partial functions in MSO(f). Given a partial function $f : A \to B$, the *graph of* f is the set $gr(f) = \{(a, b) \in A^2 | b = f(a)\}$. We denote by $dom(f)$ the domain of f, that is, the subset $\{a \in A |$exists b in B such that $(a, b) \in gr(f)\}$ of A on which f is defined. The *image of* f is the set $img(f) = \{b \in B |$exists $a \in A$ such that $(a, b) \in gr(f)\}$. A *function* f *on* A *is a function in* $A \to A$.

We introduce now the quantifier $\exists^{pf}$ and define the satisfaction relation between interpretations and formulas of the form $\exists^{pf} F\phi$ as

$$\mathfrak{I} \models \exists^{pf} F\phi$$

iff

there is $\mathbf{F}$ which is the graph of a partial function and $\mathfrak{I}^{\frac{F}{\mathbf{F}}} \models \phi$.

A partial function is either strictly partial (which means that there is some element in the domain that the function does not map to any element) or it is total. In the case a function $f : A \to A$ is strictly partial, for finite A, then it is not surjective.

Now let $a \in A$ and $f : A \to A$ be a total function. We can define the graph of a partial function as

$$\mathbf{F}' = \{(c, d) \in A^2 | d \neq a, \text{ and } (c, d) \in gr(f)\}.$$

This function is either total, in the case a does not belong the image of f, or it is strictly partial. The element a plays the role of an undefined value. In both cases it is not a surjective function. The following formula defines $\mathbf{F}'$ in the case F is interpreted as the graph of f and z is interpreted as a

$$F'(F, z, x, y) = y \neq z \wedge Fxy.$$

Conversely, let f' be a nonsurjective partial function on A. Then there is a total function f on A and $a \in A$ such that $\mathbf{F}'$ is the graph of f'. It suffices to choose an a that does not belong to the image of f' and an f whose graph contains the graph of f' and that maps any other element that does not belong to $dom(f')$ to a.

Now, let $\phi(F)$ be a formula where the binary relation variable F occurs free.

Let $\phi(F')$ be obtained from $\phi(F)$ by substituting Ftt' with

$$F'(F, z, t, t') = y \neq z \wedge Ftt'$$

for every atom Ftt' in $\phi(F)$. Since any partial function is nonsurjective or total, we have the following equivalence:

$$\exists^{pf} F\phi(F) \equiv \exists^{tf} F\exists z(\phi(F') \vee \phi(F)).$$

This show how to simulate quantification of partial functions using quantification of total functions.

On the other hand, we can simulate quantification of total functions by quantifying partial functions relativized to a statement which says that the function is total. For example, the following formula says that the relation represented by F is total:

$$Total(F) = \forall x\exists yFxy.$$

If $\phi(F)$ is a formula where F occurs free, we can express the fact that exists a total function which satisfies $\phi(F)$ with the formula

$$\exists^{tf} F\phi(F) \equiv \exists^{pf} F(Total(F) \wedge \phi(F)).$$

It follows that quantifying partial functions has the same expressive power as quantifying total functions.

In order to show that a structure has even cardinality, it is sufficient to show that there is an injective partial function where every element in the universe of the structure is either in the domain or in the image of the function. The following sentence states the existence of such a function:

$$\phi_{EVEN} = \exists^{pf} F(injective(F) \wedge \theta(F)),$$

where

$$injective(F) = \forall x\forall y\forall z((Fxz \wedge Fyz) \rightarrow x = y) \tag{4.1}$$

says that F represents an injective function and

$$\theta(F) = \forall x(\exists yFxy \oplus \exists yFyx)$$

says that every element is either in the domain or in the image of the F. A structure satisfies ϕ_{EVEN} iff it has even cardinality.

Lemma 4.1 MSO(f) *is strictly more expressive than* MSO.

In [GO98], Grandjean and Olive study the expressive power of the existential fragment of MSO(f) on classes of unary functions. They show that the classes of unary functions computable by an NRAM in nondeterministic linear time are exactly those expressible in a particular fragment of existential MSO(f).

Definition 4.3 (Unary Structures) *A unary structure is a structure on some signature* $\{>, f_1, \ldots, f_l, c_1, \ldots, c_s\}$*, where* $>$ *is a binary relation interpreted as a linear order, each* f_i *is a unary function symbol and each* c_j *is a constant symbol.*

In [GO98], Grandjean and Olive study the NLIN complexity class. They are motivated by the search for a suitable definition of linear time computation [Grä90, GS89]. Grandjean introduces a nondeterministic random access machine (NRAM) which has a set of registers $R_0, R_1, \ldots$ and a set of accumulators $a_0, a_1, \ldots$. A program for an NRAM is a finite sequence of instructions $I(1)$, $I(2)$, ..., $I(r)$ of the following types (where X, Y and Z are either R or a):

(1) $X_i := j$
(2) $X_i := Y_j$
(3) $X_i := Y_j + 1$
(4) $X_i := Y_j \dot{-} 1$
(5) $X_i := Y_{Z_j}$
(6) $X_{Y_j} := Z_i$
(7) *guess* (X_i)
(8) *goto* $I(i_0)$ *or* $I(i_1)$
(9) *if* $X_i = X_j$ *then goto* $I(i_0)$ *else goto* $I(i_1)$
(10) *accept*
(11) *reject*

where i, j, i_0, j_0 natural numbers with $1 \leq i_0, i_1 \leq r$.

An input for such an NRAM is a function $f : m \to m \backslash \{0\}$. In the beginning of the computation, each value $f(i)$ is stored in the register R_i, $i < m$. The other registers and the accumulators hold the value 0. The instruction $X_i := Y_j \dot{-} 1$ stores in R_i the maximum between the value stored in Y_j minus 1 and 0. The instructions (7) and (8) are nondeterministic: *guess* (X_i) stores the value of any register in X_i and *goto* $I(i_0)$ *or* $I(i_1)$ performs the instructions $I(i_0)$ or $I(i_1)$

nondeterministically. A function is *accepted* by a program if the instruction *accept* is reached at some point of the computation. Otherwise it is rejected. A set of functions is *recognized* by a program if it is exactly the set of functions accepted by the program. Let $S = \{f_1, \ldots, f_k, c_1, \ldots, c_r\}$ be a functional symbol set and $\mathfrak{A}$ an S-structure of cardinality n. $\mathfrak{A}$ can be regarded as an input for a NRAM where the values of the functions $f_1, \ldots, f_k$ are stored in the first nk registers of the NRAM followed by the values of the r constants $c_1, \ldots, c_r$.

In the following we will give some examples of NRAM programs. We will use other instructions that can be defined using those above. Note that the instruction *goto* $I(k)$, that makes de NRAM to jump to the instruction k, for some natural number k, can be defined as *goto* $I(k)$ *or* $I(k)$. The binary operator *monus*, denoted by $\dot{-}$, is defined as $a \dot{-} b = max\{a - b, 0\}$. The addition instruction $X_i := Y_j + Z_k$ and the monus instruction $X_i := Y_j \dot{-} Z_k$ on registers and accumulators can be defined using the instructions above and can be executed in time linear on the values of registers or accumulators Y_j and Z_k. The assignment of values linear in the input size $X_i := cm$ can be computed in linear time in the input size.

Example 4.3 Let $S = \{f\}$ be a symbol set. The following program evaluates the formula

$$\exists x f(x) = x$$

in linear time:

	Evaluate the formula $\exists x f(x) = x$**.**	
1.	$a_0 := 0$	
2.	$a_1 := 0$	
3.	$a_2 := R_{a_1}$	
4.	$if\ a_2 = a_0\ then\ goto\ I(9)\ else\ goto\ I(5)$	\\ *checks if the algorithm* \\ *reached the register* R_n
5.	$if\ a_2 = a_1\ goto\ I(6)\ else\ goto\ I(7)$	
6.	*accept*	
7.	$a_1 := a_1 + 1$	
8.	*goto* $I(3)$	
9.	*reject*	

The accumulator a_1 holds the address (the index) of a register. The accumulator a_2 holds the value stored in the register of index a_1. The value of a_0 is not

changed, that is, is equal to 0. Given an S-structure $\mathfrak{A} = (A, f)$ of cardinality n, the registers $R_0, \ldots, R_{n-1}$ hold the values of the function f. By definition, all values taken by f are nonzero. Hence, the first register that stores the value 0 is R_n. The program checks, for each value a_1 (beginning from 0 and being iteratively increased by 1) such that R_{a_1} is nonzero, if R_{a_1} (whose value is stored in a_2) is equal to a_1. If so, the program accepts the input, otherwise it increases the value of the a_0 by 1 and repeat the process.

Example 4.4 Let $S = \{>, f_1, \ldots, f_k, c_1, \ldots, c_r\}$ be a symbol set and $\mathfrak{A}$ an S-structure of cardinality n. Let ϕ be a quantifier free S-sentence. Without loss of generality let us suppose that the only connectives in ϕ are $\neg$ and $\wedge$. Let us show how to write a program P_ϕ that evaluates ϕ on $\mathfrak{A}$. For each sub-formula ψ of ϕ we will denote by R_ψ an accumulator and Q_ψ a subroutine that stores in R_ϕ the value 1 if $\mathfrak{A} \models \psi$ and 0 otherwise. We define Q_ψ recursively as:

	$Q_{c_i=c_j}$ **Evaluate the formula** $c_i = c_j$**.**
1.	$a_0 := 0$
2.	$if\ R_{c_i} = R_{c_j}\ then\ goto\ I(3)\ else\ I(4)$
3.	$a_0 := 1$
4.	$R_{c_1=c_j} := a_0$

	$Q_{c_i<c_j}$ **Evaluate the formula** $c_i < c_j$**.**
1.	$a_0 := 1$
2.	$a_1 := R_{c_i} - R_{c_j}$
3.	$a_2 := R_{c_j} - R_{c_i}$
4.	$if\ a_1 = a_2\ then\ goto\ I(5)\ else\ goto\ I(6)$
5.	$a_0 := 0$
6.	$R_{c_i<c_j} := a_1$

	$Q_{f_l(c_i)=f_h(c_j)}$ **Evaluate the formula** $f_l(c_i) = f_h(c_j)$**.**
1.	$a_0 := ln$
2.	$a_1 := a_0 + R_{c_i}$
3.	$a_2 := R_{a_1}$
4.	$a_1 := a_0 + R_{c_j}$
5.	$a_2 := R_{a_1}$
6.	$a_3 := 0$
7.	$if\ a_1 = a_2\ then\ goto\ I(8)\ else\ goto\ I(9)$
8.	$a_3 := 1$
9.	$R_{f_l(c_i)=f_h(c_j)} := a_3$

	$Q_{\alpha\wedge\beta}$ **Evaluate the formula** $\alpha \wedge \beta$**.**
1.	$a_0 := 0$
2.	$if\ R_\alpha = a_0\ then\ goto\ I(5)\ else\ goto\ I(3)$
3.	$if\ R_\beta = a_0\ then\ goto\ I(5)\ else\ goto\ I(4)$
4.	$a_0 := 1$
5.	$R_{\alpha\wedge\beta}$

	$Q_{\neg\alpha}$ **Evaluate the formula** $\neg\alpha$**.**
1.	$if\ R_\alpha = 0\ then\ goto\ I(2)\ else\ goto\ I(3)$
2.	$R_{\neg\alpha} := 1$
3.	$R_{\neg\alpha} := 0$

The program P_ϕ that evaluates ϕ on a S-structure $\mathfrak{A}$ can be constructed using the subroutines Q_ψ for each subformula ψ of ϕ. For each subformula ψ of ϕ we define a program P_ψ that evaluates ψ on $\mathfrak{A}$ by concatenating programs for its subformulas. First we define P'_ϕ recursively in the following way. If ψ is atomic, $P'_\psi = Q_\psi$ and

$$P'_{\alpha\wedge\beta} = \begin{matrix} P'_\alpha \\ P'_\beta \\ Q_{\alpha\wedge\beta} \end{matrix} \quad \text{and } P'_{\neg\alpha} = \begin{matrix} P'_\alpha \\ Q_{\neg\alpha} \end{matrix}.$$

Concatenation of programs should be carried out with the appropriated modification of the *goto* commands since line numbers will change. Finally, suppose

that P'_ϕ has l lines. We have

$$P_\phi = \begin{array}{ll} & P'_\phi \\ l+1. & if\ R_\phi = 0\ then\ goto\ I(l+2)\ else\ goto\ I(l+3) \\ l+2. & accept \\ l+3. & reject \end{array}$$

Definition 4.4 (NLIN) *The complexity class NLIN is the class of sets of unary structures which can be recognized by some program running on a NRAM in nondeterministic linear time.*

The main result of [GO98] states that a class of unary structures is in NLIN iff it can be expressed in a very restricted fragment of MSO(f).

Theorem 4.1 (Grandjean and Olive 1998) *Let $\mathcal{C}$ be a set of unary structures in a signature $S = \{>, f_1, \ldots, f_l, c_1, \ldots, c_s\}$. Then $\mathcal{C}$ is in NLIN iff exists an $\exists$MSO(f) sentence $\exists F_1 \ldots \exists F_n \forall x \phi(x, \overline{F})$, where $\phi(x, \overline{F})$ is a quantifier-free formula, such that, for each unary S-structure $\mathfrak{A}$,*

$$\mathfrak{A} \in \mathcal{C} \text{ iff } \mathfrak{A} \models \exists F_1 \ldots \exists F_n \forall x \phi(x, \overline{F}).$$

The formula $\phi(x, \overline{F})$ is constructed from an NRAN program which accept a set of unary structures in linear time. It is defined so that a tuple of functions $\overline{\mathbf{F}}$ on a unary S-structure $\mathfrak{A}$ satisfies $\phi(x, \overline{F})$ iff there is an accepting run of linear size in the associated computation tree. The tuple of relation symbols $\overline{F}$ represents a run of linear size in the computation tree. The formula $\phi(x, \overline{F})$ is the conjunction of two formulas $\alpha(x, \overline{F})$ and $\beta(x, \overline{F})$. The first is satisfied by a tuple of relations $\overline{\mathbf{F}}$ iff it represents a run of linear size. The second tests if the run is an accepting run.

We can extend the definition of NRAM in order to allow universal nondeterministic behaviour. An *Alternating RAM* (ARAM) is able to perform both existential and universal nondeterministic computations. We substitute the instruction (7) with the instructions

(7') $\exists guess\ (X_i)$

(7") $\forall guess\ (X_i)$

The instruction $\exists guess\ (X_i)$ corresponds to an existential nondeterministic choice of the value for the register X_i among the values stored in the registers of the machine at the time this instruction is executed. If the the machine carries

on an accepting computation with some of these values in X_i, then the input is accepted by the program. Similarly, the $\forall guess\ (X_i)$ corresponds to a universal nondeterministic choice of value for the register X_i among the values stored in the registers of the machine. After the instruction $\forall guess\ (X_i)$ is executed, the input is accepted if and only if the machine carries on an accepting computation for each possible value of the register among those guessed by the machine. With this definition, we can define the class ALIN of sets of function accepted by a RAM in alternating nondeterministic linear time.

Definition 4.5 *A set of unary functions belongs to $ALIN^k$ if it is recognized by a ARAM in linear time using at most k alternations between existential and universal guesses. We define the class ALIN= $\bigcup_{i\in N} ALIN^k$.*

It follows from Theorem 4.1 that MSO(f) captures ALIN on unary functions. We proved the following:

Theorem 4.2 *A set of unordered unary structures is in $ALIN^k$ iff it can be expressed by a sentence in* MSO(f) *whose quantifier prefix has k blocks.*

Proof. The proof is similar to the one for Theorem 4.1 in [GO98]. Let ϕ be a MSO(f) in prenex normal form. In [DLS98], it is shown that existential quantification of linear order relations can be defined using existential quantification of unary function. Hence we can suppose the existence of an order relation. In order to evaluate ϕ on some unary structure, we can guess the values of first-order variable and unary functions occurring in the prefix of ϕ using the $\forall guess(R_i)$ and $\exists guess(R_i)$ instructions. For example, to universally guess a function we just need to guess the value of the function for each element of the domain. This can be done with m calls to the $\forall guess(R_i)$ instruction, where m is the size of the domain. Hence, guessing the variables in the quantifier prefix can be done in linear alternating time. Once the variables in the quantifier prefix are chosen, we can easily evaluate the quantifier-free part of ϕ as usual. It follows that, if a class of unary structures can be expressed in MSO(f) then it is in ALIN.

For the converse, it is sufficient to use the encoding of NRAM programs into formulas shown in [GO98]. Let P be a program for an ARAM which makes at most k alternations which accepts a class $\mathcal{C}$ of unary S-structures in linear alternating time for some functional symbol set $S = \{f_1, \ldots, f_l\}$. A run in an ARAM can be divided into segments such that in each segment only one kind

of nondeterministic command, either $\exists guess\ (R_i)$ or $\forall guess\ (R_i)$, is executed. Each segment corresponds to a run of the machine whose initial configuration is the last configuration of the previous segment. In [GO98] it is shown a formula $\alpha(\overline{f}, \overline{F})$ which says that $\overline{F}$ codes a run of linear size of a NRAM which starts with the functions $\overline{f} = f_1, \ldots, f_l\}$ stored on their registers as input. This formula can be easily modified to obtain a new formula $\alpha_l(\overline{f}, \overline{F_1}, \ldots, \overline{F_k})$, for each l, which says that each $\overline{F}_i$ represents a segment of a run where the *guess* operations are of the same type, that is, either $\forall guess$ or $\exists guess$, and such that the relations $\overline{F}_1, \ldots, \overline{F}_k$ together code a run of linear size starting with the functions $\overline{f}$ as input. A formula $\beta(\overline{F_1}, \ldots, \overline{F_k})$ just check whether the instruction *accept* was reached at some point. Without loss of generality, suppose that the first *guess* instruction is of the type $\exists guess$. Then the formula

$$\exists \overline{F}_1 \forall \overline{F}_2 \ldots Q \overline{F}_l [\alpha(\overline{f}, \overline{F}_1, \ldots, \overline{F}_l) \wedge \beta(\overline{F_1}, \ldots, \overline{F_k})]$$

express the class $\mathcal{C}$, where $Q = \forall$ if k is even and $Q = \exists$ otherwise. ∎

In the next section we will introduce the *Bounded-Degree Second-Order Logic*.

4.3 Bounded-Degree Second-Order Logic

In this section, we will define a restriction of SO in which we are allowed to quantify over relations of bounded degree.

We present below three different notions of degree of a relation. First, let us define the *Gaifman graph* of a relation.

Definition 4.6 (Gaifman Graph) *Let $R \in A^r$ be an r-ary relation on A. The Gaifman graph of R is the graph $G = (A, E)$ such that*

$$E = \{(a, a') \in A | \textit{there is } (a_1, \ldots, a_r) \in A^r, a = a_i, a' = a_j, 1 \leq i \neq j \leq r\}.$$

The first notion of degree of a relation comes from the Gaifman graph of the relation.

Definition 4.7 (Gaifman Degree) *The* Gaifman degree $d_{\mathcal{G}}(R)$ *of a relation $R \in A^r$ is the maximum degree of a vertex in the Gaifman graph of R.*

Another way to measure the degree of a relation can be defined by counting the number of tuples in which some element occurs in a relation. Let $R \subseteq A^r$ be a relation and $a \in A$. We define $d_i^R(a) = |\{(a_1, \ldots, a_r) \in A^r | a_i = a\}|$, $1 \leq i \leq r$, the number of tuples in R which have a in the i-th position, and $d^R(a) = |\{(a_1, \ldots, a_r) \in A^r | a = a_i, \text{ for some } 1 \leq i \leq r\}|$, the number of tuples R which have a in some tuple.

Definition 4.8 (Simple and Projection Degree) *The* simple degree $d(R)$ *of a relation $R \in A^r$ is the greatest number of tuples which have a common component, that is, $d(R) = \max\{d^R(a) | a \in A\}$ and the* projection degree $d_p(R)$ *of a relation is the greatest number of tuples which have a common component which appears in the same position, that is, $d(R) = \max\{d_i^R(a) | a \in A, 1 \leq i \leq r\}$.*

We will introduce a form of second-order quantification which is obtained by restricting the range of quantifiers to relations of bounded degree. As we saw above, we have three notions of degree of a relation and hence we can define bounded-degree quantification in three different ways. We will see later that these three notions are actually equivalent.

Definition 4.9 *The* Bounded-Degree Second-Order Logic *(*BDSO*) is the extension of first-order logic with second-order quantifiers $\exists^{\mathcal{G},k}$, $\exists^{s,k}$ and $\exists^{p,k}$ for each $k \in N$ which quantify over relations of Gaifman, simple and projection degree less than or equal to k respectively. If $\phi(X)$ is a formula of* BDSO *where X is a relation variable of arity r and $\mathfrak{I} = (\mathfrak{A}, \beta)$ an interpretation, then we define $\mathfrak{I} \models \exists^{\mathcal{G},k} X\phi(X)$ (respectively, $\mathfrak{I} \models \exists^{s,k} X\phi(X)$ and $\mathfrak{I} \models \exists^{p,k} X\phi(X)$) iff exists a relation $\mathbf{X} \subseteq A^r$ of Gaifman (respectively, simple and projection) degree less than or equal to k, such that $\mathfrak{I}\frac{X}{\mathbf{X}} \models \phi(X)$.*

The three notions of degree given above lead to three definitions of bounded-degree relation. These notions are actually close related, as we show in the lemma below[2].

Lemma 4.2 *For all $c, r \in N$ and $R \in A^r$ exists c' such that:*

(i) if $d_{\mathcal{G}}(R) \leq c$ then $d(R) \leq c'$;

(ii) if $d(R) \leq c$ then $d_p(R) \leq c'$; and

[2]In [Lib04, page 55] it is stated the equivalence between the notions of Gaifman degree and projection degree. In Lemma 4.2 we give a proof of the equivalence of the three notions.

(iii) if $d_p(R) \leq c$ *then* $d_{\mathcal{G}}(R) \leq c'$.

Proof. *(i)* If $d_{\mathcal{G}}(R) \leq c$, then each element $a \in A$ occurs in tuples of R related with at most $c-1$ other elements. An upper bound for $d(R)$ is the number of r-tuples we can construct with c elements. We have

$$d(R) \leq c^r.$$

(ii) Since

$$\{(a_1, \ldots, a_n) \in A^n | a_j = a\} \subseteq \{(a_1, \ldots, a_n) \in A^n | a = a_i, \text{ for some } 1 \leq i \leq n\},$$

for each $1 \leq j \leq n$, then $d_p(R) \leq c$ if $d(R) \leq c$.

(iii) Let $d_p(R) \leq c$. An element $a \in A$ appear in at most $c \cdot r$ tuples. In each tuple in R, a appears related with at most $r-1$ other elements. We have that

$$d_{\mathcal{G}}(R) \leq (r-1) \cdot r \cdot c + 1.$$

∎

We can express in first-order logic the property of a relation to have Gaifman, simple and projection degree less than or equal to certain constant.

Lemma 4.3 *Let* $c \in N$ *and* R *a relation symbol of arity* r. *There are first-order sentences* $\phi_c(R), \psi_c(R)$ *and* $\theta_c(R)$ *such that, for any* $\tau \cup \{R\}$*-structure* $\mathfrak{A}$, $\mathfrak{A} \models \phi_c(R)$, *resp.* $\psi_c(R)$, $\theta_c(R)$, *iff* $d_{\mathcal{G}}(R^{\mathfrak{A}}) \leq c$, *resp.* $d(R^{\mathfrak{A}}) \leq c$, $d_p(R^{\mathfrak{A}}) \leq c$.

Proof. Let $\alpha(x, y, R)$ be the following formula:

$$\alpha(x, y, R) = \exists x_1 \ldots \exists x_r \left[\left(\bigvee_{1 \leq i \neq j \leq r} x = x_i \wedge y = x_j \right) \wedge Rx_1 \ldots x_r \right].$$

This formula says that the elements represented by x and y occur related in some tuple of R. The formula

$$\gamma_c(x, R) = \exists x_1 \ldots \exists x_c \left(\bigwedge_{1 \leq i < j \leq c} (x_i \neq x_j) \wedge \bigwedge_{1 \leq i \leq c} \alpha(x, x_i, R) \right)$$

says that at least c elements are related to the element represented by x. We

can write the desired formula $\phi_c(R)$ as

$$\phi_c(R) = \forall x \neg \gamma_{c+1}(x).$$

Now, for each pair of variable sequences $\overline{x} = x_1, \ldots, x_r$ and $\overline{y} = y_1, \ldots, y_r$ of length r, we define the formula

$$\overline{x} \neq \overline{y} = \bigvee_{1 \leq i \leq r} x_i \neq y_i.$$

Let $\exists \overline{x}$ be an abbreviation of $\exists x_1 \ldots \exists x_r$. Let $\overline{x_i} = x_{i1}, \ldots, x_{ir}$, $1 \leq i \leq c$. We define the formulas

$$\beta_i(x, \overline{x}) = R\overline{x} \wedge x = x_i,$$

$$\psi_{\geq c}(R, x) = \exists \overline{x_1} \ldots \exists \overline{x_c} \left(\bigwedge_{1 \leq i < j \leq c} (\overline{x_i} \neq \overline{x_j}) \wedge \bigvee_{1 \leq i \leq r; 1 \leq j \leq c} (\beta_i(x, \overline{x_j})) \right)$$

and

$$\theta^i_{\geq c}(R, x) = \exists \overline{x_1} \ldots \exists \overline{x_c} \left(\bigwedge_{1 \leq i < j \leq c} \left[\overline{x_i} \neq \overline{x_j} \wedge \bigvee_{1 \leq j \leq c} (\beta_i(x, \overline{x_j})) \right] \right).$$

The formula $\beta_i(x, \overline{x})$ says that the tuple of elements represented by $\overline{x}$ is in the relation R and that the element represented by x occurs on the position i of this tuple. The formula $\psi_{\geq c}(R, x)$ (respectively, $\theta^i_{\geq c}(R, x)$) says that there are at least c tuples of elements in R in which the element represented by x occurs (respectively, on the position i). Finally we have

$$\psi_c(R) = \forall x \neg \psi_{\geq c+1}(R, x)$$

and

$$\theta_c(R) = \forall x (\bigvee_{1 \leq i \leq r} \neg \theta^i_{\geq c+1}(R, x)).$$

∎

It follows from Lemmas 4.2 and 4.3 that the three quantifiers $\exists^{\mathcal{G},c}$, $\exists^{s,c}$ and $\exists^{p,c}$ are equivalent.

Lemma 4.4 *Any formula of* BDSO *can be written using either* $\exists^{\mathcal{G},c}$, $\exists^{s,c}$ *or* $\exists^{p,c}$ *as the only kind of second-order quantifier.*

Proof. It is sufficient to relativize each quantifier to the formulas of Lemma 4.3 using the bounds of Lemma 4.2. Namely we can apply the following equivalences:

$$\begin{aligned}
\exists^{\mathcal{G},c} R\alpha(R) &\equiv \exists^{s,c^{r-1}} R(\phi_c \wedge \alpha(R)),\\
\exists^{s,c} R\alpha(R) &\equiv \exists^{p,c} R(\psi_c \wedge \alpha(R)),\\
\exists^{p,c} R\alpha(R) &\equiv \exists^{\mathcal{G},r\cdot(r-1)\cdot c} R(\psi_c \wedge \alpha(R)),
\end{aligned}$$

where r is the arity of the relation symbol R. ∎

Each quantifier introduced above gives rise to an alternating quantifier hierarchy.

Definition 4.10 *We define the classes of formulas $\Pi^{\bullet}_{i,k}$ and $\Sigma^{\bullet}_{i,k}$, for $\bullet \in \{d_{\mathcal{G}}, s, p\}$ and $i, k \in N$, recursively as:*

$$\begin{aligned}
\Pi^{\bullet}_{0,k} &= \Sigma^{\bullet}_{0,k} = FO;\\
\Pi^{\bullet}_{i+1,k} &= \forall^{\bullet,k'} R_1 \ldots \forall^{\bullet,k'} R_n \phi, \text{ where } \phi \in \Sigma^{\bullet}_{i,k} \text{ and } k' \leq k;\\
\Sigma^{\bullet}_{i+1,k} &= \exists^{\bullet,k} R_1 \ldots \exists^{\bullet,k} R_n \phi, \text{ where } \phi \in \Pi^{\bullet}_{i,k} \text{ and } k' \leq k.
\end{aligned}$$

The index i represent the number of alternating quantifier blocks and k the maximum degree of a quantified relation.

Given a fixed i, we define the degree hierarchies $\Pi^{\bullet}_i = \bigcup_{k\in N} \Pi^{\bullet}_{i,k}$ and $\Sigma^{\bullet}_i = \bigcup_{k\in N} \Sigma^{\bullet}_{i,k}$.

We can show that the degree hierarchies collapse at some point for all the three notions of degree introduced below.

Theorem 4.3 *The degree hierarchies $\Pi^{d_{\mathcal{G}}}_i$ and $\Sigma^{d_{\mathcal{G}}}_i$, Π^{p}_i and Σ^{p}_i, Π^{s}_i and Σ^{s}_i, collapse at some level regardless of i.*

Proof. We first show the result for Π^{s}_i and Σ^{s}_i. The strategy is to split a relation into relations of smaller degree. Let R be a relation of arity r and simple degree $d(R) = c$. Then we can split R into $r \cdot (c-1) + 1$ relations of degree 1. Intuitively, we can distribute the tuples into the relations so that each element occurs in only one tuple in a relation. Since the degree of R is c, each element appears in at most $c-1$ other tuples. Given a r-tuple $\overline{a} = (a_1, \ldots, a_r)$ there are at most $r \cdot (c-1)$ tuples which have some element which also occurs in $\overline{a}$. It follows that there are at most $r \cdot (c-1)$ relations of degree 1 which

contains a tuple where some of the elements in the tuple $\overline{a}$ occurs. Since we have $r \cdot (c-1)+1$ relations, there is a relation without any tuple where some of the elements $a_1, \ldots, a_r$ occurs. We can put $\overline{a}$ in this relation. It means that each relation of simple degree c is the union of $c \cdot (c-1)+1$ relations of degree at most 1.

On the other hand, any union of relations of degree 1 gives rise to a relation of degree less than or equal to c, provided that each element occurs in a tuple in at most c different relations. This condition can be expressed by the following formula:

$$SD_c(R_0, \ldots, R_{r\cdot(c-1)}) = \psi_c(\bigvee_{l=0}^{r\cdot(c-1)} R_l),$$

where $\psi_c(R)$ is the formula from Lemma 4.3 and $\psi_c(\bigvee_{l=0}^{r\cdot(c-1)} R_l)$ is obtained from $\psi_c(R)$ by substituting $Rt_1 \ldots t_r$ with $\bigvee_{l=0}^{r\cdot(c-1)} R_l t_1 \ldots t_r$ for any terms $t_1 \ldots t_r$. Based on this, we can write the following equivalence:

$$\exists^{s,c} R\phi(R) \equiv \exists^{s,1} R_0 \ldots \exists^{s,1} R_{r\cdot(c-1)}(SD_c(R_0, \ldots, R_{r\cdot(c-1)}) \wedge \phi(\bigvee_{j=0}^{r\cdot(c-1)} R_j)),$$

where $\phi(\bigvee_{j=0}^{r\cdot(c-1)} R_i)$ is obtained from $\phi(R)$ by replacing each atom of the form $Rt_1 \ldots t_r$ with

$$\bigvee_{j=0}^{r\cdot(c-1)} R_j t_1 \ldots t_r.$$

This shows how to substitute every quantifier $\exists^{s,c}$ with blocks of quantifiers $\exists^{s,1}$. Hence Π_i^s and Σ_i^s collapse to level 1. Applying Lemma 4.3 we get the collapsing results for the hierarchies $\Pi_i^{d_\mathcal{G}}$, $\Sigma_i^{d_\mathcal{G}}$, Π_i^s and Σ_i^s. ∎

In the following section we give some examples of queries which can be expressed in this logic.

4.3.1 Examples

It follows from Definition 4.9 that monadic second-order logic is included in BDSO. That is because unary relations have Gaifman degree 0. We present below some examples of queries expressible in BDSO.

Example 4.5 (Reachability) Let $\tau = \{E, s, t\}$ be a symbol set where E is a binary relation and s and t are constant symbols. τ-structures are graphs

equipped with two distinct constant symbols. The reachability query is the class of finite τ-structures in which there is a path between s and t.

Let S be a binary relation symbol. We define the field of a relation as the set of elements which appear in some tuple of the relation. The formula

$$field(S,x) = \exists y(Sxy \vee Syx)$$

defines the field of a binary relation S. Let $\phi(S)$ be a first-order formula saying that S is a "successor-like" relation, that is, each element has at most one predecessor and at most one successor and there is only one element without predecessor and only one element without successor:

$$\begin{aligned}\phi(S) \quad = \quad & \forall x(field(S,x) \rightarrow \exists^{\leq 1} y Sxy \wedge \exists^{\leq 1} y Syx) \wedge \\ & \exists^{=1} y(field(S,y) \wedge \neg\exists z Syz) \wedge \exists^{=1} y(field(S,y) \wedge \neg\exists z Szy)).\end{aligned}$$

Any interpretation of S which satisfies $\phi(S)$ is a relation of degree at most 2. Besides this, any interpretation of S can be described as a directed path together with some (or possibly none) directed cycles. Let $\psi(S)$ be a first-order formula which says that, if the pair (a,b) is in S, then it is in E and that s has no predecessor with respect to S and that t has no successor with respect to S:

$$\psi(S) = \forall x \forall y(Sxy \rightarrow Exy) \wedge \neg\exists x Sxs \wedge \neg\exists x Stx.$$

In any interpretation of S which satisfies $\phi \wedge \psi$ there is a path between s and t in the graph. Consider the following formula:

$$\theta = \exists^{\mathcal{G},2} S(\phi(S) \wedge \psi(S)).$$

A τ-structure $\mathfrak{A}$ satisfies θ iff there is a path from $s^{\mathfrak{A}}$ to $t^{\mathfrak{A}}$ in $\mathfrak{A}$.

Example 4.6 (Graph Isomorphism) Let $\tau = \{V, V', E, E'\}$ be a symbol set where V and V' are unary relations and E and E' binary relations. The *graph isomorphism* query is the set of τ-structures such that the $\{V, E\}$ and $\{V', E'\}$ reducts are isomorphic graphs. Let F be a binary relation symbol, $Img(F,x)$ a first-order formula which says that x belongs to the image of F, namely

$$Img(F,x) = \exists y F(y,x),$$

$Dom(F, x)$ a first-order formula which says that x belongs to the domain of F, namely

$$Dom(F, x) = \exists y F(x, y),$$

and $iso(F)$ be a first-order logic sentence which says that F is an isomorphism between the $\{V, E\}$ and $\{V', E'\}$ reducts, namely:

$$iso(F) = biject(F) \wedge \forall x(Img(F, x) \leftrightarrow Vx \wedge Dom(F, x) \leftrightarrow V'x) \wedge$$

$$\forall x \forall y(Vx \wedge Vy \wedge Exy \leftrightarrow \exists x' \exists y'(Fxx' \wedge Fyy' \wedge E'x'y')),$$

where $biject(F)$ is a first-order formula which says that F is the graph of a bijection. The graph isomorphism query is expressed by the following $\exists$BDSO sentence:

$$graphIso = \exists^{\mathcal{G},2} F \; iso(F).$$

In the next section, we will talk about the equivalence between bounded-degree quantification and quantification of functions and relations of linear size.

4.4 Functions and Linear Size Relations

In this Section, we give some characterizations of BDSO using other kinds of quantification. In [DLS98], Schwentick et al. studied several restrictions of second-order quantification over binary relations, as, for example, quantifying linear-order relations, unary functions, partial order relation, relations of bounded degree, relations of bounded out-degree, etc.. They studied the existential fragments of the logics obtained by using these quantifiers and divided them in the following four groups:

1. partial order relations, arbitrary binary relations;
2. unary functions, linear order relations, equivalence relations, addition, graphs with bounded outdegree, graphs with linearly many arcs;
3. permutations, successor relations, graphs with bounded degree;
4. sets.

Schwentick et al. [DLS98] showed that any existential formula obtained by quantifying relations of any kind above is equivalent to an existential formulas

using quantification of relation of any other sort in the same group. For example, each formula which can be written using a prefix of existential quantifiers ranging over linear order relations has an equivalent using a prefix of existential quantifiers ranging over graphs with bounded outdegree. Beside this, each group strictly contains the group below.

Here we will consider the entire logic, where both the existential and universal quantifiers may be used, for the quantification of unary functions and bounded-degree relations (not necessarily binary relations). First, we will show how we can quantify over linear orders using bounded-degree quantification.

Although we cannot directly quantify over relations of unbounded degree, sometimes we can define such relations from relations of bounded degree. For instance, we can existentially quantify a relation which is a successor relation on the domain and then use a formula to define a linear order from this successor relation.

Definition 4.11 *Given a binary relation R and a formula $\phi(R)$ we can construct a new formula $\exists^{lo}R\phi(R)$. We define the satisfaction relation between an interpretation $\mathfrak{I}$ and a formula of the form $\exists^{lo}R\phi(R)$ as*

$$\mathfrak{I} \models \exists^{lo}R\phi(R)$$

iff

exists a linear order $<$ on the domain of $\mathfrak{I}$ such that $\mathfrak{I}^{R}_{<} \models \phi(R)$.

As we said, group 2 is strictly contained in group 3 above with respect to existential formulas. It is noted in [DLS98] that on ordered structures these two groups coincide. It means that, if we allow both existential and universal quantification, these two groups are equivalent, since we can quantify order relations using existential and universal quantification of bounded-degree relations. We give a proof of this fact below.

Lemma 4.5 *Every formula using the quantifier $\exists^{lo}$ is equivalent to a formula using only bounded-degree quantification.*

Proof. Let $\phi(S)$ be the formula from Example 4.3.1 which says that S is a successor-like relation. Let $min(S,x) = \neg\exists ySyx$ be a first-order formula which says that x is the element of S without predecessor and let $cl(S,x,y)$ be the following MSO formula:

$$cl(S,x,y) = \forall C[(Cx \wedge \forall z\forall w(Cz \wedge Szw \rightarrow Cw)) \rightarrow Cy].$$

The formula $cl(S, x, y)$ says that y is reachable from x through S transitions. Let $succ(S)$ be the formula

$$succ(S) = \phi(S) \wedge \exists x(min(S, x) \wedge \forall y\ cl(S, x, y)).$$

This formula says that every element is reachable from x through S transitions, where x represents the element without predecessors. A relation interpreting S satisfies $succ(S)$ iff it is a successor relation. We can then define the transitive closure of S using monadic quantification, namely the formula $cl(S, x, y)$. Given a formula $\alpha(R)$, let $\alpha(cl(R))$ be obtained from $\alpha(R)$ by replacing atoms of the form Rtt' with $cl(S, t, t')$. It is clear that

$$\mathfrak{I} \models \exists^{\mathcal{G},2} S(\phi(S) \wedge succ(S) \wedge \alpha(cl(S))$$

iff

exists a linear order $\mathbf{R}$ on the domain of $\mathfrak{I}$ such that $\mathfrak{I}^{\mathbf{R}}_{R} \models \alpha(R)$.

Hence, we can define the quantifier $\exists^{lo}$ which ranges over linear orders using bounded-degree quantification as

$$\exists^{lo} R\ \alpha(R) = \exists^{\mathcal{G},2} S(\phi(S) \wedge succ(S) \wedge \alpha(cl(S)).$$

∎

We are able to show now that quantification over unary functions has the same expressive power as bounded-degree quantification. It will follow from the equivalence of existential quantification of linear order relations and unary functions [DLS98]. We will give a proof of how we can define the $\exists^{tf}$ quantifier, which quantifies over unary functions using bounded-degree quantification.

Lemma 4.6 *Every formula in* MSO(f) *has an equivalent in* BDSO.

Proof. Given a linear order $<$ on A and an injective partial function $G \subseteq A^2$ such that the least element of A w.r.t. $<$ is in the domain $dom(G)$ of G, we can define a total function $h_{<,G} : A \to A$ as $h_{<,G}(a) = g(a_G)$, where a_G is the greatest element in $dom(G)$ less than or equal to a w.r.t. $<$. Conversely, let $f : A \to A$ be a function with image $img(h)$. It is not difficult to see that there is an ordering $<$ of A such that, for all $a, b \in A$, if $h(a) = h(b)$, then for all $c \in A$ such that $a \leq c \leq b$ we have $h(c) = h(a)$. The order $<$ is composed by segments whose elements are those with the same image w.r.t. h. We can

then define a partial function $G \in A^2$ such that $a \in dom(G)$ iff it is the least element in a segment and $G(a) = h(a)$ for all $a \in dom(G)$. It means that we are able to quantify over functions using quantification over linear orders and injective partial functions. First note that any injective partial function has Gaifman degree at most 2. Let $<$ and G be binary relations and $inject(G)$ a first-order sentence saying that G is (the graph of) an injective partial function whose domain contains the least element in the domain of a structure w.r.t. $<$:

$$\begin{aligned} inject(G) \quad = \quad & \forall x(field(G,x) \rightarrow \exists^{=1} y G x y) \wedge \\ & \forall x \forall y \forall z(Gxz \wedge Gyz \rightarrow x = y) \wedge \\ & \exists x(\forall y(x \leq y) \wedge \exists y G(x,y)). \end{aligned}$$

As we explained above, given a linear order $<$ and a partial injective function G we can define the function $h_{<,G}$ as:

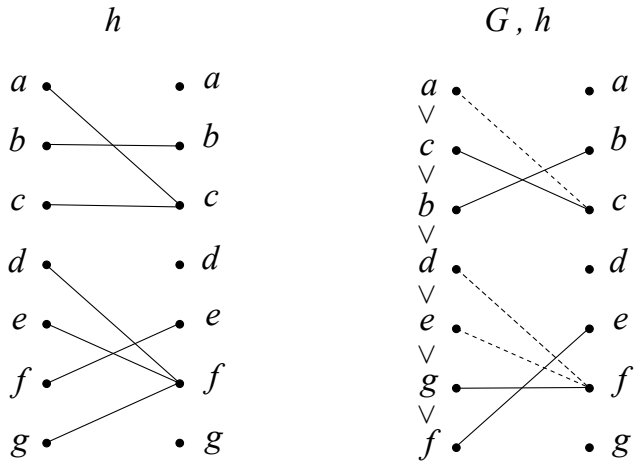
$$func(<, G, x, y) = \exists z(G(z,y) \wedge \forall z'(Dom(G,z') \wedge z < z' \rightarrow x < z')).$$

In this way, we can define the quantifier $\exists^{tf} F$ as:

$$\exists^{tf} F \alpha(F) = \exists^{lo} < \exists^{\mathcal{G},2} G(inject(G) \wedge \alpha(func(<, G))),$$

where $\alpha(func(<, G))$ is obtained from $\alpha(F)$ by replacing every atomic formula of the form Ftt' with $func(<, G, t, t')$. ∎

The picture below shows the strategy used in the proof above. Let $A = \{a, b, c, d, e, f\}$ and $h = \{(a,c), (b,b), (c,c), (d,f), (e,f), (f,e), (g,f)\}$. We order A so that the elements that h maps to the same value form a segment in the order. A possible one is $a > c > b > d > e > g > f$. The segments are $a > c$, b, $d > e > g$ and f. Now we just have to consider the values of h for the least elements of each segment. For this ordering $>$ we have $G = \{(c,c), (b,b), (g,f), (f,e)\}$. If we want to find the value $h(x)$ of some element $x \in A$, we just have to find the segment where it lies, which means to find the greatest element y less than or equal to x in $dom(G)$. Then we have $h(x) = G(y)$. For example, if $x = e$, then $y = g$ and $h(x) = G(y) = f$.



On the other hand, we can simulate quantification of relations of bounded degree using quantification of unary functions. Although the results in [DLS98] are restricted to binary relations, we can show that the same hold for relations of bounded degree of arbitrary arity.

Lemma 4.7 *Each formula of* BDSO *has an equivalent in* MSO(f).

Proof. We saw in the Section 4.2 above how to define the quantifier $\exists^{pf}$ in MSO(f). Let $\mathbf{R} \in A^r$ be an r-ary relation of simple degree 1. Let $F_i = \{(a_i, a_{i+1}) \in A^2 | (a_1, \ldots, a_r) \in \mathbf{R}\}$, $1 \leq i \leq r-1$. Since each element in A occurs at most in one tuple in $\mathbf{R}$, each F_i is the graph of a partial injective function and $\mathbf{R} = \{(a_1, \ldots, a_r) \in A^r | (a_i, a_{i+1}) \in F_i, 1 \leq i \leq r-1\}$. We call these relations a functional representation of R.

Let $F_1, \ldots, F_{r-1}$, be a binary relation symbols and $\alpha(F_1, \ldots, F_{r-1}, x_1, \ldots, x_r)$ be the following formula:

$$\alpha(F_1, \ldots, F_{r-1}, x_1, \ldots, x_r) = (F_{i,1}x_1x_2 \wedge \cdots \wedge F_{i,r-1}x_{r-1}x_r).$$

It is not difficult to see that a functional representation of $\mathbf{R}$ satisfies this formula for an assignment $a_1, \ldots, a_r$ to the variables $x_1, \ldots, x_r$ iff $(a_1, \ldots, a_r) \in \mathbf{R}$.

Let $\mathbf{R} \in A^r$ be a relation of simple degree c. By Theorem 4.3, $\mathbf{R}$ is the union of $m = r \cdot (c-1) + 1$ relations of simple degree 1, say $\mathbf{R_1}, \ldots, \mathbf{R_m}$. We call these relations a decomposition of $\mathbf{R}$ (of simple degree 1).

We need a formula which express the fact that a set of relations is a functional representation of a relation of simple degree 1. The following formula express this fact:

$$Dec(F_1, \ldots, F_{r-1}) = \forall x \exists^{\geq 1} \overline{x} (\bigvee_{i=1}^{r-1} x = x_i \wedge \bigwedge_{i=1}^{r-1} F_i x_i x_{i+1}).$$

Let $F_{1,1}, \ldots, F_{m,r-1}$ be binary relations. Each sequence $F_{i,1}, \ldots, F_{i,r-1}$ is a functional representation of R_i. That is, we first divide a relation into several relations of degree 1 and then we split each relation of degree 1 into a functional decomposition. We have the following equivalence:

$$\begin{aligned} \exists^{s,c} R\phi(R) \quad \equiv \quad & \exists^{pf} F_{1,1} \ldots \exists^{pf} F_{m,r-1} \left[\bigwedge_{i=1}^{m} Dec(F_{i,1}, \ldots, F_{i,r-1}) \wedge \right. \\ & \psi_c \left(\bigvee_{i=1}^{m} \alpha(F_{i,1}, \ldots, F_{i,r-1}, x_1, \ldots, x_r) \right) \wedge \\ & \left. \phi \left(\bigvee_{i=1}^{m} (\alpha(F_{i,1}, \ldots, F_{i,r-1}, x_1, \ldots, x_r)) \right) \right], \end{aligned}$$

where $\psi_c(R)$ is the formula from Lemma 4.3 which says that R has Gaifman degree at most c,

$$\psi_c(\bigvee_{i=1}^{m} \alpha(F_{i,1}, \ldots, F_{i,r-1}, x_1, \ldots, x_r))$$

is obtained from $\psi_c(R)$ by replacing $Rt_1 \ldots t_r$ with

$$\bigvee_{i=1}^{m} \alpha(F_{i,1}, \ldots, F_{i,r-1}, t_1, \ldots, t_r)$$

for all terms $t_1 \ldots t_r$, and

$$\phi(\bigvee_{i=1}^{m} (\alpha(F_{i,1}, \ldots, F_{i,r-1}, x_1, \ldots, x_r)))$$

is obtained from $\phi(R)$ by replacing each atom of the form $Rt_1 \ldots t_r$ with

$$\bigvee_{i=1}^{m} (\alpha(F_{i,1}, \ldots, F_{i,r-1}, t_1, \ldots, t_r)).$$

∎

It follows immediately from Lemmas 4.6 and 4.7 that, besides the degree hierarchy, the arity hierarchy also collapses.

Corollary 4.1 *Every formula of* BDSO *can be written using bounded-degree quantification over binary or monadic relations.*

Proof. We just need to apply Lemma 4.7 to get a formula in MSO(f) and then Lemma 4.6 to get new a formula in BDSO. Note that in the formula obtained from Lemma 4.7 only binary relations of bounded degree and monadic relations appear quantified. ∎

Now we will show that bounded-degree quantification is also equivalent to quantifying over relations of linear size. [DLS98] prove the fact for binary relation. We present a different proof for relations of arbitrary arity.

Definition 4.12 *Let $c \in N$ and R an r-ary relation. We define the quantifier $\exists^c$ which quantifies over relations of size $c \cdot n$, where n is the size of the structure where the formula is being interpreted. Let $\mathfrak{I}$ be an interpretation with domain A. We define the satisfaction relation for formulas of the form $\exists^c R\phi(R)$ as:*

$$\mathfrak{I} \models \exists^c R\phi(R)$$

iff

$$\text{exists a relation } \mathbf{R},\ |\mathbf{R}| < c \cdot |A|,\ \text{such that } \mathfrak{I}^{\mathbf{R}}_{R} \models \phi(R). \tag{4.2}$$

Lemma 4.8 *Each formula in* BDSO *can be written using the $\exists^c$ quantifier instead of bounded-degree quantification.*

Proof. Any relation of bounded degree has linear size. In order to see this, let $\mathbf{R} \subseteq A^r$ be an r-ary relation of simple degree at most d. Then each element from A occurs in at most d tuples in $\mathbf{R}$, which means that there are at most $d \cdot n$ tuples in $\mathbf{R}$. The same for Gaifman and projection degrees, but with different upper bounds. In order to quantify over bounded-degree relations using linear size quantification, one need only to relativize the linear size quantification to relations of bounded-degree. Let $\exists^{s,d} R\alpha(R)$ be a BDSO formula. We have the following equivalence:

$$\exists^{s,d} R\alpha(R) \equiv \exists^d R(\psi_c(R) \wedge \alpha(R)),$$

where $\psi_c(R)$ is the formula from Lemma 4.3. By applying this equivalence we can iteratively eliminate all occurrences of bounded-degree quantification. ∎

Lemma 4.9 *Each formula written with quantification of linear size relations can be rewritten using bounded-degree quantification.*

Proof. Let $\mathbf{R} \subseteq A^r$ be an r-ary relation such that $|\mathbf{R}| \leq c \cdot |A|$. Then $\mathbf{R}$ is the union of c pairwise disjoint relations $\mathbf{R}_1, \ldots, \mathbf{R}_c$ of size at most $|A|$ each. Let $f^i : A \to A^r$ be a surjective partial function. Let $F_1^i, \ldots, F_r^i \subseteq A^2$, $1 \leq i \leq c$, be defined as:

$$F_j^i = \{(a, a_i) \in A^2 | f^i(a) = (a_1, \ldots, a_r)\}.$$

It follows that $(a_1, \ldots, a_r) \in \mathbf{R}$ iff exists an $a \in A$ and $1 \leq i \leq c$ such that $F_j^i a a_j$, $1 \leq j \leq r$. Let $\exists^c R \phi(R)$ be a formula. We have the following equivalence:

$$\exists^c R \phi(R) \equiv \exists^{pf} F_1^1 \ldots \exists^{pf} F_r^c \phi(Seg(F_1^1, \ldots, F_r^c)),$$

where $\phi(Seg(F_1^1, \ldots, F_r^c))$ is obtained from $\phi(R)$ by substituting each atom of the form $Rt_1 \ldots t_r$ with

$$Seg(F_1^1, \ldots, F_r^c, t_1, \ldots, t_r) = \exists x \bigvee_{i=1}^{c} \bigwedge_{j=1}^{r} F_j^i x t_j.$$

∎

The equivalence between BDSO and MSO(f) give us a characterization of the expressive power of of BDSO in terms of the complexity classes it can express over unary structures (see Theorem 4.2).

Theorem 4.4 BDSO *captures ALIN on classes of unary structures.*

In the next section, we will add the transitive closure operator on high-order relations on relations of bounded degree. We will show that it captures linear number of registers in a NRAM provided that the values stored in the registers are linear in the input size.

4.5 Transitive Closure

In this section we will augment the BDSO logic with the ability of defining the transitive closure of higher-order relations on relations of bounded degree. To

be able to define the transitive closure of higher-order relations gives to second-order logic expressive power enough to express queries complete for PSPACE. Actually, the logic SO(TC) captures the class PSPACE of queries computable in polynomial space [Imm99]. We will show that BDSO augmented with transitive closure on high-order relations on relations of bounded degree capture linear number of registers on an NRAM provided that the values stored in the registers during the computation are linear.

Definition 4.13 (BDSO(TC)**,** BDSO(pos TC)**)** *Let* $\phi(R_1, \ldots, R_k, R'_1, \ldots, R'_k)$ *be an* S*-formula of* BDSO *with free relational variables* $R_1, \ldots, R_k, R'_1, \ldots, R'_k$ *such that* R_i *and* R'_i *have the same arity. The logic* BDSO(TC) *extends* BDSO *with the transitive closure operator*

$$[TC^d_{R_1,\ldots,R_k,R'_1,\ldots,R'_k}\phi(R_1, \ldots, R_k, R'_1, \ldots, R'_k)](\overline{X}, \overline{X'})$$

which defines the transitive closure of the binary relation on tuples of relations of bounded degree defined by $\phi(R_1, \ldots, R_k, R'_1, \ldots, R'_k)$ *when the relational free variables are interpreted as relations of degree less than or equal to* d*. Given an* S*-structure* $\mathfrak{A}$*, we define a relation*

$$\phi^{\mathfrak{A},d} \subseteq \left(A^{r_1} \times \cdots \times A^{r_k}\right)^2,$$

where r_i *is the arity of* R_i*,* $0 \leq i \leq k$*, as*

$$\begin{aligned}\phi^{\mathfrak{A},d} \quad = \quad & \{((\mathbf{R}_1, \ldots, \mathbf{R}_k), (\mathbf{R}'_1, \ldots, \mathbf{R}'_k)) \subseteq \left(A^{r_1} \times \cdots \times A^{r_k}\right)^2 \mid \\ & (\mathfrak{A}, \mathbf{R}_1, \ldots, \mathbf{R}_k, \mathbf{R}'_1, \ldots, \mathbf{R}'_k) \models \phi(R_1, \ldots, R_k, R'_1, \ldots, R'_k) \\ & \textit{and each } \mathbf{R_i} \textit{ and } \mathbf{R'_i} \textit{ has Gaifman degree less than or equal to } d\}.\end{aligned}$$

We define the satisfiability relation for transitive closure formulas as:

$$(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X}'}) \models [TC^d_{R_1,\ldots,R_k,R'_1,\ldots,R'_k}\phi(R_1, \ldots, R_k, R'_1, \ldots, R'_k)](\overline{X}, \overline{X'})$$

iff

$$\overline{\mathbf{X}}, \overline{\mathbf{X}'} \in cl(\phi^{\mathfrak{A}}),$$

where $cl(\Phi^{\mathfrak{A}})$ *is the transitive closure of* $\phi^{\mathfrak{A}}$*.*

BDSO(pos TC) *is the fragment of* BDSO(TC) *where the* TC^d *operator does not occur in the scope of negation.*

In the following, we will show some results about BDSO(TC). They will be used later to show that formulas of BDSO(TC) can be put in a specific form.

The next two lemmas show how to eliminate existential and universal quantification of formulas with the TC^d operator. We use the symbols $\emptyset$ and A to represent the empty relation and the total relation, regardless of the arity.

Lemma 4.10

$$\exists^d Y[TC^{d'}_{\overline{R},\overline{R'}}\phi(\overline{R},\overline{R'},Y)](\overline{X},\overline{X'})$$

$$\equiv$$

$$[TC^{d''}_{P,Y,\overline{R},P',Y',\overline{R'}}\phi'(P,Y,\overline{R},P',Y',\overline{R'},\overline{X},\overline{X'})](\emptyset,\emptyset,\overline{\emptyset},\emptyset,\emptyset,\overline{\emptyset})$$

where $d'' = max\{d, d'\}$ *and*

$$\phi'(P,Y,\overline{R},P',Y',\overline{R'},\overline{X},\overline{X'}) = \begin{cases} \psi_{d'}(Y) \wedge \bigwedge_{X_i \in \overline{X}} \psi_d(X_i) \wedge \\ \psi_{d'}(Y') \wedge \bigwedge_{X'_i \in \overline{X'}} \psi_d(X'_i) \wedge \\ (P = \emptyset \wedge P' = A \wedge Y = \emptyset \wedge \overline{R} = \overline{\emptyset} \wedge \overline{R'} = \overline{X}) \vee \\ (\phi(\overline{R},\overline{R'},Y) \wedge P = A \wedge P' = A \wedge Y = Y') \vee \\ (P = A \wedge P' = \emptyset \wedge Y' = \emptyset \wedge \overline{R} = \overline{X'} \wedge \overline{R'} = \overline{\emptyset}). \end{cases}$$

Proof. Let $\mathfrak{A}$ be a structure and $\overline{\mathbf{X}}$ and $\overline{\mathbf{X}'}$ be tuples of relations which interpret $\overline{X}$ and $\overline{X'}$. We have

$$(\mathfrak{A},\overline{\mathbf{X}},\overline{\mathbf{X}'}) \models [TC^{d''}_{P,Y,\overline{R},P',Y',\overline{R'}}\phi'(P,Y,\overline{R},P',Y',\overline{R'},\overline{X},\overline{X'})](\emptyset,\emptyset,\overline{\emptyset},\emptyset,\emptyset,\overline{\emptyset})$$

iff

$$((\emptyset,\emptyset,\overline{\emptyset}),(\emptyset,\emptyset,\overline{\emptyset})) \in cl(\phi^{(\mathfrak{A},\overline{\mathbf{X}},\overline{\mathbf{X}'}),d''})$$

iff there is a $\mathbf{Y}$ of degree d which interpret Y such that

$$((\emptyset,\emptyset,\overline{\emptyset}),(A,\mathbf{Y},\overline{\mathbf{X}})) \in \phi'^{(\mathfrak{A},\overline{\mathbf{X}},\overline{\mathbf{X}'}),d''}$$

and

$$((A,\mathbf{Y},\overline{\mathbf{X}}),(A,\mathbf{Y},\overline{\mathbf{X}'})) \in cl(\phi'^{(\mathfrak{A},\overline{\mathbf{X}},\overline{\mathbf{X}'}),d''})$$

and

$$((A, \mathbf{Y}, \overline{\mathbf{X}'}), (\emptyset, \emptyset, \overline{\emptyset})) \in \phi'^{(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X}'}), d''}$$

iff there is a $\mathbf{Y}$ of degree d which interpret Y such that

$$(\overline{\mathbf{X}}, \overline{\mathbf{X}'}) \in cl(\phi^{(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X}'}, \mathbf{Y}), d'})$$

iff

$$(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X}'}) \models \exists^d Y [TC^{d'}_{\overline{R}, \overline{R'}} \phi(\overline{R}, \overline{R'}, Y)](\overline{X}, \overline{X'}).$$

∎

For the case of universal quantification, we need the following definition.

Definition 4.14 ($\mathbf{X} \leq^d_k \mathbf{X}'$) *Let A be a set of cardinality m and $<$ an order relation on A. Let $<^k$ be the lexicographic order on k-tuples of elements in A induced by $<$. Given a set $\mathbf{X}$ we define a binary string $s_{\mathbf{X}}$ of size m^k which has a 1 in position i iff the i-th tuple of A^k with respect to $<^k$ belongs to $\mathbf{X}$. Let $n(s)$ be the natural number represented by the string s in binary code. Let $\mathbf{X}$ and $\mathbf{X}'$ be k-ary relations on A. We define a linear order $\leq_k$ between k-ary relations on A as*

$$\mathbf{X} \leq_k \mathbf{X}'$$

iff

$$n(s_{\mathbf{X}}) \leq n(s_{\mathbf{X}'}).$$

We define the relation $\leq^d_k$ as an order relation between k-ary relations of degree at most d as:

$$\mathbf{X} \leq^d_k \mathbf{X}'$$

iff

X and X' have Gaifman degree at most d and $n(s_{\mathbf{X}}) \leq n(s_{\mathbf{X}'})$.

We denote by $succ_k(\mathbf{X}, \mathbf{X}')$ and $succ^d_k(\mathbf{X}, \mathbf{X}')$ the successor relation with respect to $\leq_k$ and $\leq^d_k$, respectively.

Lemma 4.11 *The relations $\leq_k$, $\leq^d_k$, $succ_k$ and $succ^d_k$ can be defined in BDSO.*

Proof. The following formula defines the linear order relations $\leq_k$ and $\leq^d_k$ from Definition 4.14:

$$X \leq_k X' = \exists \overline{x} \forall \overline{y} (\overline{x} < \overline{y} \rightarrow (X\overline{y} \leftrightarrow X'\overline{y}) \wedge \overline{y} \leq \overline{x} \rightarrow (X\overline{x} \rightarrow X'\overline{x})),$$

$$X \leq_k^d X' = \phi_d(X) \wedge \phi_d(X') \wedge X \leq_k X',$$

where $\phi_d(X)$ is the formula from Lemma 4.3 which says that the relation X has Gaifman degree at most d. The relations $succ_k$ and $succ_k^d$ can be defined as:

$$succ_k(X, X') = \forall Y(Y \leq_k X' \wedge Y \neq X' \rightarrow Y \leq_k X),$$

$$succ_k^d(X, X') = \forall Y(Y \leq_k^d X' \wedge Y \neq X' \rightarrow Y \leq_k^d X).$$

∎

Lemma 4.12

$$\forall^d Y[TC^{d'}_{\overline{R},\overline{R'}}\phi(\overline{R}, \overline{R'}, Y)](\overline{X}, \overline{X'})$$

$$\equiv$$

$$[TC^{d''}_{Y,\overline{R},Y',\overline{R'}}\phi'(Y, \overline{R}, Y', \overline{R'}, \overline{X}, \overline{X'})](\emptyset, \overline{X}, A, \overline{X'})$$

where $d'' = max\{d, d'\}$ *and*

$$\phi'(Y, \overline{R}, Y', \overline{R'}, \overline{X}, \overline{X'}) = \begin{cases} \psi_{d'}(Y) \wedge \bigwedge_{X_i \in \overline{X}} \psi_d(X_i) \wedge \\ \psi_{d'}(Y') \wedge \bigwedge_{X'_i \in \overline{X'}} \psi_d(X'_i) \wedge \\ \bigwedge_{X'_i \in \overline{X'}} \psi_d(X'_i) \wedge \\ (\phi(\overline{R}, \overline{R'}, Y) \wedge Y = Y') \vee \\ succ_k^d(Y, Y') \wedge R = X \wedge R' = X'. \end{cases}$$

Proof. Let $\mathfrak{A}$ be a structure and $\overline{\mathbf{X}}$ and $\overline{\mathbf{X'}}$ be tuples of relations which interpret $\overline{X}$ and $\overline{X'}$. We have

$$(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X'}}) \models [TC^{d}_{Y,\overline{R},Y',\overline{R'}}\phi'(Y, \overline{R}, Y', \overline{R'}, \overline{X}, \overline{X'})](\emptyset, \overline{X}, A, \overline{X'})$$

iff

$$((\emptyset, \overline{\mathbf{X}}), (A, \overline{\mathbf{X'}})) \in cl(\phi'^{(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X'}}), d''})$$

iff, for all $\mathbf{Y}$ of degree d which interpret Y,

$$((\mathbf{Y}, \overline{\mathbf{X}}), (\mathbf{Y}, \overline{\mathbf{X'}})) \in cl(\phi'^{(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X'}}), d''})$$

iff, for all $\mathbf{Y}$ which interpret Y,

$$(\overline{\mathbf{X}}, \overline{\mathbf{X'}}) \in cl(\phi^{((\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X'}}, \mathbf{Y}), d'})$$

iff

$$(\mathfrak{A}, \overline{\mathbf{X}}, \overline{\mathbf{X}'}) \models \forall^d Y [TC^{d'}_{\overline{R},\overline{R'}} \phi(\overline{R}, \overline{R'}, Y)](\overline{X}, \overline{X'}).$$

∎

From the lemmas above we have the following.

Theorem 4.5 *Each formula of* BDSO(pos TC) *is equivalent to one where the* TC^d *operator does not appear in the scope of any connective or quantifier.*

Proof. It follows by induction on formulas of BDSO(pos TC). For the cases of conjunction and disjunction, it is sufficient to see that

$$\phi \wedge [TC^d_{\overline{R},\overline{R'}} \psi](\overline{X}, \overline{X'}) \equiv [TC^d_{\overline{R},\overline{R'}} \phi \wedge \psi](\overline{X}, \overline{X'})$$

and

$$\phi \vee [TC^d_{\overline{R},\overline{R'}} \psi](\overline{X}, \overline{X'}) \equiv [TC^d_{\overline{R},\overline{R'}} \phi \vee \psi](\overline{X}, \overline{X'})$$

with the appropriate renaming of bounded variables in order to avoid undesirable bind of free variables in ϕ. For the case of quantifiers, it is sufficient to apply the Lemmas 4.10 and 4.12. ∎

In the following, we will proof some theorems about the expressive power of BDSO(TC). We will see that BDSO(TC) is closely connected with the use of linearly many registers in a NRAM.

Definition 4.15 (SLIN) *The class SLIN is the class of queries over unary structures which can be computed in an NRAM using a number of registers linear in the cardinality of the structures and whose values stored in the registers during the computation is bounded by a linear function in the cardinality of the input structure.*

Lemma 4.13 *Any sentence of BDSO can be evaluated deterministically on a unary structure using a RAM which uses linearly many registers and each register stores a value which is linear on the cardinality of the structure.*

Proof. To evaluate a formula of the form $\forall^d X \phi$ it is sufficient to generate all possible values of X and check one by one if all off them satisfy ϕ. Since X ranges over bounded-degree relations of degree d, we only need linearly many registers to store X. After checking the formula ϕ for some value of X, we reuse the space used to store the current value of X to store the next value of X.

Similarly, to evaluate a formula of the type $\exists^d X\phi$ we just need to generate the bounded-degree relations of degree d and check one by one if some satisfies the formula ϕ. ∎

Theorem 4.6 *Each sentence of* BDSO(pos TC) *on a unary functional symbol set defines a query in SLIN.*

Proof. The strategy is similar to the one used to prove that FO(TC) $\subseteq$ SPACE[log$n$] [Imm99]. Let ϕ be a formula in BDSO(pos TC) on a unary functional symbol set S. Let $\overline{R}$ and $\overline{R'}$ be tuples of relation variables as in Definition 4.15. It is sufficient to show how to evaluate a formula of the type

$$\phi = [TC^d_{\overline{R},\overline{R'}}\psi(\overline{R},\overline{R'})](\overline{X},\overline{X'}).$$

Let $\mathfrak{A}$ be an S-structure and $\overline{\mathbf{X}},\overline{\mathbf{X}'}$ be interpretations for the relations in $\overline{X},\overline{X'}$. By Definition 4.13, we have:

$$(\mathfrak{A},\overline{\mathbf{X}},\overline{\mathbf{X}'}) \models [TC^d_{\overline{R},\overline{R'}}\psi(\overline{R},\overline{R'})](\overline{X},\overline{X'})$$

iff

$$(\overline{\mathbf{X}},\overline{\mathbf{X}'}) \in cl(\psi^{\mathfrak{A}}).$$

$(\overline{\mathbf{X}},\overline{\mathbf{X}'}) \in cl(\psi^{\mathfrak{A}})$ iff $\overline{\mathbf{X}'}$ can be reached from $\overline{\mathbf{X}}$ using transitions in $\psi^{\mathfrak{A},d}$. To check this we proceed in the following way. We existentially guess a tuple of relations $\overline{\mathbf{X}''}$ that interprets $\overline{X'}$. Then we check if $(\overline{\mathbf{X}},\overline{\mathbf{X}''}) \in \psi^{\mathfrak{A},d}$ by evaluating

$$(\mathfrak{A},\overline{\mathbf{X}},\overline{\mathbf{X}''}) \models \psi(\overline{R},\overline{R'}).$$

and check if $\overline{\mathbf{X}''}$ is equal to $\overline{\mathbf{X}''}$. If so, then there is a path between $\overline{\mathbf{X}}$ and $\overline{\mathbf{X}'}$. If not, then we substitute $\overline{\mathbf{X}''}$ for $\overline{\mathbf{X}}$ and existentially guess another relation $\overline{\mathbf{X}''}$ and repeat the process. If there is a path between $\overline{\mathbf{X}}$ and $\overline{\mathbf{X}'}$, we will eventually reach a $\overline{\mathbf{X}''}$ which is equal to $\overline{\mathbf{X}'}$. During this process, we do not need to store the entire path, but just check each possible step in a path from $\overline{\mathbf{X}}$ to $\overline{\mathbf{X}'}$ as described above. Hence we just have to store tree relations at a time, which can be stored using linearly many registers. ∎

Theorem 4.7 *If a class of unordered unary structures is in* SLIN *then it can be expressed in BDSO(pos TC).*

Proof. Without loss of generality we can suppose that the structures are

equipped with an order relation $<$ since we can state the existence of a linear order relation with a BDSO formula. We can also suppose the existence of a constant max for the greatest element with respect to this order relation. Let $\mathcal{C}$ be a class of unary S-structures in SLIN for some functional symbol set $S = \{f_0, \ldots, f_l\}$. There is a program $P = I(1), I(2), \ldots, I(h)$ for a NRAM that uses linearly many registers and each register stores a value which is linear in the cardinality of the input structure. Let us say that the number of registers used is bounded by cm and the values stored in the registers is bounded by $c'm$, where m is the cardinality of the structure. We can use tuples of elements of the structure to represent the value of the registers. We use a pair of elements rr' to represent a register and a pair of elements vv' which represent the value of a register. Hence we can represent the state of linearly many registers using relations of linear size. Let R be a 4-ary relation symbol. A tuple $(rr'vv')$ belongs to R iff vv' represents the value stored in the register represented by rr'. The register represented by rr' is the register R_k iff rr' is the k-th element with respect to the lexicographical ordering induced by $<$. The relation R will hold the values of the first cm registers. This can be stated with the formula

$$\begin{aligned}
\alpha(R) \quad = \quad & \exists vv' R(c-1)\ max\ vv' \wedge \\
& \forall rr' (rr' \leq (c-1)\ max \rightarrow \neg \exists vv' Rrr'vv') \wedge \\
& \forall rr' (\exists vv' Rrr'vv' \rightarrow \forall ss' (ss' <^2 rr' \rightarrow \exists ww' Rss'vv')) \wedge \\
& \forall rr'vv'ww' (Rrr'vv' \wedge Rrr'ww' \rightarrow vv' = ww').
\end{aligned}$$

We use a monadic relation I which is a singleton which stores an element which represents the next instruction to be executed on the configuration R. This can be stated with the formula

$$\beta(I) = \exists x I(x) \wedge \forall xy (I(x) \wedge I(y) \rightarrow (x = y \wedge x < h)),$$

where h is the number of instructions in P.

We will write a formula $\phi(I, R, I', R')$ that is satisfied iff R' is the configuration of the registers after executing the instruction I and I' is the next instruction to be executed. We will write a formula $\phi_j(I, R, I', R')$ for each $1 \leq j \leq h$ which is true iff R' is the state of the registers after executing instruction $I(j)$. The

formula $\phi(I, R, I', R')$ can be defined as

$$\phi(I, R, I', R') = \alpha(R) \wedge \alpha(R') \wedge \beta(I) \wedge \beta(I') \wedge \bigwedge_{1 \leq j \leq h} I(j) \rightarrow \phi_j(I, R, I', R').$$

For the sake of simplicity, we will consider only instructions involving registers. Accumulators can be treated similarly. In the following, we denote by $xy = k$ a first-order formula which say that xy is the k-th element in the lexicographic ordering. We will write below the formula ϕ_j for each type of instruction:

1. $I(j) = R_i := k$;

$$\begin{aligned} \phi_j \quad = \quad & \forall rr'(rr' \neq i \rightarrow \forall vv'(Rrr'vv' \leftrightarrow R'rr'vv')) \wedge \\ & \exists rr'vv'(rr' = i \wedge vv' = k \wedge R'rr'vv') \wedge \\ & \exists cc'(I(c) \wedge succ(c, c') \wedge \forall x(I'(x) \leftrightarrow x = c')). \end{aligned}$$

2. $I(j) = R_i := R_k$;

$$\begin{aligned} \phi_j \quad = \quad & \forall rr'(rr' \neq i \rightarrow \forall vv'(Rrr'vv' \leftrightarrow R'rr'vv')) \wedge \\ & \exists rr'ss'vv'(rr' = i \wedge ss' = k \wedge Rss'vv' \wedge R'rr'vv') \wedge \\ & \exists cc'(I(c) \wedge succ(c, c') \wedge \forall x(I'(x) \leftrightarrow x = c')). \end{aligned}$$

3. $I(j) = R_i := R_k + 1$;

$$\begin{aligned} \phi_j \quad = \quad & \forall rr'(rr' \neq i \rightarrow \forall vv'(Rrr'vv' \leftrightarrow R'rr'vv')) \wedge \\ & \exists rr'vv'ss'ww'(ss' = k \wedge Rss'ww' \wedge succ^2(ww', vv') \wedge \\ & \qquad rr' = i \wedge R'ss'vv') \wedge \\ & \exists cc'(I(c) \wedge succ(c, c') \wedge \forall x(I'(x) \leftrightarrow x = c')). \end{aligned}$$

4. $I(j) = R_i := R_k \dot{-} 1$;

$$\begin{aligned} \phi_j \quad = \quad & \forall rr'(rr' \neq i \rightarrow \forall vv'(Rrr'vv' \leftrightarrow R'rr'vv')) \wedge \\ & \exists rr'vv'ss'ww'(ss' = k \wedge Rss'ww' \wedge rr' = i \wedge \\ & \qquad (succ^2(vv'ww') \vee \neg succ^2(vv'ww') \wedge vv' = 0) \wedge R'rr'vv') \wedge \\ & \exists cc'(I(c) \wedge succ^2(c, c') \wedge \forall x(I'(x) \leftrightarrow x = c')). \end{aligned}$$

5. $I(j) = R_i := R_{R_k}$;

$$\begin{aligned}\phi_j \quad = \quad & \forall rr'(rr' \neq i \to \forall vv'(Rrr'vv' \leftrightarrow R'rr'vv')) \wedge \\ & \exists rr'vv'ss'ww'(rr' = i \wedge ss' = k \wedge \\ & \qquad Rss'ww' \wedge Rww'vv' \wedge R'rr'vv') \wedge \\ & \exists cc'(I(c) \wedge succ(c,c') \wedge \forall x(I'(x) \leftrightarrow x = c')).\end{aligned}$$

6. $I(j) = R_{R_k} := R_i$;

$$\begin{aligned}\phi_j \quad = \quad & \exists rr'vv'(rr' = k \wedge Rrr'vv' \wedge \\ & \qquad \forall ss'ww'(ss' \neq vv' \to Rss'ww' \leftrightarrow R'ss'ww')) \wedge \\ & \exists rr'vv'(rr' = k \wedge Rrr'vv' \wedge \\ & \qquad \exists ss'ww'(ss' = i \wedge Rss'ww' \wedge R'vv'ww')) \wedge \\ & \exists cc'(I(c) \wedge succ(c,c') \wedge \forall x(I'(x) \leftrightarrow x = c')).\end{aligned}$$

7. $I(j) = guess\ (R_i)$;

$$\begin{aligned}\phi_j \quad = \quad & \forall rr'(rr' \neq i \to \forall vv'(Rrr'vv' \leftrightarrow R'rr'vv')) \wedge \\ & \exists rr'(rr' \neq i \wedge \exists ss'vv'(Rss'vv' \wedge Rrr'vv') \wedge \\ & \exists cc'(I(c) \wedge succ(c,c') \wedge \forall x(I'(x) \leftrightarrow x = c')).\end{aligned}$$

8. $I(j) = goto\ I(i_0)\ or\ I(i_1)$;

$$\begin{aligned}\phi_j \quad = \quad & \forall rr'vv'(Rrr'vv' \leftrightarrow R'rr'vv') \wedge \\ & \forall x(I'(x) \leftrightarrow x = i_0) \vee \forall x(I'(x) \leftrightarrow x = i_1).\end{aligned}$$

9. $I(j) = if\ X_i = X_k\ then\ goto\ I(i_0)\ else\ goto\ I(i_1)$;

$$\begin{aligned}\phi_j \quad = \quad & \forall rr'vv'(Rrr'vv' \leftrightarrow R'rr'vv') \wedge \\ & \exists rr'vv'ss'ww'(Rrr'vv' \wedge rr' = i \wedge Rss'ww' \wedge ss' = k \wedge \\ & \qquad (vv' = ww' \to \forall x(I'(x) \leftrightarrow x = i_0)) \wedge \\ & \qquad (vv' \neq ww' \to \forall x(I'(x) \leftrightarrow x = i_1))).\end{aligned}$$

10. $I(j) = accept$;

$$\begin{aligned}\phi_j &= \forall rr'vv'(Rrr'vv' \leftrightarrow R'rr'vv') \wedge \\ &\quad \exists cc'(I(c) \wedge succ(c,c') \wedge \forall x(I'(x) \leftrightarrow x = c')).\end{aligned}$$

11. $I(j) = reject$;

$$\begin{aligned}\phi_j &= \forall rr'vv'(Rrr'vv' \leftrightarrow R'rr'vv') \wedge \\ &\quad \exists cc'(I(c) \wedge succ(c,c') \wedge \forall x(I'(x) \leftrightarrow x = c')).\end{aligned}$$

We will define now a formula $\psi(I,R)$ which says that I,R is the initial configuration of a RAM that receives an S-structure $\mathfrak{A}$ as input:

$$\psi(I,R) = \forall x(I(x) \leftrightarrow x = 0) \wedge \bigwedge_{1 \leq i \leq l} \forall x Rix0f_i x$$

Let $a_1, \ldots, a_c$ be the numbers of the lines in the program P which hold an *accept* instruction. The following formula says that I,R is and accepting configuration:

$$ACC(I) = \bigvee_{1 \leq i \leq c} I(i)$$

As we saw in Section 4.4, relations of linear size can be replaced with relations of bounded degree. Hence, instead of using a 4-ary relation R of linear size, we can use a tuple of bounded-degree relations $\overline{R}$ of degree d for some d to codify the state of the registers. And there is a formula $\phi'(I,\overline{R},I',\overline{R'})$ similar to $\phi(I,R,I'R')$ that says that $I,\overline{R}$ and $I',\overline{R'}$ are consecutive configurations of an NRAM that decides $\mathcal{C}$ and a formula $\psi'(I,\overline{R})$ which says that $I,\overline{R}$ is the initial configuration of the RAM. The formula

$$\begin{aligned}\Phi &= \exists^d I_0 \overline{R}_0 I_F \overline{R'}_F(\psi'(I_0,\overline{R}_0) \wedge ACC(I_F) \wedge \\ &\qquad [TC^d_{I,\overline{R},I',\overline{R'}}\phi'(I,\overline{R},I',\overline{R'})](I_0,\overline{R}_0,I_F,\overline{R}_F))\end{aligned}$$

expresses $\mathcal{C}$.

∎

It remains to show that BDSO(TC) = BDSO(pos TC).

Theorem 4.8 BDSO(TC) = BDSO(pos TC).

Proof. The proof is similar to the one used to prove that FO(TC) = FO(pos TC) (see, for instance, [Imm99]). We just have to show how to eliminate negations in front of transitive closure formulas. We will show the proof for the case

$$\neg[TC^d_{R,R'}\phi(R,R')](X,Y),$$

where R, R', X and Y are k-ary relations. The proof for the general case is similar.

We write formulas $DIST(X,Y,D)$ which says that there is a path of size at most D through $\phi^{\mathfrak{A},d}$ transitions from X to Y and $NDIST(X,Y,D,M)$ which says that there are at least M relations different from X whose distance to Y is at most D (D and M regarded as the numbers which correspond to their positions in the ordering $\leq_k$ of Definition 4.14). When M is the number of relations whose distance to Y is at most D, then $NDIST(X,Y,D,M)$ is equivalent to $\neg[TC^d_{R,R'}\phi(R,R')](X,Y)$. The formula $DIST(X,Y,D)$ can be written as (see [Imm99]):

$$DIST(X,Y,D) = [TC^d_{R,C,R',C'} succ^d_k(C,C') \wedge (\phi(R,R') \vee R = R')](Y,\emptyset,X,D).$$

Now let us define $NDIST(X,Y,D,M)$. First, let $\beta(R,C,R',C')$ be the following formula:

$$\begin{aligned}\beta(R,C,R',C') \quad = \quad & X \neq Y \wedge succ^d_k(R,R') \wedge \\ & (c = c' \vee (succ^d_k(C,C') \wedge DIST(R',Y,D) \wedge R' \neq X)).\end{aligned}$$

This formula says that, if there are C relations different from X less than or equal to R from which Y can be reached in at most D steps, then either Y is reachable from R' in D steps and there are $C+1$ relations less than or equal to R' different from X from which Y can be reached in at most D or there are C relations less than or equal to R' from which Y can be reached in D steps. The formula $NDIST(X,Y,D,M)$ can be defined as:

$$NDIST(X,Y,D,M) = [TC^d_{R,C,R',C'}\beta(R,C,R',C')](\emptyset,1,A,M).$$

We use the formulas $DIST$ and $NDIST$ to write an equivalent to

$$\neg[TC^d_{R,R'}\phi(R,R')](X,Y)$$

without using negated transitive closure operators.

Let $\delta(d, m, d', m')$ be the formula

$$\delta(D, M, D', M') = succ_k^d(D, D') \wedge [TC^d_{R,C,R',C'}\gamma(R, C, R', C')](\emptyset, 1, A, M')$$

where

$$\begin{aligned}\gamma(R, C, R', C') \quad = \quad & succ_k^d(R, R') \wedge ((succ_k^d(C, C') \wedge DIST(R', Y, D')) \vee \\ & (C = C' \wedge \forall^d Z(NDIST(Z, Y, D, M) \vee \\ & \qquad (Z \neq R' \wedge \neg\phi(R, R'))))).\end{aligned}$$

The formula

$$[TC^d_{D,M,D',M'}\delta(D, M, D', M')](\emptyset, 1, A, M)$$

is satisfied iff M is the number of relations from which Y can be reached. Then we have:

$$\neg[TC^d_{R,R'}\phi(R, R')](X, Y)$$

$$\equiv$$

$$\exists M([TC^d_{D,M,D',M'}\delta(D, M, D', M')](\emptyset, 1, A, M) \wedge NDIST(X, Y, A, M)).$$

∎

Corollary 4.2 BDSO(TC) = BDSO(pos TC).

Corollary 4.3 BDSO(TC) *captures* SLIN.

4.6 Conclusions

In this chapter we studied the descriptive complexity of a restriction of second-order logic where quantifiers range over relations of bounded degree. In Section 4.2 we study the descriptive complexity of MSO(f), the fragment of SO where unary functions are quantified. Based on previous work from Grandjean and Olive [GO98], we showed that MSO(f) captures the class ALIN of sets of unary structures which can be accepted by a ARAM in linear alternating time with bounded number of alternations.

In Section 4.3, we define the BDSO logic. We defined three concepts of degree and showed that they are equivalent. We showed that the degree hierarchy

collapses, since any formula of BDSO can be written with quantification of relations of simple degree at most 1.

In Section 4.4, we proved that quantification of bounded degree relations is equivalent to quantification of relations of linear size and functions. Although $\exists$MSO(f) be more expressive than $\exists$BDSO, the equivalence holds for the entire logics, considering both existential and universal quantification. It follows that any formula from BDSO can be written using only quantification of binary relations of simple degree 1. Based on results from Schwentick et al. [DLS98], we showed that BDSO captures ALIN.

In Section 4.5, we extend BDSO with the transitive closure operator. We showed that BDSO(TC) captures the class SLIN of classes of unary structures that are accepted by an NRAM using linearly many registers provided that registers hold values which are linear in the cardinality of the input structure.

Chapter 5

Conclusions

In this work we investigated the model theory of logics whose semantics differ from the classical approach either by considering relations between models or by restricting models to finite structures. We studied preferential logics, which use the concept of minimal model induced by binary relations between models to define nonmonotonic consequence relations. We investigated the finite model theory of hybrid logic with respect to the expressibility of graph properties in PH and the descriptive complexity of the BDSO and BDSO(TC) logics.

In Chapter 2 we investigated the model theory of classes of preferential logics, introduced by Shoham in [Sho87], using the approach of abstract model theory. We extended the concept of abstract logic to include a preference relation. It allows us to define nonmonotonic consequence relations by using the concept of minimal models. Our objective was to generalize some results presented in [FM11b] for Circumscription to classes of elementary preferential logics. In Section 2.2, we set up the framework on which we proved our results. We defined the concept of definable preferential logic and elementary preferential logic, whose preferential relation is defined in first-order logic. We also made some assumptions on the nature of the satisfiability relation in Definition 2.7 since we worked with structures as models. In Section 2.3 we gave some examples of elementary preferential logics. We showed that McCarthy's Circumscription and Reiter's Default Logic with normal default rules without precondition are elementary preferential logics. That is, they can be presented as preferential logics whose preference relation can be defined by a first-order formula. We are interested in the expressiveness of elementary preferential logics with respect to the expressibility of classes of minimal models. Are there classes of minimal

models of sets of sentences of some elementary preferential logic $\mathcal{L}' = (\mathcal{L}, \prec)$ which cannot be expressed in $\mathcal{L}$? The answer is positive, since there are classes of minimal models of first-order sentences which cannot be expressed in first-order logic, even if one uses infinite theories, as shown by Schlipf [Sch87]. In Section 2.4 we showed that in the case the class of minimal models of a finite theory of any EPL $\mathcal{L}'$ is $\mathcal{L}$-Δ-expressible, then it is $\mathcal{L}$-expressible, that is if it can be axiomatized in $\mathcal{L}$, which means that it can be expressed without the necessity of considering minimal models, then it is finitely axiomatizable. Our assumptions are that the abstract logic $\mathcal{L}$ has some good properties, like compactness and relativization. In Section 2.5 we used the result of the previous section to solve a problem of definability of symbols of the language. We showed that when some symbol, say P, is defined in the class of minimal models of a finite set of sentences in $\mathcal{L}'$ and such class is axiomatizable in $\mathcal{L}$, then not only such class is finitely axiomatizable, but it admits a very specific axiomatization, namely, the initial set of sentences plus an explicit definition for the symbol P.

In Chapter 3 we studied the finite model theory of hybrid logic. In [BS09], Benevides e Schechter show formulas in hybrid logic that express some problems in NP for graphs of limited size. That is, for each natural number n they present a formula that express some problem in NP, like hamiltonicity, for graphs of size n. In this way, they can reduce those problems to the problem of model checking for those formulas. Is that possible for any graph property in NP? Actually, we can do this for any graph property even in a very restrict fragment of hybrid logic. That is because it is possible to describe any graph up to isomorphism, with respect to frame definability, using just $\Diamond$, @ and nominals, as we showed in Section 3.3. The size of the formulas obtained is exponential in n and, of course, there is no hope that those formulas forms a recursive set. However, we can show that, for problems in the Polynomial Hierarchy, there are formulas of polynomial size and which can be easily generated. From Fagin's Theorem it follows that PH is captured by Second-Order Logic. That is, for each graph property in PH there is an SO-formula which express exactly this property. Using this SO-formula we constructed for each n a formula of full hybrid logic which is valid only in those frames which correspond to graphs which has the desired property. These formulas have size polynomial in n and can be easily generated from the SO-formula. Full hybrid logic has the same expressive power as first-order logic. That is because we can simulate first-order quantification using the binder $\downarrow x.$ and the global modalities E and A. If we consider only connected frames with loops, we do not need the global modalities

or nominals. In Section 3.5 we showed that for any graph property in PH on connected graphs with loops there are formulas of hybrid logic without global modalities or nominals which express that property for graphs of size n and whose size is bounded by a polynomial in n. Using the results of the previous sections, we showed in Section 3.6 an alternative proof of the NP-hardness of the model-checking problem for the fragment $\mathrm{FHL}\backslash\downarrow\Box\downarrow$.

In Chapter 4 we investigated the descriptive complexity of the restriction of second-order logic where second-order quantifiers range only over relations of bounded degree. In [DLS98] Schwentick et al. studied the expressiveness of several kinds of existential quantification over binary relations. They divide the fragments of SO defined by those quantifiers in four groups, as listed in Section 4.4. They are concerned only with the existential fragment over binary relations. Each group corresponds to a level of expressiveness and strictly contains the groups below. Among those types of quantification is the existential quantification of binary relations of bounded Gaifman degree. Another kind of quantification studied by Schwentick is the existential quantification of unary functions. The second-order existential formulas constructed with the quantifier $\exists^{\mathcal{G},d}$, which quantifies bounded-degree relations, over binary relations have equivalent using the quantifier $\exists^{f}$ which quantifies function, but the contrary does not hold [DLS98]. In Section 4.4 we gave a proof that the logics BDSO and MSO(f) are equivalent. Beside this, the arity of relations does not play a role. In fact, one can consider only quantification of binary relations of bounded degree since the arity hierarchy collapses, as we showed in Theorem 4.3. We are interested in the descriptive complexity of BDSO. In [GO98], Grandjean and Olive showed that a subset of the existential fragment of MSO(f) captures linear time on ordered unary structures. Based in the work of Grandjean and Olive and Schwentick et al., we showed that BDSO captures the class ALIN of linear alternating time over (unordered) unary structures. In Section 4.5, we introduced the logic BDSO(TC) that extends BDSO with the transitive closure operator on high-order relations over relations of bounded degree, that is, relations of relations of bounded degree. We showed that it capture the class SLIN of problems that can be solved by an NRAM using linear number of registers and provided that values stored in the registers are linear in the cardinality of the input.

Bibliography

[ABM01] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic*, 66(3):977–1010, 2001.

[AtC07] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of modal logic*, volume 3, pages 821–868. Elsevier Science Ltd, 2007.

[AV91] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43(1):62–124, 1991.

[Bar96] V. C. Barbosa. *An introduction to distributed algorithms*. The MIT Press, 1996.

[BdFV01] J.-Y. Beziau, R. P. de Freitas, and J. P. Viana. What is classical propositional logic?(a study in universal logic). *Logical Studies*, 7, 2001.

[BDRV02] P. Blackburn, M. De Rijke, and Y. Venema. *Modal logic*. Cambridge Univ Pr, 2002.

[Bet53] E. W. Beth. On Padoa's method in the theory of definitions. *Indagtiones Mathematicae*, 15, 1953.

[Bez06] J.-Y. Beziau. 13 Questions about universal logic. *Bulletin of the Section of Logic*, 35:133–150, 2006.

[Bez07] J.-Y. Beziau. From consequence operator to universal logic: a survey of general abstract logic. In Jean-Yves Beziau, editor, *Logica Universalis*, pages 3–18. Birkhäuser, 2nd edition, 2007.

[BF85] J. Barwise and S. Feferman, editors. *Model-Theoretic Logics*. Springer, New York, 1985.

[Bla06] P. Blackburn. Arthur prior and hybrid logic. *Synthese*, 150(3):329–372, 2006.

[Bre91] G. Brewka. *Nonmonotonic reasoning: logical foundations of commonsense*, volume 12. Cambridge Univ Pr, 1991.

[BS07] J. Bradfield and C. Stirling. Modal mu-calculi. *Handbook of Modal Logic*, 3:721–756, 2007.

[BS09] M. R. F. Benevides and L. M. Schechter. Using modal logics to express and check global graph properties. *Logic Journal of IGPL*, 17(5):559, 2009.

[Büc60] J. R. Büchi. Weak second order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

[CK73] C. C. Chang and J. Keisler. *Model Theory*. Number 73 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1973. Third edition, 1990.

[DLS98] A. Durand, C. Lautemann, and T. Schwentick. Subclasses of Binary NP. *Journal of Logic and Computation*, 8(2):189–207, 1998.

[EF95] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.

[EFT94a] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, New York, NY, 1994.

[EFT94b] H.D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical logic*. Springer, 1994.

[Fag74a] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R.Karp, editor, *Complexity of Computation*, volume 7 of *SIAM-AMS Proceedings*, pages 43–73. AMS, 1974.

[Fag74b] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity and Computation*, volume 7, pages 43–73. SIAM-AMS Proceedings, 1974.

[FFB+11] F. Ferreira, C. Freire, M. Benevides, L. Schechter, and A. Martins. Hybrid logics and np graph properties. In Lev Beklemishev and Ruy de Queiroz, editors, *Logic, Language, Information and Computation*, volume 6642 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin / Heidelberg, 2011.

[FM11a] F. M. Ferreira and A. T. Martins. Expressible preferential logics. *Journal of Logic and Computation*, 2011.

[FM11b] F. M. Ferreira and A. T. Martins. Expressiveness and definability results in circumscription. *Manuscrito*, 34(1):195–227, 2011.

[Gab85] D. Gabbay. Theoretical foundations for non-monotonic reasoning in expert systems. In *Logics and models of concurrent systems*, pages 439–457, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[GMV07] M. García-Matos and J. Väänänen. Abstract Model Theory as a Framework for Universal Logic. In Jean-Yves Beziau, editor, *Logica Universalis*, pages 19–34. Birkhäuser, 2nd edition, 2007.

[GO98] E. Grandjean and F. Olive. Monadic logical definability of nondeterministic linear time. *Computational Complexity*, 7(1):54–97, 1998.

[Gor94] V. Goranko. Temporal logic with reference pointers. *Temporal logic*, pages 133–148, 1994.

[Gor96] V. Goranko. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, 5(1):1–24, 1996.

[Grä90] E. Grädel. On the notion of linear time computability. *International Journal of Foundations of Computer Science*, 1, 1990.

[Grä92] E. Grädel. Capturing complexity classes by fragments of second-order logic. *Theoretical Computer Science*, 101:35–57, 1992.

[Grä01] E. Grädel. Why are modal logics so robustly decidable? In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science. Entering the 21st Century*, pages 393–408. World Scientific, 2001.

[Grä02] E. Grädel. Guarded fixed point logics and the monadic theory of countable trees. *Theoretical Computer Science*, 288(1):129–152, 2002.

[GS89] Yuri Gurevich and Saharon Shelah. Nearly linear time. In Albert R. Meyer and Michael A. Taitslin, editors, *Logic at Botik '89, Symposium on Logical Foundations of Computer Science, Pereslav-Zalessky, USSR, July 3-8, 1989, Proceedings*, volume 363 of *Lecture Notes in Computer Science*, pages 108–118. Springer, 1989.

[HAK89] W. Haken, K. Appel, and J. Koch. *Every planar map is four colorable*, volume 98 of *Contemporary Mathematics*. American Mathematical Society, 1989.

[HJ99] K. Hrbacek and T. Jech. *Introduction to Set Theory*. Marcel Dekker, 1999.

[Hum07] D. Hume. *An Enquiry Concerning Human Understanding*. 1748. Oxford University Press, 2007.

[Imm82] N. Immerman. Relational queries computable in polynomial time. In *ACM Symposium on Theory of Computing (STOC '82)*, pages 147–152, Baltimore, USA, 1982. ACM Press.

[Imm83] N. Immerman. Languages which capture complexity classes. In *15th ACM STOC Symposium*, pages 347–354, 1983.

[Imm99] N. Immerman. *Descriptive Complexity*. Graduate Texts in Computer Science. Springer, New York, 1999.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artif. Intell.*, 44(1-2):167–207, 1990.

[Lib04] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[Lif94] V. Lifschitz. Circumscription. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming: Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3, pages 297–352. Clarendon Press, Oxford, 1994.

[Lin69] P. Lindström. On extensions of elementary logic. *Theoria*, 35(1):1–11, 1969.

[LM90] D. Lehmann and M. Magidor. Preferential logics: the predicate calculus case. In *TARK '90: Proceedings of the 3rd conference on Theoretical aspects of reasoning about knowledge*, pages 57–72, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[Lyn92] J. F. Lynch. The quantifier structure of sentences that characterize nondeterministic time complexity. *Computational Complexity*, 2(1):40–66, 1992.

[Lyn96] N.A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[McC80] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

[McC86] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1), 1986.

[MD80] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13(1-2):41–72, 1980.

[MTR93] V.W. Marek, M. Truszczyński, and R. Reiter. *Nonmonotonic logic*, volume 393. Springer, 1993.

[Pad00] A. Padoa. Essai d'une théorie algébrique des nombres entiers, precedé d'une introduction logique a une théorie deductive quelconque. In *Bibliothèque Du Congrès International de Philosophie*, volume 3, pages 118–123, 1900.

[Pap03] C.H. Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.

[Poo94] D. Poole. Default logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming: Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3, pages 189–215. Clarendon Press, 1994.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[Rei82] R. Reiter. Circumscription implies predicate completion (sometimes). In *Proc. of AAAI-82*, pages 418–420, Pittsburgh, PA, 1982.

[RSST97] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2 – 44, 1997.

[Sch87] J. Schlipf. Decidability and definability with circumscription. *Annals of Pure and Applied Logic*, 35(2):173–191, 1987.

[Sho87] Y. Shoham. A semantical approach to nonmonotonic logics. In *Readings in nonmonotonic reasoning*, pages 227–250, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[tCF05] B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In L. Ong, editor, *Proceedings of Computer Science Logic 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer Verlag, 2005.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *ACM Symposium on Theory of Computing (STOC '82)*, pages 137–146, Baltimore, USA, May 1982. ACM Press.

[Var96] M.Y. Vardi. Why is modal logic so robustly decidable. In N. Immerman and P. G. Kolaitis, editors, *Descriptive complexity and finite models*, volume 31 of *Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1996.