



Universität Bremen
Fachbereich Mathematik und Informatik

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

Conservative Extensions and Satisfiability in Fragments of First-Order Logic: Complexity and Expressive Power

Mauricio Martel

Kolloquium: 01.08.2018

Gutachter:

Prof. Dr. Carsten Lutz, Universität Bremen

Prof. Dr. Ulrike Sattler, Universität Manchester

Abstract

First-order logic is a nice language with high expressive power and a well developed model theory, however, its satisfiability problem is undecidable. When we think about applications requiring effective reasoning problems, it is natural to look for fragments of first-order logic with better computational properties.

In this thesis, we investigate the decidability and computational complexity of (deductive) conservative extensions in expressive fragments of first-order logic, such as the two-variable fragment, the guarded fragment, and the guarded negation fragment. Moreover, we also investigate the complexity of (query) conservative extensions in Horn description logics with inverse roles.

Additionally, we investigate the computational complexity of the satisfiability problem in the unary negation fragment of first-order logic extended with regular path expressions. We study the satisfiability problem in order to obtain complexity results for the ontology-mediated query answering problem.

Besides computational complexity results, we also study the expressive power of a particular family of modal logics that can update the accessibility relation of a model during the evaluation of a formula. In particular, we provide translations into hybrid logic and compare their expressive power using appropriate notions of bisimulations.

Zusammenfassung

Prädikatenlogik ist eine grundlegende Logik mit hoher Ausdrucksstärke und einer gut untersuchten Modelltheorie, deren Erfüllbarkeitsproblem jedoch unentscheidbar ist. Da in vielen Anwendungen effektive Verfahren zum Schlussfolgern erforderlich sind, liegt es nahe Fragmente von Prädikatenlogik zu untersuchen, die diesbezüglich bessere Eigenschaften haben.

In dieser Arbeit untersuchen wir zuerst die Entscheidbarkeit und Berechnungskomplexität von (deduktiven) konservativen Erweiterungen in ausdrucksstarken Fragmenten der Prädikatenlogik, wie dem Zwei-Variablen-Fragment, dem Guarded-Fragment und dem Guarded Negation-Fragment. Wir untersuchen weiterhin die Komplexität von (query) konservativen Erweiterungen in Horn-Beschreibungslogiken mit inversen Rollen.

Außerdem erweitern wir das Unäre-Negationen-Fragment der Prädikatenlogik um reguläre Pfad-Ausdrücke und analysieren die Berechnungskomplexität des Erfüllbarkeitsproblems in dieser Logik. In diesem Zusammenhang erhalten wir auch Resultate für das eng verwandte Problem der Ontologie-basierten Anfragebeantwortung in der eingeführten Logik.

Darüber hinaus betrachten wir eine Familie von modalen Logiken, bei denen während der Formelauswertung der unterliegende Frame geändert werden kann. Wir geben Übersetzungen von diesen Logiken in hybride Logik an und vergleichen mit Hilfe von entsprechend definierten Bisimulationen ihre Ausdrucksstärke mit der von hybriden Logiken.

Contents

1	Introduction	1
1.1	Logic in Computer Science	1
1.2	Fragments of First-Order Logic	2
1.2.1	Complexity of Conservative Extensions	4
1.2.2	Complexity of Satisfiability and Ontology-mediated Queries	6
1.2.3	Investigating Expressive Power	8
1.3	Overview of the Thesis	10
2	Preliminaries	13
2.1	First-Order Logic	13
2.2	Two-Variable, Guarded, and Unary Negation Fragments	14
2.3	Description Logic	15
2.4	Modal Logic	17
3	Conservative Extensions in Guarded and Two-Variable Fragments	19
3.1	Deductive Conservative Extensions	21
3.2	Undecidability Results	23
3.2.1	The Guarded Fragment	23
3.2.2	The Two-Variable Fragment	26
3.3	Model-theoretic Characterization	29
3.3.1	GFO ² -Bisimulations	29
3.3.2	Characterization of Σ -Entailment	32
3.4	Decidability and Complexity	38
3.4.1	2ATAs and their Emptiness Problem	38
3.4.2	Upper Bound	42
3.4.3	Lower Bound	48
3.5	Concluding Remarks	52
4	Conservative Extensions in Horn Description Logics with Inverse Roles	55
4.1	Horn- <i>ALCHIF</i>	56
4.2	Query Conservative Extensions and Entailment	57
4.2.1	Query Entailment with Inconsistent ABoxes	58
4.3	Model-theoretic Characterization	63

4.3.1	Unraveling ABoxes	63
4.3.2	Characterization of Query Entailment	65
4.4	Decidability and Complexity	70
4.4.1	Mosaic Technique	70
4.4.2	Automata-Based Technique	76
4.5	Deductive Conservative Extensions	87
4.5.1	Lower Bound	89
4.6	Concluding Remarks	97
5	Ontology-mediated Querying in UNFO with Regular Path Expressions	99
5.1	UNFO with Regular Path Expressions	101
5.1.1	Ontology-mediated Querying	102
5.2	Model-theoretic Characterization	105
5.2.1	Normal Form	106
5.2.2	Tree-like Structures	107
5.2.3	Characterization of Satisfiability	108
5.3	Decidability and Complexity	114
5.4	OMQ Evaluation and Containment	117
5.4.1	Data Complexity	117
5.5	Model Checking	121
5.6	Concluding Remarks	124
6	Relation-Changing Modal Logics as Fragments of Hybrid Logics	127
6.1	Relation-Changing Modal Logics	129
6.2	Extensions of Modal Logic and Hybrid Logic	130
6.3	Translations to Hybrid Logics	131
6.4	Decidable Fragments	135
6.5	Comparing Expressive Power	136
6.6	Concluding Remarks	139
7	Conclusions	141
	Bibliography	143

1.1 Logic in Computer Science

Logic plays a fundamental role in computer science. We can consider computer science both as an engineering and a mathematical discipline. From a practical perspective, the aim is to design algorithms to solve problems with computer programs. In contrast, we would not be able to identify what kind of problems are solvable, or how hard it is to solve a problem, if we do not study computer science from a theoretical perspective. The subfield of theoretical computer science, in particular, is mathematical and abstract in spirit, but it derives its motivation from practical applications in real life. Logic has close ties with many subjects of computer science such as computability and complexity theory [LP97, Pap94], automata theory [GTW02], and databases [AHV95]. In this thesis, we touch on these subjects and study logics that are relevant for a theoretical study and also of practical importance.

Most of the thesis is about decidability and computational complexity results. In computability theory, a *decision problem* is a problem that can be posed as a yes-no question of the input values. There is an extensive study and classification of which mathematical problems are decidable (or computable, or effective) and which are not. Complexity theory, on the other hand, investigates how much time and space is needed to solve a particular computable problem. In addition, there is an extensive classification of computable problems into *complexity classes* according to how much computation, as a function of the size of the problem instance, is needed to answer that instance. For example, PTIME is the class of all problems that can be solved by a deterministic Turing machine in polynomial time. Other classical complexity classes include the class NP, PSPACE, EXPTIME, and NEXPTIME, among others. There are different techniques to obtain complexity results, but one technique that has been broadly used over the years consists in exploiting the relationship between logic and automata theory.

The connections between logic and automata theory are long and fruitful. They have come together in the 1960s through the fundamental work of Büchi, Elgot, Rabin and others [Bü60, Elg61, Bü62, Rab69, McN66] who showed, among other results, the equivalence of automata with logical systems such as monadic second-order logic on finite and infinite words and trees. In the latter case, the ‘moving to trees’ involves

showing that the logic possesses the tree model property: if a formula φ is satisfiable in a model, it is satisfiable in a tree-shaped model. Then decidability results for automata can be used to obtain decidability results for the logics we are interested in. Given a logical formula φ , we build an automaton \mathcal{A}_φ that recognizes the set of all models in which φ holds. Assuming that \mathcal{A}_φ belongs to a well-behaved class of automata, we can decide the satisfiability problem via a reduction to the non-emptiness problem of the automaton: φ has a model if and only if the language of \mathcal{A}_φ is not empty. Solving the emptiness problem of \mathcal{A}_φ is usually decidable for many well-behaved classes of automata and the exact complexity depends on the particular automaton model used. We will use automata-techniques extensively thorough the thesis to obtain complexity results.

Logic provides a foundation not only in theoretical areas of computer science but also in many applied areas such as in databases, which is an area concerned with storing, querying, and updating large amounts of data. Logic and databases are inextricably intertwined since the early 1970s [Cod70] given that logic provides both a unifying framework and a set of tools for formalizing and studying data management tasks. Logic can be used as a database query language to express questions asked against databases, and in fact, *conjunctive queries* and unions thereof are useful and decidable query languages that will be of particular importance for this thesis. We will study several query languages and the computational complexity of answering queries with semantic background knowledge, as well as related reasoning problems. This interaction between logic and databases is a prime example of applications of logic in computer science.

As we have seen, several areas of computer science are related to logic. But so far we haven't mentioned which logic we are talking about. For a long time, logic was associated with first-order logic (FO) [End01]. First-order logic is by now a well understood language with high expressive power and a rich model theory. But regarding its computational behavior, FO doesn't seem to behave very well: its satisfiability problem, i.e., the problem of determining whether there exists a model in which a given formula is true, is undecidable [Chu36, Tur37]. Its model checking problem, i.e., the problem of determining whether a given formula is true in a give model, is however decidable and PSPACE-complete [Sto74, Var82]. On the other hand, FO sometimes is not expressive enough as it cannot express, for example, the transitive clousure of a relation, and this might be important for certain application scenarios. For these reasons, when we think in areas that requiere applications with effective reasoning problems, such as artificial intelligence, knowledge representation, or databases, first-order logic might not be the best option. It is natural then to look for fragments with better computational properties, and fortunately, many decidable fragments of FO have been studied through the years. Next we discuss the fragments that are relevant for this thesis.

1.2 Fragments of First-Order Logic

Many different logics have been studied in computer science. In particular, modal logics [BdRV01] proved to be very useful in a number of areas where the fundamental concepts needed can be expressed in terms of graphs-like structures. The main reason for their effectiveness is their careful balance between expressive power and computational complexity. Indeed, several modal logics have been successfully tailored in such a way that they are sufficiently expressive to specify interesting properties of

a particular application and, at the same time, provide efficient algorithms for their main computational problems, such as satisfiability and model checking. Equivalent formalisms from a different application area are description logics [BHL17], a family of knowledge representation languages that underly the Web Ontology Language OWL. Description logics mainly fall into two categories. Expressive description logics, on the one hand, such as \mathcal{ALC} and extensions, aim at maximizing expressive power while still retaining decidability of standard reasoning problems, such as satisfiability and subsumption. On the other hand, lightweight description logics, such as \mathcal{EL} and DL-Lite, aim at tractability of standard reasoning problems and at scalability to very large ontologies, which requires to significantly reduce expressive power. Therefore, modal and description logics aim at establishing an attractive compromise between expressive power on one side and computational complexity on the other.

Modal and description logics are known for their *robust decidability* [Var96], meaning that the decidability of basic reasoning problems, such as satisfiability and validity, is preserved under various extensions to the syntax and semantics, for example, by the addition of transitive closure operators, inverse roles, fixpoint operators, or nominals. To understand the robust decidability of modal and description logics, it is useful to see them as fragments of first-order logic: every formula can be translated into an equivalent first-order formula with one free variable. This translation yields a small fragment that is properly contained in FO^2 , a fragment of first-order logic with only two variables. FO^2 has the finite model property, i.e., every satisfiable formula has a finite model, and is known to be decidable [Mor75, GKV97]. But despite this observation, FO^2 is highly undecidable when extended with transitive closure operators, least and greatest fixed points, etc. [GOR97, GO99], in contrast with the respective extensions in modal and description logics. Therefore, it seems the translation into FO^2 does not give a satisfactory explanation of the robust decidability of modal and description logics.

By taking a closer look at the translation of modal and description logic into first-order logic (see Section 2.3 for details), one can observe that the quantifiers are used only in a very restricted way, and this observation was what gave rise to the guarded fragment of first-order logic (GFO) [ANv98]. The restriction to use only two variables and only unary and binary predicates is dropped, but what is imposed instead is that all quantifiers must be relativized by atomic formulas. The satisfiability problem in GFO is decidable, it has the tree-like model property, i.e., if a sentence has a model then it has a model of bounded tree width, and it also has the finite model property [ANv98, Grä99]. Moreover, many important model theoretic properties which do not hold for FO^2 , but do hold for modal and description logics, hold also for the guarded fragment [GR99].

The most recent proposal to explain the good computational behavior of modal and description logics is based on restricting not the number of variables or the quantification pattern, but the use of negation. The unary negation fragment (UNFO) restricts first-order logic by constraining the use of negation to subformulas having at most one free variable. It generalizes modal and description logics and can express conjunctive queries and unions thereof (see Definition 2.3.1) as formulas in the language, which is interesting from a database perspective. The satisfiability problem for UNFO is decidable, it has the tree-like model property and the finite model property. Moreover, its extension using monadic fixpoints generalizes the two-way μ -calculus and also monadic Datalog, and is known to be decidable [tCS13]. To the best of our knowledge, UNFO extended with transitive closure or regular path expressions has not been studied yet, and we plan to investigate these extensions in the thesis.

A common generalization of both UNFO and GFO is the guarded negation fragment (GNFO) which restricts first-order logic by requiring that all occurrences of negation are guarded, where the guard is an atomic formula (possibly an equality statement) containing all the free variables of the negated formula. It is known that GNFO has the same desirable properties as modal and description logics, the unary negation fragment and the guarded fragment. In particular, its satisfiability problem is decidable and it has the tree-like model property as well as the finite model property. An extension of GNFO with a guarded fixpoint mechanism is also known to be decidable [BtCS15].

All first-order fragments discussed so far will be formally introduced in Chapter 2. In this thesis, our work can roughly be divided into two main topics. On the one hand, we study the complexity of conservative extensions, satisfiability and ontology-mediated queries in decidable fragments of first-order logic, such as the ones mentioned above. On the other hand, we are also interested in studying the expressive power of modal logics, and in particular, modal logics that can update the underlying structure. In the following sections we introduce the topics we study in this thesis as well as the problems we want to solve.

1.2.1 Complexity of Conservative Extensions

Conservative extensions are a fundamental notion in logic. A theory T_2 is said to be a *conservative extension* of a theory T_1 if the language of T_2 extends the language of T_1 such that any consequence of T_1 is also a consequence of T_2 ; and any consequence of T_2 , which uses only symbols from T_1 , is a consequence of T_1 as well. This notion plays an important role in mathematical logic and the foundations of mathematics as well as in computer science and artificial intelligence. In mathematical logic, they provide an important tool for relating logical theories such as theories of arithmetic. For example, the result that the Bernays-Gödel set theory BG (or BGC) is a conservative extension of the Zermelo-Fraenkel set theory ZF (or ZFC) means the relative consistency of BG(C): if ZF(C) is consistent then BG(C) is also consistent.¹ On the other hand, in computer science they come up in diverse areas such as software specification [DGS93], higher order theorem proving [GM93], and ontologies [KLWW09]. For example, conservative extensions can be used to define the notion of a module for ontologies: a subtheory is a module if the whole ontology is a conservative extension of the subtheory.

In this thesis we are interested in studying conservative extensions in computer science and artificial intelligence, and specifically in the area of ontologies. Ontologies are logical theories that specify a vocabulary for a domain of interest and describe the relationships between the terms in that vocabulary. Their main applications are in knowledge representation, in semantic databases, and in the semantic web. Description logics (DLs) play a key role as ontology languages, and there, ontologies are usually called TBoxes. An important reason for the success of DLs as ontology languages is the availability of a large class of reasoning services, along with implemented tool support that is integrated into ontology development systems. Traditionally, the most important reasoning services for DLs are consistency checking (check whether all terms in the specified vocabulary are free of contradictions) and classification (make explicit the is-a hierarchy between the terms of the vocabulary). These services are implemented based on the fundamental reasoning tasks satisfiability and subsumption. Over the last decade, though, it has become clear that these traditional reasoning services are not

¹Observe that a conservative extension of a consistent theory is also consistent. If it were not, then every consequence in the original theory as well as its negation would belong to the new theory, which then would not be a conservative extension.

enough. Given that ontologies from applications can reach considerable size, additional and more refined reasoning services are needed to support various aspects of ontology design and management.

A typical example of an advanced reasoning service is *ontology modularity and reuse*: when constructing a new ontology, it is often desirable to import a part of another, already existing ontology with the aim of covering selected thematic subdomains without modeling them from scratch. This raises the question whether a given subset of an existing ontology is self-contained regarding a vocabulary of interest, that is, whether the subset ‘says the same’ about the relevant vocabulary as the overall ontology. An appropriate way to formalize that a subset O' of an ontology O is self-contained regarding a vocabulary (set of relational symbols) Σ is to require that O is a conservative extension of O' regarding the symbols in Σ . To be more precise, we concentrate on *deductive conservative extensions*.

Definition 1.2.1 (Deductive Conservative Extensions). Let O_1 and O_2 be ontologies formulated in an ontology language L , and let Σ be a vocabulary. Then $O_1 \cup O_2$ is a deductive Σ -conservative extension of O_1 if for every sentence φ of L that uses only symbols from Σ , $O_1 \cup O_2 \models \varphi$ implies $O_1 \models \varphi$.

This definition gives rise to a decision problem: *Deciding deductive conservative extensions* means, given two ontologies O_1 and O_2 formulated in an ontology language L , and a vocabulary Σ , to decide whether $O_1 \cup O_2$ is a deductive conservative extension of O_1 with respect to Σ . As expected, conservative extensions are undecidable in first-order logic, but it has been observed in recent years that they are decidable in many modal and description logics [GLW06, GLWZ06, LWW07] and that they can often be characterized elegantly in terms of model theoretic notions [BKL⁺16]. Moreover, conservative extensions have become an essential technique in DL research, and tool support starts to become available, for example, in the form of module extractors.

In this thesis, we take the next step by studying conservative extensions in the more general context of decidable fragments of first-order logic, such as the two-variable fragment, the guarded fragment, and the guarded negation fragment. We believe that studying conservative extensions in more expressive fragments is an important endeavour from both a theoretical and practical perspective. From a theoretical point of view, the good computational behavior of DLs regarding (deductive) conservative extensions calls for a general explanation. We thus aim to study the following questions:

- Q1 Are conservative extensions decidable in relevant fragments of FO such as FO², GFO, and GNFO?
- Q2 What are the reasons for decidability of conservative extensions in modal and description logics and how far can the limits of decidability be pushed?

Observe that since ontologies are logical theories (sets of logical axioms) they can be translated into first-order logic, i.e., axioms correspond to universally quantified implications without free variables. In this context, instead of talking about ontologies we talk about sentences. Thus, given two sentences φ_1 and φ_2 , and Σ a vocabulary, we study whether $\varphi_1 \wedge \varphi_2$ is a (deductive) Σ -conservative extension of φ_1 . We will investigate questions Q1 and Q2 in Chapter 3.

On the other hand, from a practical perspective, there has been a very strong trend in recent years to use DL ontologies for accessing data. The general approach is known as *ontology-based data access* (OBDA) and is a very active field of research. Thus, the

recently very popular use of ontologies in OBDA and database-style applications requires for ontology languages not to preserve the logical consequences but to provide the same answers to queries. This gives rise to the notion of *query conservative extensions*. Given an ABox \mathcal{A} , an ontology O , a query $q(\mathbf{x})$, and a tuple of constants \mathbf{a} , we write $\mathcal{A} \cup O \models q(\mathbf{a})$ to say that all models of \mathcal{A} and O entail $q(\mathbf{a})$.

Definition 1.2.2 (Query Conservative Extensions). Let O_1 and O_2 be ontologies formulated in an ontology language L , and let Γ, Σ be vocabularies. Then $O_1 \cup O_2$ is a query (Γ, Σ) -conservative extension of O_1 if for all ABoxes \mathcal{A} that use only symbols from Γ , and conjunctive queries q that use only symbols from Σ , $\mathcal{A} \cup O_1 \cup O_2 \models q(\mathbf{a})$ implies $\mathcal{A} \cup O_1 \models q(\mathbf{a})$ for all tuples of constants \mathbf{a} .

For example, in OBDA applications it is a useful reasoning service to decide whether an ontology O that is used for query answering can safely be replaced by a smaller (and thus computationally more efficient) subset O' . In this case, deductive conservative extensions are too weak to formalize what we mean by ‘safe replacement’ while query conservative extensions are sufficient.

To make conservative extensions useful for OBDA applications, we also study the decidability and computational complexity of query conservative extensions in Horn description logics. The core feature of these logical languages is that they are incapable of expressing any form of disjunction. This lack of disjunction means that Horn DLs can be translated into the Horn fragment of FO. In first-order logic, Horn clauses are disjunctions of atomic formulas and negated atomic formulas that contain at most one non-negated atom. Using Horn DLs often leads to computational advantages. However, query conservative extensions in Horn DLs in the presence of inverse roles, often considered a crucial feature, have been poorly investigated. Therefore, in Chapter 4 we study the following question:

- Q3 How can advanced reasoning support for ontologies be lifted from standard DLs to Horn DLs involving inverse roles, thus facilitating ontology design and maintenance for OBDA applications?

From an OBDA perspective, it is interesting that both database instances and conjunctive queries can be expressed as formulas in the UNFO fragment. This makes it suitable to study not only query conservative extensions, but also other reasoning problems such as *ontology-mediated query answering*.

1.2.2 Complexity of Satisfiability and Ontology-mediated Queries

Another fundamental reasoning problem that we study in this thesis, other than conservative extensions, is that of *ontology-mediated query answering*. In recent years, there has been a recent trend both from academia and industry to investigate ontology-mediated queries (OMQ), in which the data is enriched by ontologies providing semantic and background knowledge to abstract the way in which the data is stored. More specifically, an *ontology-mediated query* is a triple (\mathcal{O}, Σ, q) where \mathcal{O} is a logical sentence called the *ontology*, Σ is a set of predicate symbols called the *data signature*, and q is a query.

Definition 1.2.3 (Ontology-mediated query answering). Let $Q = (\mathcal{O}, \Sigma, q)$ be an OMQ and D a database that uses only symbols from Σ . We call $\mathbf{a} \subseteq \text{dom}(D)$ a *certain answer to Q on D* if $\mathbf{a} \in \text{ans}(q, \mathfrak{A})$ for every structure \mathfrak{A} that extends D and is a model of \mathcal{O} , where \mathfrak{A} *extends* D if $\text{dom}(D) \subseteq \text{dom}(\mathfrak{A})$ and $P^D \subseteq P^{\mathfrak{A}}$ for all predicate symbols P . The set of all certain answers to Q on D is denoted $\text{cert}(Q, D)$.

To illustrate the idea, Figure 1.1 shows a typical ontology-mediated query scenario.

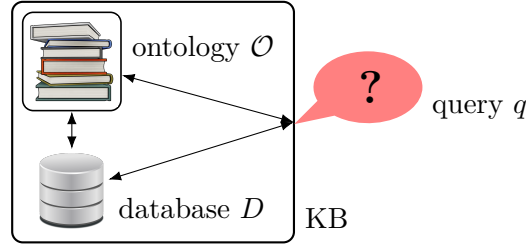


Figure 1.1: Ontology-mediated Query Answering

Typical decision problems of interest are *OMQ evaluation* and *OMQ containment*.

Definition 1.2.4 (OMQ evaluation). *OMQ evaluation* is the problem to decide, given an OMQ Q from a language L , a database D , and an $\mathbf{a} \subseteq \text{dom}(D)$, whether $\mathbf{a} \in \text{cert}(Q, D)$.

For ontology containment, we are interested in the natural special case where both ontologies are identical and the signature consists of all predicate symbols.

Definition 1.2.5 (OMQ containment). Let $Q_1 = (\mathcal{O}, \Sigma_{\text{full}}, q_1)$ and $Q_2 = (\mathcal{O}, \Sigma_{\text{full}}, q_2)$ be OMQs from a language L with the same number of answer variables and where Σ_{full} is the *full* data signature, that is, the set of all predicate symbols. We say that Q_1 is *contained in* Q_2 and write $Q_1 \subseteq Q_2$ if for every database D , $\text{cert}(Q_1, D) \subseteq \text{cert}(Q_2, D)$.

Description logics are among the most commonly used and well-studied ontology languages for ontology-mediated query answering. However, it can also be useful to consider more expressive decidable fragments of first-order logic as this serves to explore the limits of the ontology-mediated querying approach, to provide maximum expressive power for ontology formulation, and to put ontology-mediated querying into a more general logical perspective. Notably, this has been done in [BGO14, BtCS15, BtCLW14], where the guarded fragment, the unary negation fragment, and the guarded negation fragment of FO have been used as ontology languages. In particular, UNFO and GNFO, are attractive from the perspective of database theory because they can express conjunctive queries and ontologies formulated in many description logics. Then both fragments are relevant for ontology-mediated querying and, in fact, CQ evaluation under UNFO and GNFO ontologies (and thus also under DL ontologies) can be ‘expressed’ in UNFO and GNFO as a satisfiability problem. More precisely, given an OMQ $Q = (\mathcal{O}, \Sigma, q)$, where q a union of conjunctive queries and D is a Σ -database, then $D \models (\mathcal{O}, \Sigma, q)$ iff $\mathcal{O} \wedge D \wedge \neg q$ is unsatisfiable. Moreover, the containment of OMQs, as presented in Definition 1.2.5, can also be ‘expressed’ as a satisfiability problem. In this way, to study the complexity of OMQ evaluation and containment in UNFO and GNFO, it suffices to study the complexity of its satisfiability problem.

The UNFO fragment, in particular, was introduced in [tCS13] as well as its extension with fixpoint operators, and the complexity of satisfiability, finite satisfiability and model checking were investigated. However, besides fixpoints, there are no other extensions such as regular expressions or transitive closure operators, which are important features of many common description logics. Therefore, it would be interesting to extend UNFO with these features to study ontology-mediated query answering. We thus aim to study the following question:

Q4 What is the complexity of OMQ evaluation and OMQ containment when UNFO is extended with regular expressions and transitive relations?

A possible way to answer this question is by exploiting the relationship between satisfiability and ontology-mediated query answering. In fact, this is the approach we will follow in Chapter 5 where we study the complexity of satisfiability of UNFO extended with regular path expressions to obtain complexity results for OMQ evaluation and OMQ containment. This is the last chapter of the thesis where we obtain complexity results, and then move on to investigate the expressive power of a particular family of modal logics.

1.2.3 Investigating Expressive Power

Besides computational complexity results, in this thesis we are also interested in investigating the expressive power of logics. In general, one can think of expressive power as follows. A formula of a language can express an abstract property of models in which the formula is true. The more properties expressible with the language, the more expressive power the language has. But which properties can be expressed with which languages, and which cannot be expressed? We are also interested in the question of how languages are related. Given two languages, Is one more expressive than the other?

As we have discussed in Section 1.2, modal and description logics can be seen as fragments of FO^2 by providing a satisfiability-preserving translation. This illustrates one of the important strategies one can employ to show that two languages are equally expressive: provide a translation from one language to the other that provides an equivalent formula in the other language for each formula in the first language, and vice versa. We thus can compare their expressive power using appropriate notions of equivalence. In the case of modal logic, for example, *bisimulations* are the appropriate notions. As we will see in later chapters, if two models are bisimilar, then they satisfy the same formulas. Then to prove that one language is more expressive than another, we have to show that there are some properties of the more expressive language that cannot be expressed in the least expressive language, i.e., there are no equivalent formulas. In fact, we have to show that a formula in the more expressive language is not equivalent to any other formula in the least expressive language, but there are infinitely many formulas. Here is where the notion of bisimulation comes into play: if we can show that there is a formula that can distinguish between two bisimilar elements, then the formulas that are true on these elements cannot be equivalent as bisimilar models satisfy the same formulas. Therefore, we will define various notions of bisimulations thorough the thesis, tailored to the specific logics at hand. We will also mention expressivity results in some chapters of the thesis, providing examples and referring to the literature when necessary. However, most of the technical results involving expressive power will be concerned with modal logics.

In Chapter 6, we move from ontology languages to modal languages and investigate a family of modal logics named *Relation-Changing Modal Logics (RCMLs)*. RCMLs are extensions of the basic modal logic with dynamic operators that modify the accessibility relation of a model during the evaluation of a formula. These languages are equipped with dynamic modalities that are able, for example, to delete, add, and swap edges in the model, both locally and globally. RCMLs were first introduced in [AFH12] and further studied in [Fer14]. Although the logics are abstract in spirit, they are motivated by applications in dynamic epistemic logic [vDvdHK07] and sabotage modal logic [vB05].

In [AFHM17, Mar15] we showed that RCMLs, as presented in this thesis, are undecidable. These results rule out the logics to be useful in practical applications where modeling dynamic scenarios is needed. Thus, it would be interesting to identify decidable fragments. By taking a closer look at the semantics of RCMLs one can see that they are related to hybrid logics. Since it is known that there are decidable fragments for hybrid logics, it would be interesting to investigate if RCMLs can be translated into hybrid logics in order to benefit from their decidable fragments. By proving syntactic translations we would also be able to compare the expressive power between these two families of logics and get a better understanding of their relationship. We thus study the following questions:

Q5 Is it possible to provide translations of RCMLs to hybrid logics in order to obtain decidable fragments?

Q6 Are RCMLs as expressive as hybrid logics? How are they related?

It seems that the ability to name states in the model and refer to those states by their name, is a crucial feature of hybrid logics to simulate changes in the accessibility relation of a model. We will exploit these characteristics with the aim to answer the questions Q5 and Q6 above.

Chapter 6 differs from the previous chapters in the sense that we don't study ontology languages and don't obtain complexity results. However, all languages investigated in this thesis are fragments of first-order logic and it would be useful to understand how they are related to each other.

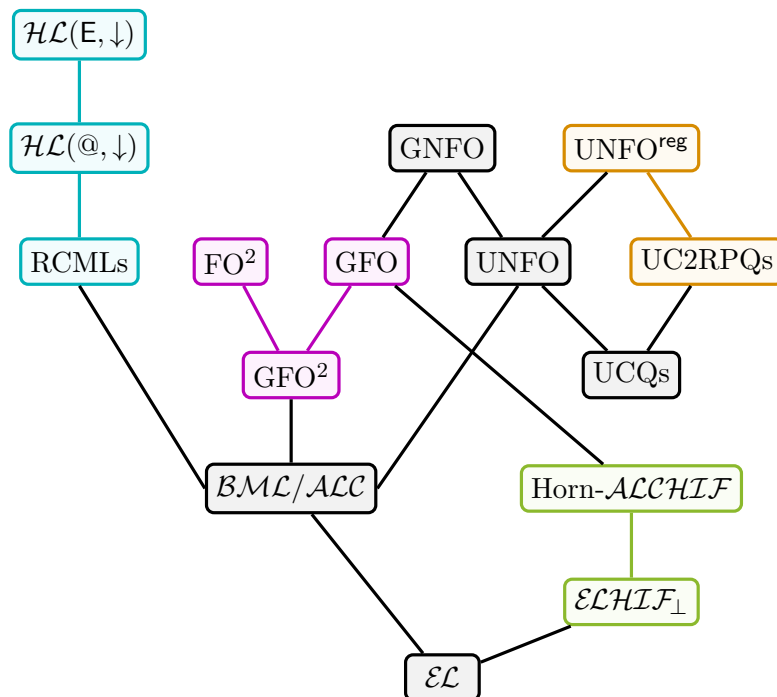


Figure 1.2: Comparison of expressive power.

- Logics studied in Chapter 3.
- Logics studied in Chapter 4.
- Logics studied in Chapter 5.
- Logics studied in Chapter 6.

In Figure 1.2, we present a global overview of the logics relevant for the thesis by comparing their expressive power. When a line is drawn between two languages \mathcal{L}' and \mathcal{L} , where \mathcal{L}' is above \mathcal{L} , it means that \mathcal{L}' is *at least as expressive* as \mathcal{L} , denoted $\mathcal{L} \leq \mathcal{L}'$; if there is no line between them, it means they are *incomparable* in terms of expressive power. The \leq relation indicates that we can embed one language into another via a translation from the first language to the second one. If they are incomparable, then none of the two languages can be embedded in the other, i.e., they are able to say different things (see Definition 6.5.2). The colors in Figure 1.2 help to identify the languages we investigate in this thesis.

Precise definitions of basic formalisms such as \mathcal{EL} , \mathcal{ALC} , \mathcal{BML} , FO^2 , GFO, UNFO, GNFO, CQs and UCQs will be provided in Chapter 2. The rest of the logics will be presented in their respective chapters. In particular, in Chapter 3 we study deductive conservative extensions in FO^2 and GFO, and in their intersection GFO^2 . In Chapter 4, we investigate query conservative extensions in Horn- \mathcal{ALCHIF} and deductive conservative extensions in \mathcal{ELHIF}_\perp . Chapter 5 is dedicated to the study of satisfiability and ontology-mediated queries in UNFO^{reg} , an extension of UNFO with regular path expressions which can express UC2RPQs. Finally, in Chapter 6 we provide translations of RCMLs to the hybrid logics $\mathcal{HL}(@, \downarrow)$ and $\mathcal{HL}(E, \downarrow)$. We give more details of the contents of the thesis in the next section.

1.3 Overview of the Thesis

The thesis can be divided into two main parts. The first part, Chapters 3 and 4, deals with conservative extensions in the two-variable fragment and the guarded fragment as well as in Horn description logics with inverse roles. The second part, Chapters 5 and 6, deals with the complexity of satisfiability and expressivity in modal-like logics. In particular, in Chapter 5 we study the computational complexity of OMQ evaluation via satisfiability in UNFO (which can be seen as a modal logic with conjunctive queries as modal operators) extended with regular paths expressions on binary relations. In Chapter 6, we study modal logics that can update the accessibility relation of a model during the evaluation of a formula and provide translations into hybrid logic. More precisely, the thesis is structured as follows.

In Chapter 2 we introduce first-order logic and some of its decidable fragments, namely the two-variable fragment, the guarded fragment, the unary negation fragment, and the guarded negation fragment. We also present the basic description logics \mathcal{ALC} and \mathcal{EL} as well as the basic modal logic \mathcal{BML} . In particular, we cover syntax and semantics of these logics, we present basic reasoning problems and discuss the relationship between description logic, modal logic, and first-order logic. The preliminary chapter is rather short, but serves as a foundation to understand the concepts and definitions in the following chapters.

In Chapter 3 we study the decidability and computational complexity of deductive conservative extensions in expressive fragments of FO, such as the two-variable fragment and the guarded fragment. We show that conservative extensions are undecidable in FO^2 and in GF and that they are decidable and 2EXPTIME -complete in the intersection GF^2 of FO^2 and GF. The undecidability proofs are by reductions from the halting problem for two-register machines in the case of GF and from a tiling problem in the case of FO^2 . For the upper bound of GF^2 we rely on a model-theoretic characterization based on a mixture of bounded and unbounded guarded bisimulations, and then we use it as a basis for a decision procedure based on (alternating) tree automata. This

chapter is based on [JLM⁺17].

In Chapter 4 we study the decidability and computational complexity of query conservative extensions in Horn description logics with inverse roles. We prove that the problem is 2EXPTIME-complete in Horn- \mathcal{ALCHIF} (and also in Horn- \mathcal{ALC} and in \mathcal{ELI}). Moreover, we obtain the same upper bound for deductive conservative extensions, for which we also prove a CONEXPTIME lower bound. To prove the upper bound in Horn- \mathcal{ALCHIF} , we provide a model-theoretic characterization in terms of a mixture of bounded and unbounded homomorphisms, and then we resort to a combination of automata and mosaic techniques to implement it. This chapter is based on [JLMS17].

In Chapter 5 we consider the natural extension of UNFO with regular expressions on binary relations. The resulting logic UNFO^{reg} can express (unions of) conjunctive two-way regular path queries (C2RPQs) and ontologies formulated in DLs that include transitive roles and regular expressions on roles. Our main results are that OMQ evaluation under UNFO^{reg} ontologies is decidable, 2EXPTIME-complete in combined complexity, and CONP-complete in data complexity, and that satisfiability in UNFO^{reg} is 2EXPTIME-complete, thus not harder than in UNFO. We additionally show that the complexity of model checking in UNFO^{reg} is the same as in UNFO, namely complete for $P^{NP[O(\log^2 n)]}$. This chapter is based on [JLMS18].

In Chapter 6 we study relation-changing modal logics, a family of modal logics that allow changes to the accessibility relation of a model during the evaluation of a formula. In particular, they are equipped with dynamic modalities that are able to delete, add, and swap edges in the model, both locally and globally. We provide translations from these logics into hybrid logic, and while RCMLs are known to be undecidable [AFHM17], we use our translations to identify decidable fragments. Additionally, we also compare the expressive power of RCMLs with hybrid logics. This chapter is based on [AFHM16, AFHM18].

Finally, we conclude in Chapter 7 giving a brief summary of the thesis.

Summary of Publications

- [JLM⁺17] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, Thomas Schneider, and Frank Wolter. Conservative Extensions in Guarded and Two-Variable Fragments. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 108:1–108:14, 2017
- [JLMS17] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, and Thomas Schneider. Query Conservative Extensions in Horn Description Logics with Inverse Roles. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1116–1122, 2017
- [JLMS18] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, and Thomas Schneider. Querying the Unary Negation Fragment with Regular Path Expressions. In *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, pages 15:1–15:18, 2018
- [AFHM18] Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Satisfiability for Relation-Changing Logics. *To appear in Journal of Logic and Computation*, 2018

-
- [AFHM17] Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Undecidability of Relation-Changing Modal Logics. In *Dynamic Logic. New Trends and Applications - First International Workshop, DALI 2017, Brasilia, Brazil, September 23-24, 2017, Proceedings*, pages 1–16, 2017
- [AFHM16] Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Relation-Changing Logics as Fragments of Hybrid Logics. In *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016*, pages 16–29, 2016

This chapter briefly introduces the basic formalisms we investigate in the thesis. We start with first-order logic and then continue with relevant decidable fragments, such as two-variable, guarded, and unary negation fragments. We then introduce the basic description logics \mathcal{ALC} and \mathcal{EL} , present their syntax and semantics as well as basic reasoning problems, and view them as fragments of first-order logic. Finally, we formally introduce the syntax and semantics of the basic modal logic together with basic reasoning problems, and state its relationship with description logic and first-order logic.

2.1 First-Order Logic

We introduce the syntax of first-order logic (FO) [End01] based on a signature Σ , a set containing constant and predicate symbols, each predicate symbol with an arity. First-order formulas over a signature Σ are built according to the following syntax rule:

$$\varphi, \psi ::= \top \mid t_1 = t_2 \mid R(t_1, \dots, t_k) \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x\varphi(x)$$

where $R \in \Sigma$ is a k -ary predicate symbol and each t_i is either a constant symbol from Σ or a variable (taken from a countably infinite supply of variable symbols). As usual, we take \vee , \rightarrow , \leftrightarrow , and \forall as defined symbols. Note that we admit equality but do not allow for function symbols except for constants. The semantics of FO is given in terms of *relational structures* $\mathfrak{A} = (A, \pi)$, where A is the *domain* of \mathfrak{A} and π is an *interpretation function* assigning to each k -ary predicate symbol R a subset $\pi(R) \subseteq A^k$ and to each constant symbol c a domain element $\pi(c) \in A$. A *valuation* for A is a function ν from the set of variables to the domain. The *satisfaction relation* \models is now defined by induction on the structure of formulas:

$(\mathfrak{A}, \nu) \models \top$	always
$(\mathfrak{A}, \nu) \models t_1 = t_2$	iff $a_1 = a_2$, where a_i is $\nu(t_i)$ if t_i is a variable and $\pi(t_i)$ otherwise;
$(\mathfrak{A}, \nu) \models R(t_1, \dots, t_k)$	iff $(a_1, \dots, a_k) \in \pi(R)$, where a_i is $\nu(t_i)$ if t_i is a variable and $\pi(t_i)$ otherwise;
$(\mathfrak{A}, \nu) \models \neg\varphi$	iff not $(\mathfrak{A}, \nu) \models \varphi$;
$(\mathfrak{A}, \nu) \models \varphi \wedge \psi$	iff $(\mathfrak{A}, \nu) \models \varphi$ and $(\mathfrak{A}, \nu) \models \psi$;
$(\mathfrak{A}, \nu) \models \exists x.\varphi(x)$	iff there is $a \in A$ with $(\mathfrak{A}, \nu[x/a]) \models \varphi(x)$, where $\nu[x/a]$ replaces every occurrence of x with a .

We indicate with $\varphi(\mathbf{x})$ that φ might have free variables among x and call formulas without free variables sentences. We say that a formula $\varphi(\mathbf{x})$ is *satisfiable* if there is a structure \mathfrak{A} and a valuation ν such that $(\mathfrak{A}, \nu) \models \varphi(\mathbf{x})$. For sentences φ , we drop the valuation and just write $\mathfrak{A} \models \varphi$. A sentence φ is *valid* if $\neg\varphi$ is not satisfiable.

We are now ready to introduce the fragments of FO that will be studied in the thesis.

2.2 Two-Variable, Guarded, and Unary Negation Fragments

The *two-variable fragment of FO*, denoted FO^2 , is obtained by fixing two variables x and y and disallowing the use of other variables [Sco62, Mor75].

For the purpose of introducing the next fragments of FO, we assume a countably infinite supply of predicate symbols of any arity. We then define the grammar rules that define the syntax of every fragment, and in all cases the semantics is the same as FO.

The *guarded fragment of FO*, denoted GFO, imposes that all quantifiers must be relativized by atomic formulas. Formulas φ in GFO are formed according to the following grammar:

$$\varphi ::= P(\mathbf{x}) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x(\alpha(\mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{x}, \mathbf{y}))$$

where P ranges over predicate symbols and $\varphi(\mathbf{x}, \mathbf{y})$ is a GFO formula with free variables among \mathbf{x}, \mathbf{y} and $\alpha(\mathbf{x}, \mathbf{y})$ is an atomic formula or an equality $x = y$ that in either case contains all variables in \mathbf{x}, \mathbf{y} . The formula α is called the *guard* of the quantifier [ANv98, Grä99]. GFO restricts the use of quantification but allows unrestricted use of negation, in contrast with the unary negation fragment.

The *unary negation fragment of FO*, denoted UNFO, restricts the negation of formulas to having only one free variable. Formulas φ in UNFO are formed according to the following grammar:

$$\varphi ::= P(\mathbf{x}) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x\varphi \mid \neg\varphi(x)$$

where P ranges over predicate symbols and in the $\neg\varphi(x)$ clause, φ has no free variables besides (possibly) x [tCS13]. In particular, $x \neq y$ is not expressible in UNFO.

The *guarded negation fragment of FO*, denoted GNFO, is a generalization of both GFO and UNFO. Formulas φ in GNFO are formed according to the following grammar:

$$\varphi ::= P(\mathbf{x}) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x\varphi \mid \alpha(\mathbf{x}, \mathbf{y}) \wedge \neg\varphi(\mathbf{x})$$

where P ranges over predicate symbols and the “guard” α is an atomic formula (possibly an equality statement) containing all the free variables of φ [BtCS15].

Observe that both UNFO and GNFO allow unrestricted use of existential quantification, and therefore, include union of conjunctive queries and monadic Datalog.

Let us now move on to description and modal logics, two related families of logics that are also fragments of FO.

2.3 Description Logic

We use standard notation for the syntax and semantics of description logic [BHLS17, BCM⁺07]. Let $\mathsf{N}_C, \mathsf{N}_R, \mathsf{N}_I$ denote countably infinite sets of concept names, role names, and individual names, respectively. We introduce the basic description logic \mathcal{ALC} . \mathcal{ALC} -concepts are formed according to the following syntax rule:

$$C, D ::= \top \mid A \mid \neg C \mid C \sqcap D \mid \exists r.C$$

where A ranges over N_C and R over N_R . We use the abbreviations $\forall r.C$ for $\neg \exists r. \neg C$, $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$, and \perp for $\neg \top$. The set of \mathcal{EL} -concepts is defined by dropping negation from the syntax rule of \mathcal{ALC} -concepts.

Description logic knowledge bases are typically separated in *terminological knowledge* and *assertional knowledge*. The former is represented in the TBox while the latter, the data, is represented in an ABox. For $L \in \{\mathcal{ALC}, \mathcal{EL}\}$, a *general L-TBox* is a set T of *concept inclusions* $C \sqsubseteq D$ with C, D L -concepts. A *classical L-TBox* is a set \mathcal{T} of *concept definitions* $A \equiv C$ such that each concept name $A \in \mathsf{N}_C$ occurs at most once in the left-hand side of a concept definition in \mathcal{T} and C is an L -concept. We will drop the reference to the TBox language when no confusion arises. Note that a classical TBox is a special case of a general TBox since we can replace $A \equiv C$ by the two concept inclusions $A \sqsubseteq C, C \sqsubseteq A$. An *ABox* \mathcal{A} is a non-empty set of *concept and role assertions* of the form $A(a)$ and $r(a, b)$, where $A \in \mathsf{N}_C, r \in \mathsf{N}_R$ and $a, b \in \mathsf{N}_I$. We write $\text{ind}(\mathcal{A})$ for the set of individuals in \mathcal{A} . A DL knowledge base is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

The semantics of DLs is given via interpretations. An *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of individuals, the *domain*, and $\cdot^{\mathcal{I}}$ is an *interpretation function* mapping each $a \in \mathsf{N}_I$ to some domain element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept name $A \in \mathsf{N}_C$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of the domain and each role name $r \in \mathsf{N}_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ over the domain.

We assume the *unique name assumption*, that is, we assume that different individuals are interpreted by different domain elements. The interpretation function is defined for \mathcal{ALC} -concepts as follows:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}}; \\ \neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}; \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}; \\ \exists R.C &= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} : (d, e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}. \end{aligned}$$

An interpretation \mathcal{I} *satisfies* or *is a model of* a concept C if $C^{\mathcal{I}} = \emptyset$; a concept inclusion $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; a concept definition $A \equiv C$, written $\mathcal{I} \models A \equiv C$, if $A^{\mathcal{I}} = C^{\mathcal{I}}$; a general TBox \mathcal{T} , written $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I} \models C \sqsubseteq D$ for all $C \sqsubseteq D \in \mathcal{T}$; an ABox \mathcal{A} , written $\mathcal{I} \models \mathcal{A}$, if for each assertion $A(a) \in \mathcal{A}$, we have $a \in A^{\mathcal{I}}$ and for each assertion $r(a, b) \in \mathcal{A}$, we have $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$.

Traditional reasoning problems for DLs are *concept satisfiability*, *knowledge base consistency*, and *subsumption*. We say that a concept C is *satisfiable relative to a TBox*

\mathcal{T} if there is a common model of C and \mathcal{T} . A concept C is *subsumed by D relative to a TBox \mathcal{T}* , written $\mathcal{T} \models C \sqsubseteq D$, when for all models \mathcal{I} of \mathcal{T} we have $\mathcal{I} \models C \sqsubseteq D$. A knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is *consistent* if there is a common model of \mathcal{T} and \mathcal{A} .

Another important reasoning problem in DLs is that of *conjunctive query evaluation*.

Definition 2.3.1 (Conjunctive queries). A *conjunctive query (CQ)* is of the form $q(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$, where \mathbf{x} and \mathbf{y} are tuples of variables and $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of *atoms* of the form $A(v)$ or $r(v, w)$ with $A \in \mathbf{N}_{\mathcal{C}}$, $r \in \mathbf{N}_{\mathcal{R}}$, and $v, w \in \mathbf{x} \cup \mathbf{y}$. We call \mathbf{x} *answer variables* and \mathbf{y} *quantified variables* of q . A *union of conjunctive queries (UCQ)* is a disjunction of CQs where all of them have the same answer variables.

Definition 2.3.2 (Semantics of CQs). A *match* of q in an interpretation \mathcal{I} is a function $\pi : \mathbf{x} \cup \mathbf{y} \rightarrow \Delta^{\mathcal{I}}$ such that $\pi(v) \in A^{\mathcal{I}}$ for every atom $A(v)$ of q and $(\pi(v), \pi(w)) \in r^{\mathcal{I}}$ for every atom $r(v, w)$ of q . We write $\mathcal{I} \models q(a_1, \dots, a_n)$ if there is a match of q in \mathcal{I} with $\pi(x_i) = a_i$ for all $i < n$. A tuple \mathbf{a} of elements from $\mathbf{N}_{\mathcal{I}}$ is a *certain answer* to q over an ABox \mathcal{A} given a TBox \mathcal{T} , written $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$, if $\mathcal{I} \models q(\mathbf{a})$ for all models of \mathcal{T} and \mathcal{A} .

Definition 2.3.3 (CQ evaluation). *CQ evaluation* is the problem to decide, given a TBox \mathcal{T} , an ABox \mathcal{A} , a CQ q , and a tuple $\mathbf{a} \in \text{ind}(\mathcal{A})$, whether $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$.

As we have discussed in Section 1.2, it is possible to define an equivalence-preserving translation of modal and description logic formulas into first-order formulas with one free variable. In particular, the description logic \mathcal{ALC} can be seen as a fragment of first-order logic by providing two translations functions, π_x and π_y , that inductively map \mathcal{ALC} concepts into first-order formulas with one free variable, x or y :

$$\begin{array}{l|l} \pi_x(A) = A(x) & \pi_y(A) = A(y) \\ \pi_x(\neg C) = \neg \pi_x(C) & \pi_y(\neg C) = \neg \pi_y(C) \\ \pi_x(C \sqcap D) = \pi_x(C) \sqcap \pi_x(D) & \pi_y(C \sqcap D) = \pi_y(C) \sqcap \pi_y(D) \\ \pi_x(\exists r.C) = \exists y(r(x, y) \wedge \pi_y(C)) & \pi_y(\exists r.C) = \exists x(r(y, x) \wedge \pi_x(C)) \end{array}$$

For knowledge bases, observe that we can translate TBox axioms as universally quantified (bi)-implications without free variables, and ABox assertions as ground facts. We translate a TBox \mathcal{T} and an ABox \mathcal{A} as follows, where $\varphi[x \mapsto a]$ is the formula obtained from φ by replacing all free occurrences of x with a :

$$\begin{aligned} \pi(\mathcal{T}) &= \forall x. \bigwedge_{C \sqsubseteq D \in \mathcal{T}} (\pi_x(C) \Rightarrow \pi_x(D)), \\ \pi(\mathcal{A}) &= \bigwedge_{C(a) \in \mathcal{A}} \pi_x(C)[x \mapsto a] \wedge \bigwedge_{r(a, b) \in \mathcal{A}} r(a, b). \end{aligned}$$

With this translation we can view DL interpretations as first-order interpretations and vice versa. This not only provides an alternative way of defining the semantics of \mathcal{ALC} , but also tell us that \mathcal{ALC} is a decidable fragment of first-order logic: the translation uses only variables x and y , and thus yields a formula in FO^2 . Similarly, the translation uses guarded quantification, and thus yields a formula in GFO.

Description logics can be seen as cousins of modal logics, yet they have been developed independently. We introduce next the basic modal logic and show its relationship with description logic.

2.4 Modal Logic

We formally introduce the basic modal logic, denoted \mathcal{BML} , and refer to the standard literature for more details [BdRV01, BvBW06].

Definition 2.4.1 (Syntax). Let PROP be a countable, infinite set of propositional symbols. Then the set FORM of *formulas* of \mathcal{ML} over PROP is defined as:

$$\text{FORM} ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \diamond\varphi,$$

where $p \in \text{PROP}$ and $\varphi, \psi \in \text{FORM}$. $\Box\varphi$ is a shorthand for $\neg\diamond\neg\varphi$. Other operators are defined as usual.

The semantics is defined in terms of *Kripke models*.

Definition 2.4.2 (Kripke Models). A *Kripke model* \mathcal{M} is a triple $\mathcal{M} = \langle W, R, V \rangle$, where W is a non-empty set whose elements are called *points* or *states*; $R \subseteq W \times W$ is the *accessibility relation*; and $V : \text{PROP} \rightarrow \mathcal{P}(W)$ is a *valuation*. Informally we think of $V(p)$ as the set of points in \mathcal{M} where p is true.

Modal logics describe Kripke structures from an internal perspective. This means that modal formulas are evaluated at some particular point of the model. For this purpose we use *pointed models*: pairs of the form (\mathcal{M}, w) , where w is a state in \mathcal{M} ; we usually drop parentheses and call \mathcal{M}, w a pointed model.

Definition 2.4.3 (Semantics). Given a pointed model \mathcal{M}, w and a formula φ we say that \mathcal{M}, w *satisfies* φ (notation, $\mathcal{M}, w \models \varphi$) when

$$\begin{array}{ll} \mathcal{M}, w \models \top & \text{always} \\ \mathcal{M}, w \models p & \text{iff } w \in V(p) \\ \mathcal{M}, w \models \neg\varphi & \text{iff } \mathcal{M}, w \not\models \varphi \\ \mathcal{M}, w \models \varphi \wedge \psi & \text{iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\ \mathcal{M}, w \models \diamond\varphi & \text{iff for some } v \in W \text{ s.t. } (w, v) \in R, \mathcal{M}, v \models \varphi. \end{array}$$

A formula φ is *satisfiable* if there exists a pointed model \mathcal{M}, w such that $\mathcal{M}, w \models \varphi$. A formula φ is *globally satisfiable* in a model \mathcal{M} if it is satisfied at all points in \mathcal{M} , and if this is the case we write $\mathcal{M} \models \varphi$. A formula φ is *valid* if it is globally satisfied in all models, and if this is the case we write $\models \varphi$.

As we mentioned before, modal logics are closely related to description logics. It is not hard to see that \mathcal{ALC} -concepts can be viewed as syntactic variants of formulas of multimodal \mathcal{BML} . Kripke structures can easily be viewed as DL interpretations, and vice versa; we can then view concept names as propositional variables, and role names as modal parameters. We formalize this relationship through the mapping π from multimodal \mathcal{BML} to \mathcal{ALC} as follows:

$$\begin{array}{ll} \pi(p) & = A \text{ such that } p \in A, \text{ for } p \in \text{PROP and } A \in \mathbf{N}_C \\ \pi(\neg\varphi) & = \neg\pi(\varphi) \\ \pi(\varphi \wedge \psi) & = \pi(\varphi) \sqcap \pi(\psi) \\ \pi(\langle r \rangle\varphi) & = \exists r.\pi(\varphi) \end{array}$$

With this translation, modal logics can also be seen as fragments of first-order logic and, in particular, as fragments of FO^2 and GFO.

The translation of DL knowledge bases is not immediate: a TBox \mathcal{T} is satisfied only in those structures where, for each $C \sqsubseteq D$, $\neg\pi(C) \vee \pi(D)$ holds *globally*, i.e., in each point of the Kripke structure. This can be expressed in modal logic using the *universal modality*. Like TBoxes, ABoxes do not have a direct correspondence in modal logic, but they can be seen as a special case of *nominals*. The universal modality and nominals will be introduced in Section 6.2.

 Conservative Extensions in Guarded and Two-Variable Fragments

In the area of description logic, deciding whether a logical theory is a conservative extension of another theory is a fundamental reasoning task with applications in ontology modularity and reuse, ontology versioning, and ontology summarization [BKL⁺16]. In these applications, it would be very useful to decide, given two sentences φ_1 and φ_2 , whether $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1 . As expected, this problem is undecidable in first-order logic (FO). In contrast, for many modal and description logics, conservative extensions are often decidable [GLW06, GLWZ06, LWW07]. The main approach to proving decidability of conservative extensions is to first establish a suitable model-theoretic characterization based on bisimulations, simulations, or homomorphisms, which is then used as a foundation for a decision procedure based on tree automata [LW11, BKL⁺16].

Regarding decidability, conservative extensions thus seem to behave similarly to the classical satisfiability problem, which is also undecidable in FO while it is decidable for modal and description logics. In the case of satisfiability, the aim to understand the deeper reasons for this discrepancy and to push the limits of decidability to more expressive fragments of FO has sparked a long line of research that identified prominent decidable FO fragments such as the two-variable fragment FO^2 [Sco62, Mor75], the guarded fragment GFO [ANv98, Grä99], and the guarded negation fragment GNFO [BtCS15]. These fragments have sometimes been used as a replacement for the modal and description logics that they generalize, and in particular the guarded fragment has been proposed as an ontology language [BGO14]. Motivated by this situation, and given that conservative extensions are now well understood in many modal and description logics, we take the next step by studying the decidability and computational complexity of conservative extensions in more expressive decidable fragments of FO such as FO^2 , GFO, and GNFO. In particular, we study the first two questions of Section 1.2.1:

- Q1 Are conservative extensions decidable in relevant fragments of FO such as FO^2 , GFO, and GNFO?
- Q2 What are the reasons for decidability of conservative extensions in modal and description logics and how far can the limits of decidability be pushed?

In this chapter, we show that conservative extensions are undecidable in any fragment

of FO that contains FO² or GFO (even the three-variable fragment thereof), and that they are decidable and 2EXPTIME-complete in the two-variable guarded fragment GFO², which lies in the intersection of FO² and GFO.

To be more precise, we concentrate on *deductive* conservative extensions, that is, $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1 if for every sentence ψ formulated in the signature of φ_1 , $\varphi_1 \wedge \varphi_2 \models \psi$ implies $\varphi_1 \models \psi$. There is also a *model-theoretic* notion of conservative extension which says that $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1 if every model of φ_1 can be extended to a model of φ_2 by interpreting the additional symbols in φ_2 . Model-theoretic conservative extensions imply deductive conservative extensions, but the converse fails unless one works with a very expressive logic such as second-order logic [KLWW09]. Deductive conservative extensions are also closely related to other important notions in logic, such as uniform interpolation [Vis96, BtCV15]. For example, in logics that enjoy Craig interpolation, a decision procedure for conservative extensions can also be used to decide whether a given sentence φ_2 is a uniform interpolant of a given sentence φ_1 regarding the symbols used in φ_2 .

In applications, it is often useful to consider a vocabulary Σ of interest. Instead of concentrating only on conservative extensions we also consider two related reasoning problems that we call Σ -entailment and Σ -inseparability, where Σ denotes a signature. The definitions are as follows: a sentence φ_1 Σ -entails a sentence φ_2 if for every sentence ψ formulated in Σ , $\varphi_2 \models \psi$ implies $\varphi_1 \models \psi$. This can be viewed as a more relaxed notion of conservative extension where it is not required that one sentence actually extends the other as in the conjunction $\varphi_1 \wedge \varphi_2$ used in the definition of conservative extensions. Two sentences φ_1, φ_2 are Σ -inseparable if they Σ -entail each other. Note that conservative extensions and Σ -inseparability reduce in polynomial time to Σ -entailment (with two calls to Σ -entailment required in the case of Σ -inseparability), and moreover, conservative extensions reduce in polynomial time to Σ -inseparability. Therefore, we obtain the same decidability and complexity results for all three problems.

We start by showing that conservative extensions are undecidable in FO² and (the three-variable fragment of) GFO, and in fact in all fragments of FO that contain at least one of the two; note that the latter is not immediate because the separating sentence ψ in the definition of conservative extensions ranges over all sentences from the considered fragment, giving greater separating power when we move to a larger fragment. The proofs are by reductions from the halting problem for two-register machines and a tiling problem, respectively. We note that undecidability of conservative extensions also implies that there is no extension of the logic in question in which consequence is decidable and that has effective uniform interpolation (in the sense that uniform interpolants exist and are computable). We then we show as our second main result that, in the two-variable guarded fragment GFO², Σ -entailment is decidable and 2EXPTIME-complete. Regarding the satisfiability problem, GFO² behaves fairly similarly to modal and description logics. It is thus surprising that deciding Σ -entailment in GFO² turns out to be much more challenging. However, regarding conservative extensions, the complexity is not harder than in most modal and description logics. We observe that a 2EXPTIME lower bound from [GLW06] for conservative extensions in description logics can be adapted to GFO². For the upper bound, we use an automata-based technique as for modal and description logics, but the model-theoretic characterizations that we present are much more complex. In GFO², a model-theoretic characterization in terms of an appropriate notion of (guarded) bisimulation fails. Instead, one has to demand the existence of *k-bounded* guarded bisimulations, *for all k*, and while tree automata can easily handle bisimulations, it is not clear how they can deal with such an infinite

family of bounded bisimulations. We solve this problem by a very careful analysis of the situation and by providing another characterization that can be viewed as being ‘half way’ between a model-theoretic characterization and an automata-encoding of Σ -entailment. Since it is known that GFO^2 enjoys Craig interpolation, our results are also relevant to deciding uniform interpolants and to a stronger version of conservative extensions in which the separating sentence ψ can also use ‘helper symbols’ that occur neither in φ_1 nor in φ_2 .

3.1 Deductive Conservative Extensions

We consider the two-variable fragment (FO^2) and the guarded fragment (GFO) of first-order logic (FO) as introduced in Chapter 2. We generally admit equality and disallow function symbols and constants. In FO^2 and fragments thereof, we generally admit only predicates of arity one and two, which is without loss of generality [GKV97]. The k -variable fragment of GFO, denoted GFO^k , is the set of GFO formulas involving no more than k variables.

A *signature* Σ is a finite set of predicates. We use $\text{GFO}(\Sigma)$ to denote the set of all GFO-sentences that use only predicates from Σ (and possibly equality), and likewise for $\text{GFO}^2(\Sigma)$ and other fragments. We use $\text{sig}(\varphi)$ to denote the set of predicates that occur in the FO formula φ . Note that we consider equality to be a logical symbol, rather than a predicate, and it is thus never part of a signature. We write $\varphi_1 \models \varphi_2$ if φ_2 is a logical consequence of φ_1 . The next definition introduces the central notions of this chapter.

Definition 3.1.1. Let F be a fragment of FO, φ_1, φ_2 F -sentences and Σ a signature. Then

1. φ_1 Σ -entails φ_2 , written $\varphi_1 \models_{\Sigma} \varphi_2$, if for all $F(\Sigma)$ -sentences ψ , $\varphi_2 \models \psi$ implies $\varphi_1 \models \psi$;
2. φ_1 and φ_2 are Σ -inseparable if $\varphi_1 \models_{\Sigma} \varphi_2$ and $\varphi_2 \models_{\Sigma} \varphi_1$;
3. $\varphi_1 \wedge \varphi_2$ is a *conservative extension* of φ_1 if $\varphi_1 \text{ sig}(\varphi_1)$ -entails $\varphi_1 \wedge \varphi_2$.

Note that logical consequence, $\varphi_1 \models \varphi_2$, can be expressed as a validity or (un)satisfiability problem: φ_1 implies φ_2 is valid, or $\varphi_1 \wedge \neg\varphi_2$ is unsatisfiable. Then Σ -entailment could equivalently be defined as follows when F is closed under negation: $\varphi_1 \models_{\Sigma} \varphi_2$ if for all $F(\Sigma)$ -sentences ψ , satisfiability of $\varphi_1 \wedge \psi$ implies satisfiability of $\varphi_2 \wedge \psi$. In other words, if φ_1 does not Σ -entail φ_2 , there is thus an $F(\Sigma)$ -sentence ψ such that $\varphi_1 \wedge \psi$ is satisfiable while $\varphi_2 \wedge \psi$ is not. We refer to such ψ as a *witness sentence* for non- Σ -entailment.

Example 3.1.1. (1) Σ -entailment is a weakening of logical consequence, that is, $\varphi_1 \models \varphi_2$ implies $\varphi_1 \models_{\Sigma} \varphi_2$ for any Σ . The converse is true when $\text{sig}(\varphi_2) \subseteq \Sigma$.

(2) Consider the GFO^2 -sentences $\varphi_1 = \forall x \exists y (Rxy \wedge x \neq y)$ and $\varphi_2 = \forall x (\exists y (Rxy \wedge Ay) \wedge \exists y (Rxy \wedge \neg Ay))$ and let $\Sigma = \{R\}$. Let \mathfrak{A} be the model of φ_1 that consists of an infinite R -path with an initial element, and let \mathfrak{B} be the model of φ_2 that consists of two infinite branching R -paths with an initial element. Then $\varphi_1 \models_{\Sigma} \varphi_2$ since the two models cannot be distinguished (see Theorem 3.3.6) as GFO^2 cannot count the number of R -successors. If φ_1 is replaced by $\varphi'_1 = \forall x \exists y Rxy$ then φ'_1 does not Σ -entail φ_2 since the sentence $\psi = \forall xy (Rxy \rightarrow x = y)$ is a witness for non- Σ -entailment.

It is important to note that different fragments F of FO give rise to different notions of Σ -entailment, Σ -inseparability and conservative extensions. For example, if φ_1 and φ_2 belong to GFO², then they also belong to GFO and to FO², but it might make a difference whether witness sentences range over all GFO²-sentences, over all GFO-sentences, or over all FO²-sentences. If we want to emphasize the fragment F in which witness sentences are formulated, we speak of $F(\Sigma)$ -entailment instead of Σ -entailment and write $\varphi_1 \models_{F(\Sigma)} \varphi_2$, and likewise for $F(\Sigma)$ -inseparability and $F(\Sigma)$ -conservative extensions.

Example 3.1.2. Let φ_1 , φ_2 , and $\Sigma = \{R\}$ be from Example 3.1.1 (2). Then φ_1 GFO²(Σ)-entails φ_2 but φ_1 does not FO(Σ)-entail φ_2 ; a witness is given by $\forall xy_1y_2((Rxy_1 \wedge Rxy_2) \rightarrow y_1 = y_2)$.

Note that conservative extensions and Σ -inseparability reduce in polynomial time to Σ -entailment (with two calls to Σ -entailment required in the case of Σ -inseparability). Moreover, conservative extensions reduce in polynomial time to Σ -inseparability. We thus state our upper bounds in terms of Σ -entailment and lower bounds in terms of conservative extensions.

There is a natural variation of each of the three notions in Definition 3.1.1 obtained by allowing to use additional ‘helper predicates’ in witness sentences. For a fragment F of FO, F -sentences φ_1, φ_2 , and a signature Σ , we say that φ_1 *strongly* Σ -entails φ_2 if φ_1 Σ' -entails φ_2 for any Σ' with $\Sigma' \cap \text{sig}(\varphi_2) \subseteq \Sigma$. Strong Σ -inseparability and strong conservative extensions are defined accordingly. Strong Σ -entailment implies Σ -entailment, but the converse may fail as shown in the following example.

Example 3.1.3. GFO(Σ)-entailment does not imply strong GFO(Σ)-entailment. Let φ_1 state that the binary predicate R is irreflexive and symmetric and let φ_2 be the conjunction of φ_1 and $\forall x(Ax \rightarrow \forall y(Rxy \rightarrow \neg Ay)) \wedge \forall x(\neg Ax \rightarrow \forall y(Rxy \rightarrow Ay))$. Thus, an $\{R\}$ -structure satisfying φ_1 can be extended to a model of φ_2 if it contains no R -cycles of odd length. Now observe that any satisfiable GFO($\{R\}$) sentence is satisfiable in a forest $\{R\}$ -structure (see Section 3.3 for a precise definition). Hence, if a GFO($\{R\}$)-sentence is satisfiable in an irreflexive and symmetric structure then it is satisfiable in a structure without odd cycles and so φ_1 GFO($\{R\}$)-entails φ_2 . In contrast, for the fresh ternary predicate Q and $\psi = \exists x_1x_2x_3(Qx_1x_2x_3 \wedge Rx_1x_2 \wedge Rx_2x_3 \wedge Rx_3x_1)$ we have $\varphi_2 \models \neg\psi$ but $\varphi_1 \not\models \neg\psi$ and so ψ witnesses that φ_1 does not GFO($\{R, Q\}$)-entail φ_2 .

The example above is inspired by proofs that GFO does not enjoy Craig interpolation [HM02, DL15]. This is not accidental, as we explain next. Recall that a fragment F of FO *has Craig interpolation* if for all F -sentences ψ_1, ψ_2 with $\psi_1 \models \psi_2$ there exists an F -sentence ψ (called an *F-interpolant for ψ_1, ψ_2*) such that $\psi_1 \models \psi \models \psi_2$ and $\text{sig}(\psi) \subseteq \text{sig}(\psi_1) \cap \text{sig}(\psi_2)$. F *has uniform interpolation* if one can always choose an F -interpolant that does not depend on ψ_2 , but only on ψ_1 and $\text{sig}(\psi_1) \cap \text{sig}(\psi_2)$. Thus, given ψ_1, ψ and Σ with $\psi_1 \models \psi$ and $\text{sig}(\psi) \subseteq \Sigma$, then ψ is a *uniform F(Σ)-interpolant of ψ_1* iff ψ strongly $F(\Sigma)$ -entails ψ_1 . Both Craig interpolation and uniform interpolation have been investigated extensively, for example for intuitionistic logic [Pit92], modal logics [Vis96, DH00, MSV15], guarded fragments [DL15], and description logics [LW11]. The following observation summarizes the connection between (strong) Σ -entailment and interpolation.

Theorem 3.1.1. *Let F be a fragment of FO that enjoys Craig interpolation. Then $F(\Sigma)$ -entailment implies strong $F(\Sigma)$ -entailment. In particular, if $\varphi_2 \models \varphi_1$ and $\text{sig}(\varphi_1) \subseteq \Sigma$, then φ_1 is a uniform $F(\Sigma)$ -interpolant of φ_2 iff φ_1 $F(\Sigma)$ -entails φ_2 .*

Proof. Assume φ_1 does not strongly $F(\Sigma)$ -entail φ_2 . Then there is an F -sentence ψ with $\text{sig}(\psi) \cap \text{sig}(\varphi_2) \subseteq \Sigma$ such that $\varphi_2 \models \psi$ and $\varphi_1 \wedge \neg\psi$ is satisfiable. Let χ be an interpolant for φ_2 and ψ in F . Then $\neg\chi$ witnesses that φ_1 does not $F(\Sigma)$ -entail φ_2 . \square

The *uniform interpolant recognition problem for F* is the problem to decide whether a sentence ψ is a uniform $F(\Sigma)$ -interpolant of a sentence ψ' . It follows from Theorem 3.1.1 that in any fragment F of FO that enjoys Craig interpolation, this problem reduces in polynomial time to Σ -inseparability in F and that, conversely, conservative extension in F reduces in polynomial time to the uniform interpolant recognition problem in F . Neither GFO nor FO^2 nor description logics with role inclusions enjoy Craig interpolation [HM02, Com69, KLWW09], but GFO^2 does [HM02]. Thus, our decidability and complexity results for Σ -entailment in GFO^2 also apply to strong Σ -entailment and the uniform interpolant recognition problem.

3.2 Undecidability Results

We prove that conservative extensions are undecidable in GFO^3 and in FO^2 , and consequently so are Σ -entailment and Σ -inseparability (as well as strong Σ -entailment and the uniform interpolant recognition problem). These results hold already without equality and in fact apply to all fragments of FO that contain at least one of GFO^3 and FO^2 such as the guarded negation fragment [BtCS15].

3.2.1 The Guarded Fragment

We start with the case of GFO^3 , using a reduction from the halting problem of two-register machines.

Definition 3.2.1. A (deterministic) *two-register machine (2RM)* is a pair $M = (Q, P)$ with $Q = q_0, \dots, q_\ell$ a set of *states* and $P = I_0, \dots, I_{\ell-1}$ a sequence of *instructions*. By definition, q_0 is the *initial state*, and q_ℓ the *halting state*. For all $i < \ell$,

- either $I_i = +(p, q_j)$ is an *incrementation instruction* with $p \in \{0, 1\}$ a register and q_j the subsequent state;
- or $I_i = -(p, q_j, q_k)$ is a *decrementation instruction* with $p \in \{0, 1\}$ a register, q_j the subsequent state if register p contains 0, and q_k the subsequent state otherwise.

A *configuration* of M is a triple (q, m, n) , with q the current state and $m, n \in \mathbb{N}$ the register contents. We write $(q_i, n_1, n_2) \Rightarrow_M (q_j, m_1, m_2)$ if one of the following holds:

- $I_i = +(p, q_j)$, $m_p = n_p + 1$, and $m_{1-p} = n_{1-p}$;
- $I_i = -(p, q_j, q_k)$, $n_p = m_p = 0$, and $m_{1-p} = n_{1-p}$;
- $I_i = -(p, q_k, q_j)$, $n_p > 0$, $m_p = n_p - 1$, and $m_{1-p} = n_{1-p}$.

The *computation* of M on input $(n, m) \in \mathbb{N}^2$ is the unique longest configuration sequence $(p_0, n_0, m_0) \Rightarrow_M (p_1, n_1, m_1) \Rightarrow_M \dots$ such that $p_0 = q_0$, $n_0 = n$, and $m_0 = m$. The halting problem for 2RMs is to decide, given a 2RM M , whether its computation on input $(0, 0)$ is finite (which implies that its last state is q_ℓ).

We show how to convert a given 2RM M into GFO³-sentences φ_1 and φ_2 such that M halts on input $(0, 0)$ iff $\varphi_1 \wedge \varphi_2$ is not a conservative extension of φ_1 . Let $M = (Q, P)$ with $Q = q_0, \dots, q_\ell$ and $P = I_0, \dots, I_{\ell-1}$. We assume w.l.o.g. that $\ell \geq 1$ and that if $I_i = -(p, q_j, q_k)$, then $q_j \neq q_k$. In φ_1 , we use the following set Σ of predicates:

- a binary predicate N connecting a configuration to its successor configuration;
- binary predicates R_1 and R_2 that represent the register contents via the length of paths;
- unary predicates q_0, \dots, q_ℓ representing the states of M ;
- a unary predicate S denoting points where a computation starts.

We define φ_1 to be the conjunction of several GFO²-sentences. First, we say that there is a point where the computation starts:¹

$$\exists x Sx \wedge \forall x (Sx \rightarrow (q_0x \wedge \neg \exists y R_0xy \wedge \neg \exists y R_1xy))$$

And second, we add that whenever M is not in the final state, there is a next configuration. For $0 \leq i < \ell$:

$$\begin{aligned} \forall x (q_ix \rightarrow \exists y (Nxy \wedge q_jy)) & \quad \text{if } I_i = +(p, q_j) \\ \forall x ((q_ix \wedge \neg \exists y R_pxy) \rightarrow \exists y (Nxy \wedge q_jy)) & \quad \text{if } I_i = -(p, q_j, q_k) \\ \forall x ((q_ix \wedge \exists y R_pxy) \rightarrow \exists y (Nxy \wedge q_ky)) & \quad \text{if } I_i = -(p, q_j, q_k) \end{aligned}$$

The second sentence φ_2 is constructed so as to express that either M does not halt or the representation of the computation of M contains a defect, using the following additional predicates:

- a unary predicate P used to represent that M does not halt;
- binary predicates D_p^+, D_p^-, D_p^- used to describe defects in incrementing, decrementing, and keeping register $p \in \{0, 1\}$;
- ternary predicates $H_1^+, H_2^+, H_1^-, H_2^-, H_1^-, H_2^-$ used as guards for existential quantifiers.

In fact, φ_2 is the disjunction of two sentences. The first sentence says that the computation does not terminate:

$$\exists x (Sx \wedge Px) \wedge \forall x (Px \rightarrow \exists y (Nxy \wedge Py))$$

while the second says that registers are not updated properly:

$$\begin{aligned} \exists x \exists y (Nxy \wedge (& \bigvee_{I_i=+(p, q_j)} (q_ix \wedge q_jy \wedge (D_p^+xy \vee D_{1-p}^-xy)) \\ & \vee \bigvee_{I_i=-(p, q_j, q_k)} (q_ix \wedge q_ky \wedge (D_p^-xy \vee D_{1-p}^-xy)) \\ & \vee \bigvee_{I_i=-(p, q_j, q_k)} (q_ix \wedge q_jy \wedge (D_p^-xy \vee D_{1-p}^-xy))) \\ \wedge \forall x \forall y (D_p^+xy \rightarrow & (\neg \exists z R_pyz \vee (\neg \exists z R_pxz \wedge \exists z (R_pyz \wedge \exists x R_pzx))) \\ & \vee \exists z (H_1^+xyz \wedge R_pxz \wedge \exists x (H_2^+xzy \wedge R_pyx \wedge D_p^+zx))). \end{aligned}$$

¹The formulas that are not syntactically guarded can easily be rewritten into such formulas.

In this second sentence, additional conjuncts that implement the desired behaviour of D_p^- and D_p^+ are also needed; they are constructed analogously to the last three lines above (but using guards H_j^- and H_j^+). We now prove the correctness of the reduction.

Lemma 3.2.1.

1. If M halts, then $\varphi_1 \wedge \varphi_2$ is not a GFO^2 -conservative extension of φ_1 .
2. If there exists a Σ -structure that satisfies φ_1 and cannot be extended to a model of φ_2 (by interpreting the predicates in $\text{sig}(\varphi_2) \setminus \text{sig}(\varphi_1)$), then M halts.

In the proof of Point 1, the sentence that witnesses non-conservativity describes a halting computation of M , up to global $\text{GFO}^2(\Sigma)$ -bisimulations. This can be done using only two variables. We split the proof of Lemma 3.2.1 into two parts.

Lemma 3.2.2. *If M halts then $\varphi_1 \wedge \varphi_2$ is not a GFO^2 -conservative extension of φ_1 .*

Proof. Assume that M halts. We define a witness ψ for non-conservativity. It says that every element participates in a substructure that represents the computation of M on input $(0, 0)$, that is: if the computation is $(q_0, n_0, m_0), \dots, (q_k, n_k, m_k)$, then there is an N -path of length k (but not longer) such that any object reachable in $i \leq k$ steps from the beginning of the path is labeled with q_i , has an outgoing R_0 -path of length n_i and no longer outgoing R_0 -path, and likewise for R_1 and m_i . In more detail, consider the Σ -structure \mathfrak{A} with

$$A = \{0, \dots, k\} \cup \{a_j^i \mid 0 < i \leq k, 0 < j < n_i\} \cup \{b_j^i \mid 0 < i \leq k, 0 < j < m_i\}$$

in which

$$\begin{aligned} N^{\mathfrak{A}} &= \{(i, i+1) \mid i < k\} \\ R_1^{\mathfrak{A}} &= \bigcup_{i \leq k} \{(i, a_1^i), (a_1^i, a_2^i), \dots, (a_{n_i-2}^i, a_{n_i-1}^i)\} \\ R_2^{\mathfrak{A}} &= \bigcup_{i \leq k} \{(i, b_1^i), (b_1^i, b_2^i), \dots, (b_{m_i-2}^i, b_{m_i-1}^i)\} \\ S^{\mathfrak{A}} &= \{0\} \\ q^{\mathfrak{A}} &= \{i \mid q_i = q\} \text{ for any } q \in Q. \end{aligned}$$

Then let ψ be a $\text{GFO}^2(\Sigma)$ -sentence that describes \mathfrak{A} up to global $\text{GFO}^2(\Sigma)$ -bisimulations. Clearly \mathfrak{A} satisfies $\varphi_1 \wedge \psi$. It thus remains to show that $\varphi_1 \wedge \varphi_2 \wedge \psi$ is unsatisfiable. But this is clear as there are no N -paths of length $> k$ in any model of ψ and since there are no defects in register updates in any model of ψ . \square

Lemma 3.2.3. *If there exists a Σ -structure that satisfies φ_1 and cannot be extended to a model of φ_2 , then M halts.*

Proof. Let \mathfrak{A} be a Σ -structure satisfying φ_1 that cannot be extended to a model of φ_2 . Then $S^{\mathfrak{A}} \neq \emptyset$ and there exists an N -path labeled with states in Q starting in S . Since \mathfrak{A} cannot be extended to a model of φ_2 the computation starting from S is finite. Moreover, one can readily prove by induction that no register update defects occur since otherwise φ_2 can be satisfied. \square

The following result is an immediate consequence of Lemma 3.2.1.

Theorem 3.2.4. *In any fragment of FO that extends the three-variable guarded fragment GFO^3 , the following problems are undecidable: conservative extensions, Σ -inseparability, Σ -entailment, and strong Σ -entailment.*

Since Point 1 of Lemma 3.2.1 ensures GFO^2 -witnesses, Theorem 3.2.4 can actually be strengthened to say that GFO^2 -conservative extensions of GFO^3 -sentences are undecidable.

3.2.2 The Two-Variable Fragment

Our result for FO^2 is proved by a reduction of a tiling problem that asks for the tiling of a rectangle (of any size) such that the borders are tiled with certain distinguished tiles.

Definition 3.2.2. A *tiling system* $\mathfrak{D} = (\mathfrak{T}, H, V, \text{Right}, \text{Left}, \text{Top}, \text{Bottom})$ consists of a finite set \mathfrak{T} of *tiles*, horizontal and vertical *matching relations* $H, V \subseteq \mathfrak{T} \times \mathfrak{T}$, and subsets **Right**, **Left**, **Top**, and **Bottom** of \mathfrak{T} containing the *right* tiles, *left* tiles, *top* tiles, and *bottom* tiles, respectively. A *solution* to \mathfrak{D} is a triple (n, m, τ) where $n, m \in \mathbb{N}$ and $\tau : \{0, \dots, n-1\} \times \{0, \dots, m-1\} \rightarrow \mathfrak{T}$ such that the following hold:

1. $(\tau(i, j), \tau(i+1, j)) \in H$, for all $i < n$ and $j \leq m$;
2. $(\tau(i, j), \tau(i, j+1)) \in V$, for all $i \leq n$ and $j < m$;
3. $\tau(0, j) \in \text{Left}$ and $\tau(n, j) \in \text{Right}$, for all $j \leq m$;
4. $\tau(i, 0) \in \text{Bottom}$ and $\tau(i, m) \in \text{Top}$, for all $i \leq n$.

We show how to convert a tiling system \mathfrak{D} into FO^2 -sentences φ_1 and φ_2 such that \mathfrak{D} has a solution iff $\varphi_1 \wedge \varphi_2$ is not a conservative extension of φ_1 . In particular, models of witness sentences will define solutions of \mathfrak{D} .

Let $\mathfrak{D} = (\mathfrak{T}, H, V, \text{Right}, \text{Left}, \text{Top}, \text{Bottom})$ be a tiling system. The formula φ_1 uses the following set Σ of predicates:

- binary predicates R_h and R_v (representing a grid) and S_h and S_v (for technical reasons),
- unary predicates T for every $T \in \mathfrak{T}$, G (for the domain of the grid), O (for the lower left corner of the grid), B_{\rightarrow} , B_{\leftarrow} , B_{\uparrow} , and B_{\downarrow} (for the borders of the grid).

Then φ_1 is the conjunction of the following sentences:

1. Every position in the $n \times m$ grid is labeled with exactly one tile and the matching conditions are satisfied:

$$\begin{aligned} & \forall x (Gx \rightarrow \bigvee_{T \in \mathfrak{T}} (Tx \wedge \bigwedge_{T' \in \mathfrak{T}, T' \neq T} \neg T'x)) \\ & \forall x (Gx \rightarrow \bigwedge_{T \in \mathfrak{T}} (Tx \rightarrow (\bigvee_{(T, T') \in H} \forall y (R_h xy \rightarrow T'y) \wedge \bigvee_{(T, T') \in V} \forall y (R_v xy \rightarrow T'y))))). \end{aligned}$$

2. The predicates B_{\rightarrow} , B_{\leftarrow} , B_{\uparrow} , and B_{\downarrow} mark the borders of the grid:

$$\begin{aligned} & \forall x(Gx \wedge B_{\rightarrow}x \rightarrow (\neg\exists yR_hxy \wedge \forall y(R_vxy \rightarrow B_{\rightarrow}y) \wedge \forall y(R_vyx \rightarrow B_{\rightarrow}y))) \\ & \forall x(Gx \wedge \neg B_{\rightarrow}x \rightarrow \exists yR_hxy) \end{aligned}$$

and similarly for B_{\leftarrow} , B_{\uparrow} , and B_{\downarrow} .

3. There is a grid origin:

$$\exists x(Ox \wedge B_{\leftarrow}x \wedge B_{\downarrow}x).$$

4. The grid elements are marked by G :

$$\forall x(Ox \rightarrow Gx), \quad \forall x(Gx \rightarrow \forall y(R_hxy \rightarrow Gy)), \quad \forall x(Gx \rightarrow \forall y(R_vxy \rightarrow Gy)).$$

5. The tiles on border positions are labeled with appropriate tiles:

$$\forall x(B_{\rightarrow}x \rightarrow \bigvee_{T \in \text{Right}} T(x)).$$

and similarly for B_{\leftarrow} , B_{\uparrow} , and B_{\downarrow} .

6. The predicates S_h and S_v occur in φ_1 : any FO²-tautology using them.

This finishes the definition of φ_1 . The sentence φ_2 introduces two new unary predicates Q and P and is the conjunction of $\exists x(Ox \wedge Qx)$ and

$$\forall x(Qx \rightarrow (\exists y(R_hxy \wedge Qy) \vee \exists y(R_vxy \wedge Qy) \vee \varphi_Dx))$$

where

$$\varphi_Dx = \exists y(R_hxy \wedge \forall x(R_vyx \rightarrow Px)) \wedge \exists y(R_vxy \wedge \forall x(R_hyx \rightarrow \neg Px))$$

Thus, φ_D describes a defect in the grid: there exist an R_h -successor y_1 and an R_v -successor y_2 of x such that every R_v -successor of y_1 is labeled with P and every R_h -successor of y_2 is labeled with $\neg P$. Informally, we can satisfy φ_2 only if from some element of O , there is an infinite R_h/R_v -path or a non-closing grid cell can be reached by a finite such path. To make this precise, we introduce some notation. Let $\Sigma' \supseteq \Sigma$ and let \mathfrak{B} be a Σ' -structure. Then the Σ -structure \mathfrak{A} obtained from \mathfrak{B} by removing the interpretation of predicates in $\Sigma' \setminus \Sigma$ is called the Σ -*reduct* of \mathfrak{B} and \mathfrak{B} is called a $\Sigma' \setminus \Sigma$ -*extension* of \mathfrak{A} . For a Σ -structure \mathfrak{A} , we say that $a \in A$ is the *root of a non-closing grid cell* if there are $(a, b_1) \in R_h^{\mathfrak{A}}$ and $(a, b_2) \in R_v^{\mathfrak{A}}$ such that there does not exist a $c \in A$ with $(b_1, c) \in R_v^{\mathfrak{A}}$ and $(b_2, c) \in R_h^{\mathfrak{A}}$. Now, we show the following characterization of φ_2 .

Lemma 3.2.5. φ_2 can be satisfied in a $\{Q, P\}$ -extension of a Σ -structure \mathfrak{A} iff there exists an element of $O^{\mathfrak{A}}$ that starts an infinite R_h/R_v -path or a finite R_h/R_v -path to a root of a non-closing grid cell.

Proof.(sketch) Assume that φ_2 is satisfied in a $\{Q, P\}$ -extension of a Σ -structure \mathfrak{A} . By definition of φ_2 , we can find an assignment of Q that satisfies φ_2 such that there is an infinite outgoing R_h/R_v -path starting at an element of $O^{\mathfrak{A}}$; or we can find an assignment of $(Q$ and) P that satisfies φ_2 such that there is a finite R_h/R_v -path to a root of a non-closing grid cell.

For the other direction, assume there exists an element of $O^{\mathfrak{A}}$ that starts an infinite R_h/R_v -path or there is a finite R_h/R_v -path to a root of a non-closing grid cell. Then φ_1 cannot be satisfied in such a Σ -structure \mathfrak{A} . We thus define a structure \mathfrak{B} as a $\{Q, P\}$ -extension of \mathfrak{A} , and therefore, φ_2 is clearly satisfied in \mathfrak{B} . \square

We now argue that \mathfrak{D} has a solution iff $\varphi_1 \wedge \varphi_2$ is not a conservative extension of φ_1 .

Lemma 3.2.6. *If \mathfrak{D} has a solution then $\varphi_1 \wedge \varphi_2$ is not an FO^2 -conservative extension of φ_1 .*

Proof. Assume that \mathfrak{D} has a solution (n, m, τ) . We define a witness ψ , first using additional fresh unary predicates and then arguing that these can be removed. Thus introduce fresh unary predicates $P_{i,j}$ for all $i < n$ and $j < m$. Intuitively, $P_{i,j}$ identifies grid position (i, j) . Set

$$\begin{aligned} \psi = & \forall x (Gx \rightarrow \bigvee_{i,j} P_{i,j}x) \\ & \wedge \bigwedge_{(i,j) \neq (i',j')} \forall x \neg (P_{i,j}x \wedge P_{i',j'}x) \\ & \wedge \forall x \forall y (R_hxy \leftrightarrow \bigvee_{i,j} P_{i,j}x \wedge P_{i+1,j}y) \\ & \wedge \forall x \forall y (R_vx, y \leftrightarrow \bigvee_{i,j} P_{i,j}x \wedge P_{i,j+1}y) \\ & \wedge \forall x (Ox \rightarrow P_{0,0}x). \end{aligned}$$

We first show that $\varphi_1 \wedge \psi$ is satisfiable. As the model, simply take the standard $n \times m$ -grid in which all positions are labeled with $P_{i,j}$, G , O , B_{\rightarrow} etc in the expected way, and that is tiled according to τ . It is easily verified that this structure satisfies both φ_1 and ψ . It thus remains to show that $\varphi_1 \wedge \varphi_2 \wedge \psi$ is unsatisfiable. By Lemma 3.2.5 it suffices to show that there is no model \mathfrak{A} of $\varphi_1 \wedge \psi$ in which there exists an element of $O^{\mathfrak{A}}$ starting an infinite R_h/R_v -path or a finite R_h/R_v -path leading to a root of a non-closing grid cell. Assume for a proof by contradiction that there exists such a model \mathfrak{A} and $a \in O^{\mathfrak{A}}$. Then we find a sequence $a_0 R_{z_0}^{\mathfrak{A}} a_1 R_{z_1}^{\mathfrak{A}} a_2 \cdots$ with $a_0 = a$ and $z_i \in \{h, v\}$ such that either some a_h is the root of a non-closing grid cell or the sequence is infinite. By φ_1 and the first conjunct of ψ for each a_k there exists $P_{i,j}$ with $a_k \in P_{i,j}^{\mathfrak{A}}$. By the last conjunct of ψ , $a_0 \in P_{0,0}^{\mathfrak{A}}$. By the remaining conjuncts of ψ we have $k \geq i + j$ if $a_k \in P_{i,j}^{\mathfrak{A}}$ and it follows that there is no a_k with $k > n + m$. Thus, assume some a_k is the root of a non-closing grid. Then we have $(a_k, b_1) \in R_h^{\mathfrak{A}}$ and $(a_k, b_2) \in R_v^{\mathfrak{A}}$ such that there is no $c \in A$ with $(b_1, c) \in R_v^{\mathfrak{A}}$ and $(b_2, c) \in R_h^{\mathfrak{A}}$. By ψ , there are i, j with $b_1 \in P_{i+1,j}^{\mathfrak{A}}$ and $b_2 \in P_{i,j+1}^{\mathfrak{A}}$. By the second set of conjuncts of φ_1 there exists $(b_1, c) \in R_v^{\mathfrak{A}}$. By ψ , $c \in P_{i+1,j+1}^{\mathfrak{A}}$. But then again using ψ we obtain that $(b_2, c) \in R_h^{\mathfrak{A}}$ and we have derived a contradiction.

We now show how to remove the additional predicates $P_{i,j}$. To this end, we use the binary predicates S_h, S_v . In the sentence ψ , we replace every occurrence of $P_{i,j}(x)$ with a formula saying: there is an outgoing S_h -path of length i , but not of length $i + 1$ and an outgoing S_v -path of length j , but not of length $j + 1$. When we build a model of $\varphi_1 \wedge \psi$, we now need to introduce additional elements for the “counting paths”. We make the predicate G false on all those elements and true everywhere on the grid. \square

Lemma 3.2.7. *If there exists a Σ -structure that satisfies φ_1 and cannot be extended to a model of φ_2 , then \mathfrak{D} has solution.*

Proof. Take a Σ -structure \mathfrak{A} satisfying φ_1 that cannot be extended to a model of φ_2 . By the conjunct of φ_1 given in Item 3, $O^{\mathfrak{A}} \cap B_{\leftarrow}^{\mathfrak{A}} \cap B_{\downarrow}^{\mathfrak{A}} \neq \emptyset$. Take $a \in O^{\mathfrak{A}} \cap B_{\leftarrow}^{\mathfrak{A}} \cap B_{\downarrow}^{\mathfrak{A}}$. By Lemma 3.2.5, a does not start an infinite R_h/R_v -path or a finite R_h/R_v -path leading to the root of a non-closing grid cell. Using the conjuncts of φ_1 it is now straightforward to read off a tiling from \mathfrak{A} . \square

The following result is an immediate consequence of Lemmas 3.2.6 and 3.2.7.

Theorem 3.2.8. *In any fragment of FO that extends FO², the following problems are undecidable: conservative extensions, Σ -inseparability, Σ -entailment, and strong Σ -entailment.*

Note that the sentences φ_1 and φ_2 can be replaced by equivalent \mathcal{ALC} -TBoxes: in φ_2 , we can replace the conjunct $\exists x(Ox \wedge Qx)$, which cannot be expressed in \mathcal{ALC} , by the concept inclusion $\top \sqsubseteq \exists S.(O \sqcap Q)$ with S a fresh binary predicate. Consequently, the proof of Theorem 3.2.8 also shows that FO²-conservative extensions of \mathcal{ALC} -TBoxes are undecidable while it follows from our results in the next section that GFO²-conservative extensions of \mathcal{ALC} -TBoxes are decidable.

3.3 Model-theoretic Characterization

The undecidability results established in the previous section show that neither the restriction to two variables nor guardedness alone are sufficient for decidability of conservative extensions and related problems. We now show that adopting both restrictions simultaneously results in decidability of Σ -entailment (and thus also of conservative extensions and of inseparability). We proceed by first establishing a suitable model-theoretic characterization based on bisimulations to then use it as the foundation for a decision procedure based on tree automata. We in fact establish two versions of the characterization, the second one building on the first one.

3.3.1 GFO²-Bisimulations

In this section we introduce GFO²-bisimulations and variants thereof that will be needed to give a model-theoretic characterization of Σ -entailment.

We start with some preliminaries. An *atomic 1-type for Σ* is a maximal satisfiable set τ of atomic GFO²(Σ)-formulas and their negations that use the variable x , only. We use $\text{at}_{\mathfrak{A}}^{\Sigma}(a)$ to denote the atomic 1-type for Σ realized by the element a in the structure \mathfrak{A} . An *atomic 2-type for Σ* is a maximal satisfiable set τ of atomic GFO²(Σ)-formulas and their negations that use the variables x and y , only, and contains $\neg(x = y)$. We say that τ is *guarded* if it contains an atom of the form Rxy or Ryx , R a predicate symbol. We use $\text{at}_{\mathfrak{A}}^{\Sigma}(a, b)$ to denote the atomic 2-type for Σ realized by the elements a, b in the structure \mathfrak{A} . A relation $\sim \subseteq A \times B$ is a *GFO²(Σ)-bisimulation between \mathfrak{A} and \mathfrak{B}* if the following conditions hold whenever $a \sim b$:

1. $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$;
2. for every $a' \neq a$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ is guarded, there is a $b' \neq b$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a') = \text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ and $a' \sim b'$ (forth condition);
3. for every $b' \neq b$ such that $\text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ is guarded, there is an $a' \neq a$ such that $\text{at}_{\mathfrak{B}}^{\Sigma}(b, b') = \text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ and $a' \sim b'$ (back condition).

We write $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$ and say that (\mathfrak{A}, a) and (\mathfrak{B}, b) are *GFO²(Σ)-bisimilar* if there is a GFO²(Σ)-bisimulation \sim between \mathfrak{A} and \mathfrak{B} with $a \sim b$. If the domain and range of \sim coincide with A and B , respectively, then \sim is called a *global GFO²(Σ)-bisimulation*.

We continue giving a bounded version of bisimulations. For $k \geq 0$, we write $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$ and say that (\mathfrak{A}, a) and (\mathfrak{B}, b) are *k-GFO²(Σ)-bisimilar* if there is a $\sim \subseteq A \times B$ such that the first condition for bisimulations holds and the back and forth

conditions can be iterated up to k times starting from a and b . More formally, let \mathfrak{A} and \mathfrak{B} be structures, $a \in A$, and $b \in B$. The definition k -GFO²(Σ) bisimilarity is by induction on $k \geq 0$. Then $(\mathfrak{A}, a) \sim_{\Sigma}^0 (\mathfrak{B}, b)$ iff $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$ and $(\mathfrak{A}, a) \sim_{\Sigma}^{k+1} (\mathfrak{B}, b)$ iff $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$ and

1. for every $a' \neq a$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ is guarded, there exists $b' \neq b$ such that $\text{at}_{\mathfrak{B}}^{\Sigma}(b, b') = \text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ and $(\mathfrak{A}, a') \sim_{\Sigma}^k (\mathfrak{B}, b')$
2. for every $b' \neq b$ such that $\text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ is guarded, there exists $a' \neq a$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a') = \text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ and $(\mathfrak{A}, a') \sim_{\Sigma}^k (\mathfrak{B}, b')$.

Call structures \mathfrak{A} and \mathfrak{B} *globally k -GFO²(Σ)-bisimilar* if for all $a \in A$ there exists $b \in B$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$ and, conversely, for every $b \in B$ there exists $a \in A$ with $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$. \mathfrak{A} and \mathfrak{B} are *globally finitely GFO²(Σ)-bisimilar* iff they are globally k -GFO²(Σ)-bisimilar for all $k \geq 0$.

Example 3.3.1. Consider the GFO²-sentences $\varphi_1 = \forall x \exists y Rxy$ and $\varphi_2 = \varphi_1 \wedge \exists x Bx \wedge \forall x (Bx \rightarrow \exists y (Ryx \wedge By))$ and let $\Sigma = \{R\}$. Let \mathfrak{A} be the model of φ_1 that consists of an infinite R -path with an initial element. Then there is no model of φ_2 that is globally GFO²($\{R\}$)-bisimilar to \mathfrak{A} since any such model has to contain an infinite R -path with no initial element. Yet, φ_2 is a conservative extension of φ_1 as all possible witnesses are finite and can thus see only a finite part of the infinite backwards path. To capture this insight, we need to consider bounded bisimulations, for arbitrarily large bounds (see Theorem 3.3.6 below).

Expressive Power

We now show that GFO²(Σ)-bisimulations characterize the expressive power of GFO²(Σ)-sentences. The proofs are standard [GO14, GO06, ANv98]. We first characterize the expressive power of a fragment of GFO², that we call openGFO^2 , and then we do it for the general case.

Denote by openGFO^2 the fragment of GFO² consisting of all GFO² formulas with one free variable in which equality is not used as a guard and which do not contain a subformula that is a sentence. It is not difficult to prove the following result.

Lemma 3.3.1. *Every GFO²-sentence is equivalent to a Boolean combination of sentences of the form $\forall x \varphi(x)$, where $\varphi(x)$ is an openGFO^2 formula.*

To characterize GFO² we often need structures that satisfy certain saturation conditions. A structure \mathfrak{A} is ω -saturated if for every finite set $\{a_1, \dots, a_n\} \subseteq A$ and every set $\Gamma(x)$ of FO formulas using elements of $\{a_1, \dots, a_n\}$ as constants the following holds: if every finite subset of $\Gamma(x)$ is satisfiable in the structure $(\mathfrak{A}, a_1, \dots, a_n)$, then $\Gamma(x)$ is satisfiable in $(\mathfrak{A}, a_1, \dots, a_n)$. For every structure \mathfrak{A} there exists an elementary extension \mathfrak{A}' of \mathfrak{A} that is ω -saturated [CK90]. Mostly we only require a weaker form of saturation. A structure \mathfrak{A} is *successor-saturated* if for any $a \in A$ and set $\Gamma(x)$ of openGFO^2 formulas the following holds for any atomic guarded binary type τ : if for any finite subset Γ' of Γ there exists $a' \neq a$ with $\text{at}_{\mathfrak{A}}(a, a') = \tau$ and $\mathfrak{A} \models \psi(a')$ for all $\psi \in \Gamma'$, then there exists $b' \neq a$ with $\text{at}_{\mathfrak{A}}(a, b') = \tau$ and $\mathfrak{A} \models \psi(b')$ for all $\psi \in \Gamma$. Observe that structures of finite outdegree and ω -saturated structures are successor-saturated.

The *depth* of a GFO² formula φ is the number of nestings of guarded quantifications in φ . We first characterize openGFO^2 .

Lemma 3.3.2. *Let \mathfrak{A} and \mathfrak{B} be structures, Σ a signature, and $a \in A$, $b \in B$.*

1. The following conditions are equivalent for all $k \geq 0$:
 - $\mathfrak{A} \models \varphi(a)$ iff $\mathfrak{B} \models \varphi(b)$ holds for all open $GFO^2(\Sigma)$ formulas $\varphi(x)$ of depth k ;
 - $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$.
2. If $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$, then $\mathfrak{A} \models \varphi(a)$ iff $\mathfrak{B} \models \varphi(b)$ holds for all open $GFO^2(\Sigma)$ formulas $\varphi(x)$. The converse direction holds if \mathfrak{A} and \mathfrak{B} are successor-saturated.

The proof is standard and omitted. We also require the following link between bounded bisimulations and unbounded bisimulations which follows from Lemma 3.3.2.

Lemma 3.3.3. *Let \mathfrak{A} and \mathfrak{B} be successor-saturated structures, $a \in A$, and $b \in B$. If $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$ for all $k \geq 0$, then $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$.*

In order to characterize GFO^2 we need to consider global bounded bisimulations. The following characterization result now follows from Lemma 3.3.1 and Lemma 3.3.2.

Lemma 3.3.4. *Let \mathfrak{A} and \mathfrak{B} be structures and Σ a signature.*

1. The following conditions are equivalent:
 - $\mathfrak{A} \models \varphi$ iff $\mathfrak{B} \models \varphi$ holds for all $GFO^2(\Sigma)$ sentences φ ;
 - \mathfrak{A} and \mathfrak{B} are globally finitely $GFO^2(\Sigma)$ -bisimilar.
2. (a) If \mathfrak{A} and \mathfrak{B} are globally $GFO^2(\Sigma)$ -bisimilar, then $\mathfrak{A} \models \varphi$ iff $\mathfrak{B} \models \varphi$ holds for all $GFO^2(\Sigma)$ sentences φ . (b) The converse direction holds if \mathfrak{A} and \mathfrak{B} are ω -saturated.

Observe that in Lemma 3.3.4 we cannot replace ω -saturation by successor-saturation or finite outdegree.

Theorem 3.3.5. *An FO-sentence φ is equivalent to a GFO^2 -sentence iff its models are preserved under global $GFO^2(\text{sig}(\varphi))$ -bisimulations.*

Proof. One direction follows by Lemma 3.3.4 2.(a). For the other direction, let φ be an FO-sentence whose models are preserved under global $GFO^2(\text{sig}(\varphi))$ -bisimulations. We want to show that φ is equivalent to a GFO^2 -sentence.

We first show that whenever two structures agree on all sentences of GFO^2 , they agree on φ . Suppose \mathfrak{A} and \mathfrak{B} satisfy the same GFO^2 -sentences. Without loss of generality, we can assume that \mathfrak{A} and \mathfrak{B} are ω -saturated. By Lemma 3.3.4 2.(b) we have that \mathfrak{A} and \mathfrak{B} are globally $GFO^2(\text{sig}(\varphi))$ -bisimilar. Assume now that $\mathfrak{A} \models \varphi$. As the models of φ are preserved under global $GFO^2(\text{sig}(\varphi))$ -bisimulations, and \mathfrak{A} and \mathfrak{B} are globally $GFO^2(\text{sig}(\varphi))$ -bisimilar, this implies that $\mathfrak{B} \models \varphi$. By symmetry we get $\mathfrak{A} \models \varphi$ iff $\mathfrak{B} \models \varphi$ as desired.

Now assume that φ is satisfiable, i.e., there is a structure \mathfrak{A} such that $\mathfrak{A} \models \varphi$, and let Θ be the set of all GFO^2 -sentences θ such that $\mathfrak{A} \models \theta$. We show that $\Theta \models \varphi$ (i.e., any model of Θ is a model of φ). If this were not the case then we have a structure \mathfrak{B} such that $\mathfrak{B} \models \Theta \wedge \neg\varphi$. But because Θ contains each GFO^2 -sentence or its negation we have that \mathfrak{A} and \mathfrak{B} are globally $GFO^2(\text{sig}(\varphi))$ -bisimilar and \mathfrak{A} and \mathfrak{B} disagree on φ . This contradicts the claim of the previous paragraph.

Since $\Theta \models \varphi$ then, by compactness, there is a finite subset Θ' of Θ such that $\Theta' \models \varphi$. By construction, this implies that φ is equivalent to the conjunction of all the sentences in Θ' and therefore equivalent to a GFO^2 -sentence. \square

With these notions at hand we are now ready to give a model-theoretic characterization of Σ -entailment using an appropriate notion of GFO^2 -bisimulation.

3.3.2 Characterization of Σ -Entailment

In modal and description logics, global Σ -bisimulations can often be used to characterize Σ -entailment in the following natural way [LW11]: φ_1 Σ -entails φ_2 iff for every (tree) model \mathfrak{A} of φ_1 , there exists a (tree) model \mathfrak{B} of φ_2 that is globally Σ -bisimilar to \mathfrak{A} . Such a characterization enables decision procedures based on tree automata, but does not hold for GFO^2 as shown in Example 3.3.1.

We give our first characterization theorem that uses unbounded bisimulations in one direction and bounded bisimulations in the other.

Theorem 3.3.6. *Let φ_1, φ_2 be GFO^2 -sentences and Σ a signature. Then $\varphi_1 \models_{\Sigma} \varphi_2$ iff for every model \mathfrak{A} of φ_1 of finite outdegree, there is a model \mathfrak{B} of φ_2 such that*

1. *for every $a \in A$ there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$*
2. *for every $b \in B$ and every $k \geq 0$, there is an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$*

Proof. “if”. Assume that for every model \mathfrak{A} of φ_1 of finite outdegree, there is a model \mathfrak{B} of φ_2 as described in Theorem 3.3.6. Take a Σ -sentence ψ such that $\varphi_1 \wedge \psi$ is satisfiable. We have to show that $\varphi_2 \wedge \psi$ is satisfiable. We find a model \mathfrak{A} of $\varphi_1 \wedge \psi$ that has finite outdegree. By assumption, there is a model \mathfrak{B} of φ_2 that satisfies Conditions 1 and 2 of Theorem 3.3.6. It suffices to show that \mathfrak{B} satisfies ψ . But this follows from Lemma 3.3.4.

“only if”. Assume that $\varphi_1 \models_{\Sigma} \varphi_2$. Let \mathfrak{A} be a model of φ_1 of finite outdegree. Let Γ denote the set of all $\text{GFO}^2(\Sigma)$ sentences ψ with $\mathfrak{A} \models \psi$. Then $\varphi_1 \wedge \bigwedge \Gamma$ is satisfiable for every finite subset Γ' of Γ . As $\varphi_1 \models_{\Sigma} \varphi_2$, $\varphi_2 \wedge \bigwedge \Gamma'$ is satisfiable for every finite subset Γ' of Γ . By compactness $\{\varphi_2\} \cup \Gamma$ is satisfiable. Then there exists an ω -saturated model \mathfrak{B} of $\{\varphi_2\} \cup \Gamma$. By ω -saturatedness, for every $a \in A$ there exists $b \in B$ such that $\mathfrak{A} \models \varphi(a)$ iff $\mathfrak{B} \models \varphi(b)$ holds for all formulas $\varphi(x)$ in $\text{openGFO}^2(\Sigma)$. By Lemma 3.3.2, we have $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$, as required for Condition 1. Condition 2 follows from Lemma 3.3.2. \square

Because of the use of k -bounded bisimulations (for unbounded k), it is not clear how to use Theorem 3.3.6 to find a decision procedure based on tree automata. In the following, we formulate a more ‘operational’ but also more technical characterization that no longer mentions bounded bisimulations. It additionally refers to forest models \mathfrak{A} of φ_1 (of finite outdegree) instead of unrestricted models, but we remark that Theorem 3.3.6 also remains true under this modification.

A structure \mathfrak{A} is a *forest* if its Gaifman graph is a forest. Thus, a forest admits cycles of length one and two, but not of any higher length. A (Σ) -*tree* in a forest structure \mathfrak{A} is a maximal (Σ) -connected substructure of \mathfrak{A} . When working with forest structures \mathfrak{A} , we will typically view them as directed forests rather than as undirected ones. This can be done by choosing a root for each tree in the Gaifman graph of \mathfrak{A} , thus giving rise to notions such as successor, descendant, etc. Which node is chosen as the root will always be irrelevant. Note that the direction of binary relations does not need to reflect the successor relation. When speaking of a *path* in a forest structure \mathfrak{A} , we mean a path in the directed sense; when speaking of a *subtree*, we mean a tree that is obtained by choosing a root a and restricting the structure to a and its descendants. We say that \mathfrak{A} is *regular* if it has only finitely many subtrees, up to isomorphism.

To see how we can get rid of bounded bisimulations, reconsider Theorem 3.3.6. The characterization is still correct if we pull out the quantification over k in Point 2 so that the theorem reads ‘...iff for every model \mathfrak{A} of φ_1 of finite outdegree and every $k \geq 0$, there is...’. In fact, this modified version of Theorem 3.3.6 is even closer to the definition of Σ -entailment. It also suggests that we add a marking $A_\perp \subseteq A$ of elements in \mathfrak{A} , representing ‘break-off points’ for bisimulations, and then replace k -bisimulations with bisimulations that stop whenever they have encountered the *second* marked element on the same path—in this way, the distance between marked elements (roughly) corresponds to the bound k in k -bisimulations. However, we would need a marking A_\perp , for any $k \geq 0$, such that there are infinitely many markers on any infinite path and the distance between any two markers in a tree is at least k . It is easy to see that such a marking may not exist, for example when $k = 3$ and \mathfrak{A} is the infinite full binary tree. We solve this problem as follows. First, we only demand that the distance between any two markers *on the same path* is at least k . And second, we use the markers only when following bisimulations upwards in a tree while downwards, we use unbounded bisimulations. This does not compromise correctness of the characterization.

We next introduce a version of bisimulations that implement the ideas just explained. Let \mathfrak{A} and \mathfrak{B} be forest models, Σ a signature, and $A_\perp \subseteq A$. Two relations $\sim_{\Sigma}^{A_\perp, 0}, \sim_{\Sigma}^{A_\perp, 1} \subseteq A \times B$ form an A_\perp -delimited $GFO^2(\Sigma)$ -bisimulation between \mathfrak{A} and \mathfrak{B} if the following conditions are satisfied:

1. if $(\mathfrak{A}, a) \sim_{\Sigma}^{A_\perp, 0} (\mathfrak{B}, b)$, then $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$ and
 - (a) for every $a' \neq a$ with $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ guarded, there is a $b' \neq b$ such that $(\mathfrak{A}, a') \sim_{\Sigma}^{A_\perp, i} (\mathfrak{B}, b')$ where $i = 1$ if a' is the predecessor of a and $a' \in A_\perp$, and $i = 0$ otherwise;
 - (b) for every $b' \neq b$ with $\text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ guarded, there is an $a' \neq a$ such that $(\mathfrak{A}, a') \sim_{\Sigma}^{A_\perp, i} (\mathfrak{B}, b')$ where $i = 1$ if a' is the predecessor of a and $a' \in A_\perp$, and $i = 0$ otherwise;
2. if $(\mathfrak{A}, a) \sim_{\Sigma}^{A_\perp, 1} (\mathfrak{B}, b)$ and the predecessor of a in \mathfrak{A} is not in A_\perp , then $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$ and
 - (a) for every $a' \neq a$ with $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ guarded, there is a $b' \neq b$ such that $(\mathfrak{A}, a') \sim_{\Sigma}^{A_\perp, i} (\mathfrak{B}, b')$ where $i = 0$ if a is the predecessor of a' and $a \in A_\perp$, and $i = 1$ otherwise;
 - (b) for every $b' \neq b$ with $\text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ guarded, there is an $a' \neq a$ such that $(\mathfrak{A}, a') \sim_{\Sigma}^{A_\perp, i} (\mathfrak{B}, b')$ where $i = 0$ if a is the predecessor of a' and $a \in A_\perp$, and $i = 1$ otherwise.

Then (\mathfrak{A}, a) and (\mathfrak{B}, b) are A_\perp -delimited $GFO^2(\Sigma)$ -bisimilar, in symbols $(\mathfrak{A}, a) \sim_{\Sigma}^{A_\perp} (\mathfrak{B}, b)$, if there exists an A_\perp -delimited $GFO^2(\Sigma)$ -bisimulation $\sim_{\Sigma}^{A_\perp, 0}, \sim_{\Sigma}^{A_\perp, 1}$ between \mathfrak{A} and \mathfrak{B} such that $(\mathfrak{A}, a) \sim_{\Sigma}^{A_\perp, 0} (\mathfrak{B}, b)$.

Let φ be a GFO^2 -sentence. We use $\text{cl}(\varphi)$ to denote the set of all subformulas of φ closed under single negation and renaming of free variables (using only the available variables x and y). A 1-type for φ is a subset $t \subseteq \text{cl}(\varphi)$ that contains only formulas of the form $\psi(x)$ and such that $\varphi \wedge \exists x \wedge t(x)$ is satisfiable. For a model \mathfrak{A} of φ and $a \in A$, we use $\text{tp}_{\mathfrak{A}}(a)$ to denote the 1-type $\{\psi(x) \in \text{cl}(\varphi) \mid \mathfrak{A} \models \psi(a)\}$, assuming that φ is understood from the context. We say that the 1-type t is *realized* in \mathfrak{A} if there is an $a \in A$ with $\text{tp}_{\mathfrak{A}}(a) = t$. We are now ready to formulate our final characterization.

Theorem 3.3.7. *Let φ_1, φ_2 be GFO^2 -sentences and Σ a signature. Then $\varphi_1 \models_{\Sigma} \varphi_2$ iff for every regular forest model \mathfrak{A} of φ_1 that has finite outdegree and for every set $A_{\perp} \subseteq A$ with $A_{\perp} \cap \rho$ infinite for any infinite Σ -path ρ in \mathfrak{A} , there is a model \mathfrak{B} of φ_2 such that*

1. *for every $a \in A$, there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$;*
2. *for every 1-type t for φ_2 that is realized in \mathfrak{B} , there are $a \in A$ and $b \in B$ such that $\text{tp}_{\mathfrak{B}}(b) = t$ and $(\mathfrak{A}, a) \sim_{\Sigma^{\perp}}^{A_{\perp}} (\mathfrak{B}, b)$.*

Before we come to the proof of Theorem 3.3.7 we prove another characterization of Σ -entailment in GFO^2 . If \mathfrak{A} is a forest structure with $a, a' \in A$, then we write $a \prec a'$ iff a and a' are part of the same Σ -tree in \mathfrak{A} and a is an ancestor of a' (recall that a Σ -tree in a forest structure \mathfrak{A} is a maximal Σ -connected substructure of \mathfrak{A} and that we always assume a fixed root in trees within forest structures). For \mathfrak{A} and \mathfrak{B} structures and $a_{\perp} \in A$, an a_{\perp} -delimited $GFO^2(\Sigma)$ -bisimulation between \mathfrak{A} and \mathfrak{B} is defined like a $GFO^2(\Sigma)$ -bisimulation except that Conditions 2 and 3 are not required to hold when $a = a_{\perp}$. We indicate the existence of an a_{\perp} -delimited bisimulation by writing $(\mathfrak{A}, a) \sim_{\Sigma^{\perp}}^{a_{\perp}} (\mathfrak{B}, b)$. This requires $a_{\perp} \preceq a$. We now give a characterization of Σ -entailment using forest models in which we replace the bounded backward condition by an unbounded condition.

Theorem 3.3.8. *Let φ_1, φ_2 be GFO^2 -sentences and Σ a signature. Then $\varphi_1 \models_{\Sigma} \varphi_2$ iff for every regular forest model \mathfrak{A} of φ_1 that has finite outdegree there is a model \mathfrak{B} of φ_2 such that*

1. *for every $a \in A$ there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$*
2. *for every $b \in B$, one of the following holds:*
 - (a) *there is an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$;*
 - (b) *there are $a_{\perp}, a_0, a_1, \dots, a'_0, a'_1, \dots \in A$ such that $a_{\perp} \prec a_0 \prec a_1 \prec \dots$ and, for all $i \geq 0$, $a_i \prec a'_i$ and $(\mathfrak{A}, a'_i) \sim_{\Sigma^{\perp}}^{a_{\perp}} (\mathfrak{B}, b)$.*

Proof. We first observe that (i) every (successor-saturated/finite outdegree) structure \mathfrak{A} can be unfolded into a globally $GFO^2(\Sigma)$ -bisimilar (successor saturated/finite outdegree) forest model \mathfrak{B} , and consequently, (ii) every satisfiable GFO^2 formula is satisfiable in a regular forest model of finite outdegree. Then, using the proof of Theorem 3.3.6 and the observations (i) and (ii), one can easily prove the following variant of Theorem 3.3.6 based on forest models:

Fact 1. Let φ_1, φ_2 be GFO^2 -sentences and Σ a signature. Then $\varphi_1 \models_{\Sigma} \varphi_2$ iff for every regular forest model \mathfrak{A} of φ_1 that has finite outdegree there is a (successor saturated) forest model \mathfrak{B} of φ_2 such that

1. for every $a \in A$ there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$
2. for every $b \in B$ and every $k \geq 0$, there is an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$.

To show Theorem 3.3.8 it therefore suffices to show that for every regular forest model \mathfrak{A} of φ_1 that has finite outdegree and every successor-saturated forest model \mathfrak{B} of φ_2 , Condition 2 in Fact 1 is equivalent to Condition 2 of Theorem 3.3.8.

Thus, let \mathfrak{A} and \mathfrak{B} be as described. The interesting direction is to prove that if Condition 2 in Fact 1 holds then Condition 2 of Theorem 3.3.8 holds. Thus, assume

that Condition 2 in Fact 1 holds. Take $b \in B$. We may assume it is a root b of a Σ -tree in \mathfrak{B} . Then there are $a_0, a_1, \dots \in A$ such that for all k , $(\mathfrak{A}, a_k) \sim_{\Sigma}^k (\mathfrak{B}, b)$. If infinitely many of the a_i are identical, then there is an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^k (\mathfrak{B}, b)$ for all $k \geq 0$, thus $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$ by Lemma 3.3.3 and we are done. Therefore, assume that there are infinitely many distinct a_i . By ‘skipping’ elements in the sequence a_0, a_1, \dots , we can then achieve that the a_i are all distinct.

Two nodes $a, a' \in A$ are *downwards isomorphic*, written $a \sim_{\downarrow} a'$, if they are the roots of isomorphic subtrees. For a forest structure \mathfrak{A} , $a \in A$, and $i \geq 0$, we denote by $\mathfrak{A}|_a^{\uparrow i}$ the path structure obtained by restricting \mathfrak{A} to those elements that can be reached from a by traveling at most i steps towards the root of the tree in \mathfrak{A} that a is part of (including a itself). For $a, a' \in A$ and $i \geq 0$, we write $a \approx_i a'$ if there is an isomorphism ι from $\mathfrak{A}|_a^{\uparrow i}$ to $\mathfrak{A}|_{a'}^{\uparrow i}$ with $\iota(a) = a'$ such that $c \sim_{\downarrow} \iota(c)$ for all c . Since \mathfrak{A} is regular, \mathfrak{A} contains only finitely many equivalence classes for each \approx_i . By skipping a_i ’s, we can thus achieve that

$$(*) \quad a_i \approx_k a_j \text{ for all } i, k, j \text{ with } k \leq i \text{ and } j > i.$$

This also implies that each a_i is at least i steps away from the root of the tree in \mathfrak{A} that it is in (since there are infinitely many a_i , they must be unboundedly deep in their respective tree, and it remains to apply $(*)$). Let c_i denote the element of A reached from a_i by traveling i steps towards the root. Since \mathfrak{A} is regular, there must be an infinite subsequence $a_{\ell_0}, a_{\ell_1}, \dots$ of a_0, a_1, \dots such that $c_{\ell_i} \sim_{\downarrow} c_{\ell_j}$ for all i, j .

Choose some $a_{\perp} \in A$ with $a_{\perp} \sim_{\downarrow} c_{\ell_i}$ for all i (equivalently: for some i). We can assume w.l.o.g. that each a_{ℓ_i} is in the subtree rooted at a_{\perp} and that when traveling ℓ_i steps from a_{ℓ_i} towards the root of the subtree that a_{\perp} is in, then we reach exactly a_{\perp} .

Let \mathfrak{A}^* be the structure obtained in the limit of the neighborhoods $\mathfrak{A}|_{a_{\ell_0}}^0, \mathfrak{A}|_{a_{\ell_1}}^1, \dots$. That is, we start with the subtree of \mathfrak{A} rooted at a_{ℓ_0} , renaming a_{ℓ_0} to a^* , and then proceed as follows: after the i -th step, the constructed structure is isomorphic to the subtree of \mathfrak{A} rooted at a_{\perp} via an isomorphism that maps a^* to a_{ℓ_i} and the root to a_{\perp} ; by $(*)$, we can thus add a path of predecessor to the root of the structure constructed so far, and then add additional subtrees to the nodes on the path as additional successors, making sure that the obtained structure is isomorphic to the subtree of \mathfrak{A} rooted at a_{\perp} via an isomorphism that maps a^* to $a_{\ell_{i+1}}$ and the new root to a_{\perp} . By construction, $(\mathfrak{A}^*, a^*) \sim_{\Sigma}^k (\mathfrak{B}, b)$ for all $k \geq 0$ and thus Lemma 3.3.3 yields $(\mathfrak{A}^*, a^*) \sim_{\Sigma} (\mathfrak{B}, b)$.

Take some a_{ℓ_i} . We aim to show that $(\mathfrak{A}, a_{\ell_i}) \sim_{\Sigma}^{a_{\perp}} (\mathfrak{B}, b)$. Let c be the element reached from a^* in \mathfrak{A}^* by traveling ℓ_i steps upwards and recall that a_{\perp} is the element reached from a_{ℓ_i} in \mathfrak{A} by traveling ℓ_i steps upwards. By construction of \mathfrak{A}^* , we find an isomorphism from the subtree in \mathfrak{A}^* rooted at c to the subtree in \mathfrak{A} rooted at a_{\perp} that takes c to a_{\perp} and a^* to a_i . From $(\mathfrak{A}^*, a^*) \sim_{\Sigma} (\mathfrak{B}, b)$, we thus obtain the desired a_{\perp} -delimited Σ -bisimulation that witnesses $(\mathfrak{A}, a_i) \sim_{\Sigma}^{a_{\perp}} (\mathfrak{B}, b)$.

It remains to show the existence of the required elements a'_0, a'_1, \dots , that is, to show that there is a path through the subtree of \mathfrak{A} rooted at a_{\perp} such that each a_{ℓ_i} is either on the path or can be reached by branching off at a different point of the path. This can be done in the following straightforward way. Starting at a_{\perp} , we define the path step by step. In every step, there must be at least one successor which is the root of a subtree that contains infinitely many a_{ℓ_i} ’s since \mathfrak{A} has finite outdegree. We always proceed by choosing such a successor. This almost achieves the desired result, except that not all a_{ℓ_i} are reachable from a *distinct* node on the path by traveling downwards. However, there are infinitely many nodes on the path from which at least one a_{ℓ_i} can be reached by traveling downwards, so the problem can be cured by skipping a_{ℓ_i} ’s. \square

We are now in a position to prove Theorem 3.3.7. We require the following extended version of k -GFO²-bisimilarity which respects the successor relation in forest structures. Let \mathfrak{A} and \mathfrak{B} be forest structures, $a \in A$, and $b \in B$. The definition is by induction on $k \geq 0$. Then $(\mathfrak{A}, a) \sim_{\Sigma}^{0, \text{succ}} (\mathfrak{B}, b)$ iff $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$ and $(\mathfrak{A}, a) \sim_{\Sigma}^{k+1, \text{succ}} (\mathfrak{B}, b)$ iff $\text{at}_{\mathfrak{A}}^{\Sigma}(a) = \text{at}_{\mathfrak{B}}^{\Sigma}(b)$ and

1. for every $a' \neq a$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a')$ is guarded there exists $b' \neq b$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(a, a') = \text{at}_{\mathfrak{B}}^{\Sigma}(b, b')$ and b' is a successor of b in \mathfrak{B} iff a' is a successor of a in \mathfrak{A} and $(\mathfrak{A}, a') \sim_{\Sigma}^{k, \text{succ}} (\mathfrak{B}, b')$
2. for every $b' \neq b$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(b, b')$ is guarded there exists $a' \neq a$ such that $\text{at}_{\mathfrak{A}}^{\Sigma}(b, b') = \text{at}_{\mathfrak{B}}^{\Sigma}(a, a')$ and a' is a successor of a in \mathfrak{A} iff b' is a successor of b in \mathfrak{B} and $(\mathfrak{A}, a') \sim_{\Sigma}^{k, \text{succ}} (\mathfrak{B}, b')$.

Proof.[Theorem 3.3.7] (\Leftarrow) It suffices to show that for every $m > 0$ and every regular forest model \mathfrak{A} of φ_1 that has finite outdegree there exists a model \mathfrak{B} of φ_2 such that \mathfrak{A} and \mathfrak{B} are globally m -GFO²(Σ)-bisimilar. Assume $m > 0$ and a regular forest model \mathfrak{A} of φ_1 that has finite outdegree is given. Let m' be the maximum of m and the guarded quantifier depth of φ_2 . Then $f(m, \varphi_2)$ denotes the maximal number of nodes in any $\Sigma \cup \text{sig}(\varphi_2)$ -forest model \mathfrak{C} which are pairwise $\sim^{m', \text{succ}}$ -incomparable. Define $A_{\perp} \subseteq A$ on every Σ -tree with root r in \mathfrak{A} in such a way that $a \in A_{\perp}$ iff the distance between r and a is $kf(m, \varphi_2)$ for some $k \geq 0$. Let \mathfrak{B} be a forest shaped model of φ_2 satisfying the conditions of Theorem 3.3.7. One can easily modify \mathfrak{B} in such a way that in addition to the conditions given in the theorem

(*) every 1-type t for φ_2 that is realized in \mathfrak{B} is realized in the root of a Σ -tree in \mathfrak{B} and for every root r of a Σ -tree in \mathfrak{B} there exists $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^{A_{\perp}} (\mathfrak{B}, r)$.

To show (*) first pick for every $a \in A$ a $b \in B$ with $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$. Let S_1 be the set of b 's just picked and let \mathfrak{B}_1 be the disjoint union of the structures induced in \mathfrak{B} by the Σ -trees whose roots are in S_1 . Next pick for every 1-type t for φ_2 that is realized in \mathfrak{B} a $b \in B$ that realizes t . Let S_2 be the set of b 's just picked and let \mathfrak{B}_2 be the disjoint union of the structures induced in \mathfrak{B} by the Σ -trees whose roots are in S_2 . Finally, we add (recursively) witnesses for guarded existential quantifiers not involving binary predicates from Σ to the disjoint union \mathfrak{B}' of \mathfrak{B}_1 and \mathfrak{B}_2 . In detail, take for any b in \mathfrak{B}' its copy b' in \mathfrak{B} and assume c' in \mathfrak{B} is such that $\{R \mid (b', c') \in R^{\mathfrak{B}}$ or $(c', b') \in R^{\mathfrak{B}}\}$ is non-empty and contains no predicate in Σ . Then add to \mathfrak{B}' a copy of the Σ -tree in \mathfrak{B}' whose root c realizes the same 1-type for φ_2 as c' and connect c to b by adding for all binary predicates R the pair (b, c) to the extension of R if $(b', c') \in R^{\mathfrak{B}}$ and the pair (c, b) to the extension of R if $(c', b') \in R^{\mathfrak{B}}$. We apply this procedure recursively to the new structure (in a fair way) and obtain the desired structure as the limit of the resulting sequence of structures.

We now modify \mathfrak{B} in such a way that the resulting structure is still a model of φ_2 but in addition globally m -GFO²(Σ)-bisimilar to \mathfrak{A} . Consider the structure \mathfrak{B}_r induced by the Σ -tree with root r in \mathfrak{B} . If there exists an $a \in A$ with $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, r)$ then we do not modify \mathfrak{B}_r and set $\mathfrak{B}_r^u = \mathfrak{B}_r$. If no such a exists, then we modify \mathfrak{B}_r in such a way that every b in the resulting Σ -tree is m -GFO²(Σ)-bisimilar to some $a \in A$. Note that we only know that there exists $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^{A_{\perp}} (\mathfrak{B}, r)$. By construction of A_{\perp} this implies that (\mathfrak{A}, a) and (\mathfrak{B}, r) are $f(m, \varphi_2)$ -GFO²(Σ)-bisimilar. Thus, it suffices to modify \mathfrak{B}_r in such a way that every node b in the Σ -tree becomes m -GFO²(Σ)-bisimilar to some b' in the original \mathfrak{B}_r with distance $\leq f(m, \varphi_2) - m$

from r . To ensure that φ_2 is still satisfied we make sure that the following stronger condition holds: every node b in the Σ -tree rooted at r is m' -GFO²-bisimilar to some b' in the original \mathfrak{B}_r with distance $\leq f(m, \varphi_2) - m'$ from r . The construction is by a standard pumping argument. For $a, b \in B$ we say that a *blocks* b if $a \prec b$ and $(\mathfrak{B}, a) \sim^{m', \text{succ}} (\mathfrak{B}, b)$ and there is no $b' \prec b$ such that there is an a' with $a' \prec b'$ and $(\mathfrak{B}, a') \sim^{m', \text{succ}} (\mathfrak{B}, b')$. The universe B_r^u of \mathfrak{B}_r^u is the set of words $a_0 \cdots a_n$ with a_0, \dots, a_n in \mathfrak{B}_r^u and $a_0 = r$ such that either a_{i+1} is a successor of a_i or there is a successor b_{i+1} of a_i such that a_{i+1} blocks b_{i+1} . Let $\text{tail}(a_0 \dots a_n) = a_n$. For every unary R and $w \in A_r^u$ we set $w \in R^{\mathfrak{A}^u}$ if $\text{tail}(w) \in R^{\mathfrak{A}}$ and for every binary R we set for $w \in A_r^u$: $(w, w) \in R^{\mathfrak{A}^u}$ if $(\text{tail}(w), \text{tail}(w)) \in R^{\mathfrak{A}}$ and for $wb \in A_r^u$:

- $(w, wb) \in R^{\mathfrak{A}^u}$ if $(\text{tail}(w), b) \in R^{\mathfrak{A}}$ or there is an a such that b blocks a and $(\text{tail}(w), a) \in R^{\mathfrak{A}}$;
- $(wb, w) \in R^{\mathfrak{A}^u}$ if $(b, \text{tail}(w)) \in R^{\mathfrak{A}}$ or there is an a such that b blocks a and $(a, \text{tail}(w)) \in R^{\mathfrak{A}}$.

We now replace \mathfrak{B}_r by \mathfrak{B}_r^u in \mathfrak{B} . In more detail, take the disjoint union \mathfrak{B}^d of all \mathfrak{B}_r^u , r the root of a Σ -tree in \mathfrak{B} . Then add (recursively) witnesses for guarded existential quantifiers not involving binary predicates from Σ to \mathfrak{B}^d : take for any w in \mathfrak{B}_r^u and any 1-type t for φ_2 that is realized in some node c in \mathfrak{B} such that $\{R \mid (\text{tail}(w), c) \in R^{\mathfrak{B}}$ or $(c, \text{tail}(w)) \in R^{\mathfrak{B}}\}$ is non-empty and contains no predicate in Σ the root r' of a structure $\mathfrak{B}_{r'}^u$ such that r' realizes t in $\mathfrak{B}_{r'}^u$. Then add to \mathfrak{B}^d a new copy of $\mathfrak{B}_{r'}^u$ and connect r' to b by adding for any binary predicate R the pair (r, r') to $R^{\mathfrak{B}^d}$ if $(\text{tail}(w), c) \in R^{\mathfrak{B}}$ and the pair (r', r) to $R^{\mathfrak{B}^d}$ if $(c, \text{tail}(w)) \in R^{\mathfrak{B}}$. We apply this procedure recursively to the new structure (in a fair way) and obtain the desired structure \mathfrak{B}' as the limit of the resulting sequence of structures.

(\Rightarrow) Assume that $\varphi_1 \models_{\Sigma} \varphi_2$. Let \mathfrak{A} be a regular forest model \mathfrak{A} of φ_1 that has finite outdegree and let $A_{\perp} \subseteq A$ be such that $A_{\perp} \cap \rho$ is infinite for any maximal infinite Σ -path ρ in \mathfrak{A} . By Theorem 3.3.8, there is a model \mathfrak{B} of φ_2 such that

1. for every $a \in A$ there is a $b \in B$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$
2. for every $b \in B$, one of the following holds:
 - (a) there is an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$;
 - (b) there are $a_{\perp}, a_0, a_1, \dots, a'_0, a'_1, \dots \in A$ such that $a_{\perp} \prec a_0 \prec a_1 \prec \dots$ and, for all $i \geq 0$, $a_i \prec a'_i$ and $(\mathfrak{A}, a'_i) \sim_{\Sigma}^{a_{\perp}} (\mathfrak{B}, b)$.

Let t be a 1-type for φ_2 realized by some $b \in B$. We have to find an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma}^{A_{\perp}} (\mathfrak{B}, b)$. If there is an $a \in A$ such that $(\mathfrak{A}, a) \sim_{\Sigma} (\mathfrak{B}, b)$ then we are done as $(\mathfrak{A}, a) \sim_{\Sigma}^{A_{\perp}} (\mathfrak{B}, b)$ follows. Otherwise there are $a_{\perp}, a_0, a_1, \dots, a'_0, a'_1, \dots \in A$ such that $a_{\perp} \prec a_0 \prec a_1 \prec \dots$ and, for all $i \geq 0$, $a_i \prec a'_i$ and $(\mathfrak{A}, a'_i) \sim_{\Sigma}^{a_{\perp}} (\mathfrak{B}, b)$. Then let ρ be a Σ -path containing $a_{\perp}, a_0, a_1, \dots$. $A_{\perp} \cap \rho$ is infinite and so we can choose an a'_i such that there are at least two elements of A_{\perp} on the path from a_{\perp} to a'_i . It follows from the definition of $\sim_{\Sigma}^{A_{\perp}}$ that $(\mathfrak{A}, a'_i) \sim_{\Sigma}^{A_{\perp}} (\mathfrak{B}, b)$, as required. \square

Regularity and finite outdegree are used in the proof of Theorem 3.3.7, but it follows from the automata constructions in Section 3.4.2 that the theorem is still correct when these qualifications are dropped.

3.4 Decidability and Complexity

We show that Σ -entailment in GFO² is decidable and 2EXPTIME-complete, and thus so are conservative extensions and Σ -inseparability. The upper bound is based on Theorem 3.3.7 and uses alternating parity automata on infinite trees. Since Theorem 3.3.7 does not provide us with an obvious upper bound on the outdegree of the involved tree models, we use alternating tree automata which can deal with trees of any finite outdegree, similar to the ones introduced by Wilke [Wil01], but with the capability to move both downwards and upwards in the tree.

3.4.1 2ATAs and their Emptiness Problem

A *tree* is a non-empty (and potentially infinite) set of words $T \subseteq (\mathbb{N} \setminus 0)^*$ closed under prefixes. We generally assume that trees are finitely branching, that is, for every $w \in T$, the set $\{i \mid w \cdot i \in T\}$ is finite. For any $w \in (\mathbb{N} \setminus 0)^*$, as a convention we set $w \cdot 0 := w$. If $w = n_0 n_1 \cdots n_k$, we additionally set $w \cdot -1 := n_0 \cdots n_{k-1}$. For an alphabet Θ , a Θ -labeled tree is a pair (T, L) with T a tree and $L : T \rightarrow \Theta$ a node labeling function.

A *two-way alternating tree automata (2ATA)* is a tuple $\mathcal{A} = (Q, \Theta, q_0, \delta, \Omega)$ where Q is a finite set of *states*, Θ is the *input alphabet*, $q_0 \in Q$ is the *initial state*, δ is a *transition function* as specified below, and $\Omega : Q \rightarrow \mathbb{N}$ is a *priority function*, which assigns a priority to each state. The transition function maps a state q and some input letter $\theta \in \Theta$ to a *transition condition* $\delta(q, \theta)$ which is a positive Boolean formula over the truth constants *true* and *false* and transitions of the form $q, \langle - \rangle q, [-]q, \diamond q, \square q$ where $q \in Q$. The automaton runs on Θ -labeled trees. Informally, the transition q expresses that a copy of the automaton is sent to the current node in state q , $\langle - \rangle q$ means that a copy is sent in state q to the predecessor node, which is then required to exist, $[-]q$ means the same except that the predecessor node is not required to exist, $\diamond q$ means that a copy is sent in state q to some successor, and $\square q$ that a copy is sent in state q to all successors. The semantics is defined in terms of runs. Let $\mathcal{A} = (Q, \Theta, q_0, \delta, \Omega)$ be a 2ATA and (T, L) a Θ -labeled tree. A *run for \mathcal{A} on (T, L)* is a $T \times Q$ -labeled tree (T_r, r) such that:

- $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$;
- For all $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, L(x)) = \varphi$, there is an assignment v of truth values to the transitions in φ such that v satisfies φ and:
 - if $v(p) = 1$, then $r(y') = (x, p)$ for some successor y' of y in T_r ;
 - if $v(\langle - \rangle p) = 1$, then $x \neq \varepsilon$ and there is a successor y' of y in T_r with $r(y') = (x \cdot -1, p)$;
 - if $v([-]p) = 1$, then $x = \varepsilon$ or there is a successor y' of y in T_r such that $r(y') = (x \cdot -1, p)$;
 - if $v(\diamond p) = 1$, then there is a successor x' of x in T and a successor y' of y in T_r such that $r(y') = (x', p)$;
 - if $v(\square p) = 1$, then for every successor x' of x in T , there is a successor y' of y in T_r such that $r(y') = (x', p)$.

Let $\gamma = i_0 i_1 \cdots$ be an infinite path in T_r and denote, for all $j \geq 0$, with q_j the state such that $r(i_0 \cdots i_j) = (x, q_j)$. The path γ is *accepting* if the largest number m such that $\Omega(q_j) = m$ for infinitely many j is even. A run (T_r, r) is *accepting*, if all infinite

paths in T_r are accepting. Finally, a tree is accepted if there is some accepting run for it.

We use $L(\mathcal{A})$ to denote the set of all Θ -labeled trees accepted by \mathcal{A} . It is standard to verify that 2ATAs are closed under complementation and intersection. We now show that the emptiness problem for 2ATAs can be solved in time exponential in the number of states. For proving this, we reduce it to the emptiness problem of the standard two-way alternating tree automata over trees of fixed outdegree [Var98].

We start by introducing strategy trees similar to [Var98, Section 4]. A *strategy tree* for a 2ATA \mathcal{A} is a tree (T, τ) where τ labels every node in T with a subset $\tau(x) \subseteq 2^{Q \times (\mathbb{N} \cup \{-1\}) \times Q}$, that is, with a graph with nodes from Q and edges labeled with natural numbers or -1 . Intuitively, $(q, i, p) \in \tau(x)$ expresses that, if we reached node x in state q , then we should send a copy of the automaton in state p to $x \cdot i$. For each label ζ , we define $\mathbf{state}(\zeta) = \{q \mid (q, i, q') \in \zeta\}$, that is, the set of sources in the graph ζ . A strategy tree is *on an input tree* (T', L) if $T = T'$, $q_0 \in \mathbf{state}(\tau(\varepsilon))$, and for every $x \in T$, the following conditions are satisfied:

1. if $(q, i, p) \in \tau(x)$, then $x \cdot i \in T$;
2. if $(q, i, p) \in \tau(x)$, then $p \in \mathbf{state}(\tau(x \cdot i))$;
3. if $q \in \mathbf{state}(\tau(x))$, then the truth assignment $v_{q,x}$ defined below satisfies $\delta(q, L(x))$:
 - (a) $v_{q,x}(p) = 1$ iff $(q, 0, p) \in \tau(x)$;
 - (b) $v_{q,x}(\langle - \rangle p) = 1$ iff $(q, -1, p) \in \tau(x)$;
 - (c) $v_{q,x}([-]p) = 1$ iff $x = \varepsilon$ or $(q, -1, p) \in \tau(x)$;
 - (d) $v_{q,x}(\diamond p) = 1$ iff there is some i with $(q, i, p) \in \tau(x)$;
 - (e) $v_{q,x}(\square p) = 1$ iff $(q, i, p) \in \tau(x)$, for all $x \cdot i \in T$;

A *path* β in a strategy tree (T, τ) is a sequence $\beta = (u_1, q_1)(u_2, q_2) \cdots$ of pairs from $T \times Q$ such that for all $\ell > 0$, there is some i such that $(q_\ell, i, q_{\ell+1}) \in \tau(u_\ell)$ and $u_{\ell+1} = u_\ell \cdot i$. Thus, β is obtained by following moves prescribed by the strategy tree. We say that β is *accepting* if the largest number m such that $\Omega(q_i) = m$, for infinitely many i , is even. A strategy tree (T, τ) is *accepting* if all infinite paths in (T, τ) are accepting.

Lemma 3.4.1. *A 2ATA accepts a Θ -labeled tree (T, L) iff there is an accepting strategy tree for \mathcal{A} on (T, L) .*

Proof. The “if”-direction is immediate: just read off an accepting run from the accepting strategy tree.

For the “only if”-direction, we observe that acceptance of an input tree can be defined in terms of a parity game between Player 1 (trying to show that the input is accepted) and Player 2 (trying to challenge that). The initial configuration is (ε, q_0) and Player 1 begins. Consider a configuration (x, q) . Player 1 chooses a satisfying truth assignment v of $\delta(q, L(x))$. Player 2 chooses a transition α with $v(\alpha) = 1$ and the next configuration is determined as follows:

- if $\alpha = p$, then the next configuration is (x, p) ,
- if $\alpha = \langle - \rangle p$, then the next configuration is $(x \cdot -1, p)$ unless $x = \varepsilon$ in which case Player 1 loses immediately.

- if $\alpha = [-]p$, then the next configuration is $(x \cdot -1, p)$ unless $x = \varepsilon$ in which case Player 2 loses immediately;
- if $\alpha = \diamond p$, then Player 1 chooses some i with $x \cdot i \in T$ (and loses if no such i exists) and the next configuration is $(x \cdot i, p)$;
- if $\alpha = \square p$, then Player 2 chooses some i with $x \cdot i \in T$ (and loses if no such i exists) and the next configuration is $(x \cdot i, p)$.

Player 1 wins an infinite play $(x_0, q_0)(x_1, q_1) \cdots$ if the largest number m such that $\Omega(q_i) = m$, for infinitely many i , is even. It is not difficult to see that Player 1 has a winning strategy on an input tree iff \mathcal{A} accepts the input tree. Observe now that the defined game is a parity game and thus Player 1 has a winning strategy iff she has a *memoryless* winning strategy [EJ91]. It remains to observe that a memoryless winning strategy is nothing else than an accepting strategy tree. \square

Based on the previous lemma, we show that, if $L(\mathcal{A})$ is not empty, then it contains a tree of small outdegree.

Lemma 3.4.2. *If $L(\mathcal{A}) \neq \emptyset$, then there is a $(T, L) \in L(\mathcal{A})$ such that the outdegree of T is bounded by the number of states in \mathcal{A} .*

Proof. Let (T, L) be a Θ -labeled tree and τ an accepting strategy tree on T . We construct a tree $T' \subseteq T$ and an accepting strategy tree τ' on (T', L') where L' is the restriction of L to T' . Start with $T' = \{\varepsilon\}$ and τ' the empty mapping. Then exhaustively repeat the following step. Select an $x \in T'$ with $\tau'(x)$ undefined, in a fair way. Then construct $\tau'(x)$ as follows:

1. for every $(q, i, p) \in \tau(x)$ with $i \in \{-1, 0\}$, include (q, i, p) in $\tau'(x)$;
2. for every $p \in Q$, choose an i such that $(q, i, p) \in \tau(x)$ for some q , if existant. Then add $x \cdot i$ to T' and include (q', i, p) in $\tau'(x)$ for all $(q', j, p) \in \tau(x)$;
3. further include (q, i, p) in $\tau'(x)$ whenever $x \cdot i \in T'$ and $(q, j, p) \in \tau(x)$ for all j with $x \cdot j \in T$.

Clearly, T' has the desired outdegree. It remains to show that τ' is an accepting strategy tree on (T', L') . Observe that the following properties hold for all $x \in T'$, and $p, q \in Q$:

- (i) $(q, i, p) \in \tau(x)$ iff $(q, i, p) \in \tau'(x)$, for $i \in \{-1, 0\}$;
- (ii) $(q, i, p) \in \tau(x)$ for some $i > 0$ with $x \cdot i \in T$ iff $(q, j, p) \in \tau'(x)$ for some $j > 0$ with $x \cdot j \in T$.

Observe that we have $q_0 \in \text{state}(\tau'(\varepsilon))$, by Points (i) and (ii) and since $q_0 \in \text{state}(\tau(\varepsilon))$. It can be verified that Conditions 1 and 2 of a strategy tree being on an input tree are satisfied due to the construction of T' and τ' . For Condition 3, take any $x \in T'$ and $q \in \text{state}(\tau'(x))$. As $q \in \text{state}(\tau(x))$, we know that the truth assignment $v_{q,x}$ defined for τ satisfies $\delta(q, V(x))$. Let $v'_{q,x}$ be the truth assignment for τ' , q, x . It suffices to show that, for all transitions α , $v_{q,x}(\alpha) = 1$ implies $v'_{q,x}(\alpha) = 1$. By Point (i), this is the case for transitions of the form $p, \langle - \rangle p, [-]p$. For $\alpha = \diamond p$, we know that there is some i, p with $(q, i, p) \in \tau(x)$. By Point (ii), we know that $(q, i', p) \in \tau'(x)$ for some i' with $x \cdot i' \in T'$, and thus, $v'_{q,x}(\alpha) = 1$. For $\alpha = \square p$, we know that $(q, i, p) \in \tau(x)$ for all i with $x \cdot i \in T$. By construction of τ' , it follows that $(q, i, p) \in \tau'(x)$ for all i with $x \cdot i \in T'$, as required.

We finally argue that τ' is also accepting. Let $\beta = (u_1, q_1)(u_2, q_2) \cdots$ be an infinite path in (T', τ') . We construct an infinite path $\beta' = (u'_1, q_1)(u'_2, q_2)(u'_3, q_3) \cdots$ in (T, τ) as follows:

- $u'_1 = u_1$;
- if $u_{i+1} = u_i \cdot \ell$ with $\ell \in \{-1, 0\}$, then $u'_{i+1} = u'_i \cdot \ell$.
- if $u_{i+1} = u_i \cdot \ell$ for some $\ell \geq 0$ with $(q_i, \ell, q_{i+1}) \in \tau'(x)$, then, by Point (ii), there is some ℓ' with $(q_i, \ell', q_{i+1}) \in \tau(x)$ and $x \cdot \ell' \in T'$. Set $u'_{i+1} = u_i \cdot \ell'$.

Since every infinite path in (T, τ) is accepting, so is β' , and thus β . \square

We are now ready to reduce the emptiness problem of 2ATAs to the emptiness of alternating automata running on trees of fixed outdegree [Var98], which we recall here. A tree T is k -ary if every node has exactly k successors. A *two-way alternating tree automaton over k -ary trees* ($2ATA^k$) that are Θ -labeled is a tuple $\mathcal{A} = (Q, \Theta, q_0, \delta, \Omega)$ where Q is a finite set of *states*, Θ is the *input alphabet*, $q_0 \in Q$ is an *initial state*, δ is the *transition function*, and $\Omega : Q \rightarrow \mathbb{N}$ is a *priority function*. The transition function maps a state q and some input letter θ to a *transition condition* $\delta(q, \theta)$, which is a positive Boolean formula over the truth constants *true*, *false*, and transitions of the form $(i, q) \in [k] \times Q$ where $[k] = \{-1, 0, \dots, k\}$. A *run* of \mathcal{A} on a Θ -labeled tree (T, L) is a $T \times Q$ -labeled tree (T_r, r) such that

1. $r(\varepsilon) = (\varepsilon, q_0)$;
2. for all $x \in T_r$, $r(x) = (w, q)$, and $\delta(q, \tau(w)) = \varphi$, there is a (possibly empty) set $\mathcal{S} = \{(m_1, q_1), \dots, (m_n, q_n)\} \subseteq [k] \times Q$ such that \mathcal{S} satisfies φ and for $1 \leq i \leq n$, we have $x \cdot i \in T_r$, $w \cdot m_i$ is defined, and $\tau_r(x \cdot i) = (w \cdot m_i, q_i)$.

Accepting runs and accepted trees are defined as for 2ATAs. The emptiness problem for $2ATA^k$ s can be solved in time exponential in the number of states [Var98].

Theorem 3.4.3. *The emptiness problem for 2ATAs can be solved in time exponential in the number of states.*

Proof. Let $\mathcal{A} = (Q, \Theta, q_0, \delta, \Omega)$ be a 2ATA with n states. We transform \mathcal{A} into a $2ATA^n$ $\mathcal{A}' = (Q', \Theta', q'_0, \delta', \Omega)$, running over n -ary Θ' -labeled trees, where $Q' = Q \uplus \{q'_0, q', q_r\}$ and $\Theta' = \Theta \times \{0, 1\}$. The extended alphabet and the extra states q'_0, q', q_r are used to simulate transitions of the form $[-]p$. We make sure that the additional component labels the root node with 1 and all other nodes with 0, and based on this use q_r to check whether we are at the root of the input tree.

Formally, we proceed as follows. For all $q \in Q$, $\theta \in \Theta$, and $b \in \{0, 1\}$ obtain $\delta'(q, (\theta, b))$ from $\delta(q, \theta)$ by replacing q with $(0, q)$, $\langle - \rangle q$ with $(-1, q)$, $[-]q$ with $(0, q_r) \vee (-1, q)$, $\diamond q$ with $\bigvee_{i=1}^n (i, q)$, and $\square q$ with $\bigwedge_{i=1}^n (i, q)$. To enforce the intended labeling in the second component and the correct behaviour for q_r , we set:

$$\begin{aligned} \delta'(q'_0, (\theta, b)) &= \begin{cases} \text{false} & \text{if } b = 0 \\ (0, q_0) \wedge \bigwedge_{i=1}^k (i, q') & \text{otherwise} \end{cases} \\ \delta'(q', (\theta, b)) &= \begin{cases} \bigwedge_{i=1}^k (i, q') & \text{if } b = 0 \\ \text{false} & \text{otherwise} \end{cases} \\ \delta'(q_r, (\theta, b)) &= \begin{cases} \text{true} & \text{if } b = 1 \\ \text{false} & \text{otherwise} \end{cases} \end{aligned}$$

Using Lemma 4.4.6 it is straightforward to verify that $L(\mathcal{A}) \neq \emptyset$ iff $L(\mathcal{A}') \neq \emptyset$. Since the translation can be done in polynomial time and the emptiness problem for 2ATA^k s is in EXPTIME , also emptiness for 2ATAs is in EXPTIME . \square

3.4.2 Upper Bound

We aim to show that given two GFO^2 -sentences φ_1 and φ_2 and a signature Σ , one can construct a 2ATA \mathcal{A} such that $L(\mathcal{A}) = \emptyset$ iff $\varphi_1 \models_{\text{GFO}^2(\Sigma)} \varphi_2$. The number of states of the 2ATA \mathcal{A} is polynomial in the size of φ_1 and exponential in the size of φ_2 , which yields the desired 2EXPTIME upper bounds.

Let φ_1 , φ_2 , and Σ be given. Since the logics we are concerned with have Craig interpolation, we can assume w.l.o.g. that $\Sigma \subseteq \text{sig}(\varphi_1)$. With Θ , we denote the set of all pairs (τ, M) where τ is an atomic 2-type for $\text{sig}(\varphi_1)$ and $M \in \{0, 1\}$. For $p = (\tau, M) \in \Theta$, we use p^1 to denote τ and p^2 to denote M . A Θ -labeled tree (T, L) represents a forest structure $\mathfrak{A}_{(T,L)}$ with universe $A_{(T,L)} = T$ and where $w \in A^{\mathfrak{A}_{(T,L)}}$ if $A(y) \in L(w)$ and $(w, w') \in R^{\mathfrak{A}_{(T,L)}}$ if one of the following conditions is satisfied: (1) $w = w'$ and $Ryy \in L(w)^1$; (2) w' is a successor of w and $Rxy \in L(w')^1$; (3) w is a successor of w' and $Ryx \in L(w)^1$. Thus, the atoms in a node label that involve only the variable y describe the current node, the atoms that involve both variables x and y describe the connection between the predecessor and the current node, and the atoms that involve only the variable x are ignored. The M -components of node labels are used to represent a set of markers $A_\perp = \{w \in A_{(T,L)} \mid L(w)^2 = 1\}$. It is easy to see that, conversely, for every tree structure \mathfrak{A} over Σ , there is a Θ -labeled tree (T, L) such that $\mathfrak{A}_{(T,L)} = \mathfrak{A}$.

To obtain the desired 2ATA \mathcal{A} , we construct three 2ATAs $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ and then define \mathcal{A} so that it accepts $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)} \cap L(\mathcal{A}_3)$. The 2ATA \mathcal{A}_3 only makes sure that the set $A_\perp \subseteq A_{(T,L)}$ is such that for any infinite Σ -path ρ , $A_\perp \cap \rho$ is infinite (as required by Theorem 3.3.7), we omit details. We construct \mathcal{A}_1 so that it accepts a Θ -labeled tree (T, L) iff $\mathfrak{A}_{(T,L)}$ is a model of φ_1 . The number of states of \mathcal{A}_1 is polynomial in the size of φ_1 and independent of φ_2 .

Lemma 3.4.4. *Let φ_1 be a GFO^2 -sentence. There is a 2ATA \mathcal{A}_1 that accepts a Θ -labeled tree (T, L) iff $\mathfrak{A}_{(T,L)}$ is a model of φ_1 .*

We assume that in all subformulas of φ_1 of the form $\exists \mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{x}, \mathbf{y}))$ and $\forall \mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \rightarrow \varphi(\mathbf{x}, \mathbf{y}))$, \mathbf{y} consists of exactly one variable and $\alpha(\mathbf{x}, \mathbf{y})$ is a relational atom with two variables or an equality atom. This can be done w.l.o.g. because each sentence $\exists xy\varphi(x, y)$ can be rewritten into $\exists x(x = x \wedge \exists y\varphi(x, y))$, each sentence $\exists x(\alpha(x) \wedge \varphi(x))$ with α a relational atom can be rewritten into $\exists x(x = x \wedge \alpha(x) \wedge \varphi(x))$, and likewise for universal quantifiers. We further assume that φ_1 has no subformulas of the form $\exists x(x = y \wedge \varphi(x, y))$ with $x \neq y$; such formulas are equivalent to $\varphi[y/x]$, that is, the result of replacing in φ all occurrences of x with y . The result of these assumptions is that each formula $\exists \mathbf{y}(\alpha(\mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{x}, \mathbf{y}))$ takes the form $\exists x(x = x \wedge \psi(x))$ or $\exists x\psi(x, y)$, and likewise for universally quantified formulas. We define $\mathcal{A}_1 = (Q_1, \Theta, q_{\varphi_1}, \delta_1, \Omega_1)$ where

$$Q_1 = \{q_{\varphi(x)} \mid \varphi(x) \in \text{cl}(\varphi_1)\} \cup \{q_{\varphi(x,y)}, q_{\varphi(\underline{x},y)} \mid \varphi(x, y) \in \text{cl}(\varphi_1)\}$$

and Ω_1 assigns two to all states except those of the form $q_{\exists x(x=x \wedge \psi(x))}$, to which it assigns one. The underlining in states of the form $q_{\varphi(x,y)}$ and $q_{\varphi(\underline{x},y)}$ serves as a marking of the variable that is bound to the tree node to which the state is assigned. We define

the transition function δ_1 as follows, for each $\sigma = (\tau, M)$:

$$\begin{aligned} \delta_1(q_{Az}, \sigma) &= \begin{cases} \text{true} & \text{if } Ay \in \tau \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_1(q_{\neg Az}, \sigma) &= \begin{cases} \text{true} & \text{if } Ay \notin \tau \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_1(q_{\varphi(z) \circ \psi(z)}, \sigma) &= q_{\varphi(z)} \circ q_{\psi(z)} \\ \delta_1(q_{\exists z(z=z \wedge \psi(z))}) &= q_{\psi(z)} \vee \langle -1 \rangle q_{\exists z(z=z \wedge \psi(z))} \vee \diamond q_{\exists z(z=z \wedge \psi(z))} \\ \delta_1(q_{\forall z(z=z \rightarrow \psi(z))}) &= q_{\psi(z)} \wedge [-] q_{\forall z(z=z \rightarrow \psi(z))} \wedge \square q_{\forall z(z=z \rightarrow \psi(z))} \\ \delta_1(q_{\exists z' \varphi(z, z')}, \sigma) &= \diamond q_{\varphi(z, z')} \vee q_{\varphi(z', z)} \\ \delta_1(q_{\forall z' \varphi(z, z')}, \sigma) &= \square q_{\varphi(z, z')} \wedge q_{\varphi(z', z)} \\ \\ \delta_1(q_{Rz z'}, \sigma) &= \begin{cases} \text{true} & \text{if } Rxy \in \tau \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_1(q_{Rz z'}, \sigma) &= \begin{cases} \text{true} & \text{if } Rxy \in \tau \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_1(q_{\neg Rz z'}, \sigma) &= \begin{cases} \text{true} & \text{if } Rxy \notin \tau \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_1(q_{\neg Rz z'}, \sigma) &= \begin{cases} \text{true} & \text{if } Rxy \notin \tau \\ \text{false} & \text{otherwise} \end{cases} \\ \\ \delta_1(q_{\varphi(z, z') \circ \psi(z, z')}, \sigma) &= q_{\varphi(z, z')} \circ q_{\psi(z, z')} \\ \delta_1(q_{\varphi(z, z') \circ \psi(z)}, \sigma) &= q_{\varphi(z, z')} \circ \langle -1 \rangle q_{\psi(z)} \\ \delta_1(q_{\varphi(z, z') \circ \psi(z')}, \sigma) &= q_{\varphi(z, z')} \circ q_{\psi(z')} \\ \delta_1(q_{\varphi(z) \circ \psi(z')}, \sigma) &= \langle -1 \rangle q_{\varphi(z)} \circ q_{\psi(z')} \end{aligned}$$

where σ ranges over Θ , z, z' range over $\{x, y\}$, and \circ ranges over $\{\wedge, \vee\}$. With $\varphi(z', z)$, we mean the result of exchanging in $\varphi(z, z')$ the variables z and z' , and $\overline{\varphi(z, z')}$ denotes the negation normal form of the negation of $\varphi(z, z')$.

We now construct \mathcal{A}_2 , the most interesting automaton.

Lemma 3.4.5. *There is a 2ATA \mathcal{A}_2 that accepts a Θ -labeled tree (T, L) iff there is a model \mathfrak{B} of φ_2 s.t. Conditions 1 and 2 from Theorem 3.3.7 are satisfied when \mathfrak{A} is replaced with $\mathfrak{A}_{(T, L)}$.*

The general idea of the construction of \mathcal{A}_2 is to check the existence of the desired model \mathfrak{B} of φ_2 by verifying that there is a set of 1-types for φ_2 from which \mathfrak{B} can be assembled, represented via the states that occur in a successful run. Before we can give details, we introduce some preliminaries.

A *0-type* s for φ_2 is a maximal set of sentences $\psi() \in \text{cl}(\varphi_2)$ such that $\varphi_2 \wedge s$ is satisfiable. A *2-type* λ for φ_2 is a maximal set of formulas $\psi(x, y) \in \text{cl}(\varphi_2)$ that contains $\neg(x = y)$ and such that $\varphi_2 \wedge \exists xy \lambda(x, y)$ is satisfiable. If a 2-type λ contains the atom Rxy or Ryx for at least one binary predicate R , then it is *guarded*. If additionally $R \in \Sigma$, then it is Σ -*guarded*. Note that each 1-type contains a (unique) 0-type and each 2-type contains two (unique) 1-types. Formally, we use λ_x to denote the 1-type obtained by restricting the 2-type λ to the formulas that do not use the variable y , and likewise for λ_y and the variable x . We use TP_n to denote the set of n -types for φ_2 , $n \in \{0, 1, 2\}$. For $t \in \text{TP}_1$ and a $\lambda \in \text{TP}_2$, we say that λ is *compatible with* t and write

$t \approx \lambda$ if the sentence $\varphi_2 \wedge \exists xy(t(x) \wedge \lambda(x, y))$ is satisfiable; for $t \in \text{TP}_1$ and $T \subseteq \text{TP}_2$ a set of guarded 2-types, we say that T is a *neighborhood for t* and write $t \approx T$ if the sentence

$$\varphi_2 \wedge \exists x(t(x) \wedge \bigwedge_{\lambda \in T} \exists y \lambda(x, y) \wedge \forall y \bigvee_{R \in \text{sig}(\varphi_2)} ((Rxy \vee Ryx) \rightarrow \bigvee_{\lambda \in T} \lambda(x, y)))$$

is satisfiable. Note that each of the mentioned sentences is formulated in GFO^2 and at most single exponential in size (in the size of φ_1 and φ_2), thus satisfiability can be decided in 2EXPTIME .

To build the automaton \mathcal{A}_2 from Lemma 3.4.5, set $\mathcal{A}_2 = (Q_2, \Theta, q_0, \delta_2, \Omega_2)$ where Q_2 is

$$\begin{aligned} & \{q_0, q_\perp\} \cup \text{TP}_0 \cup \{t, t^\sharp, t_\uparrow, t_\downarrow, t_\&, t^i, t_\uparrow^i, t_\downarrow^i \mid t \in \text{TP}_1, i \in \{0, 1\}\} \cup \\ & \{\lambda, \lambda_\uparrow, \lambda^i, \lambda_\uparrow^i \mid \lambda \in \text{TP}_2, i \in \{0, 1\}\}, \end{aligned}$$

Ω_2 assigns two to all states except for those of the form t^\sharp , to which it assigns one.

The automaton begins by choosing the 0-type s realized in the forest model \mathfrak{B} of φ_2 whose existence it aims to verify. For every $\exists x\varphi(x) \in s$, it then chooses a 1-type t in which $\varphi(x)$ is realized in \mathfrak{B} and sends off a copy of itself to find a node where t is realized. To satisfy Condition 1 of Theorem 3.3.7, at each node it further chooses a 1-type that is compatible with s , to be realized at that node. This is implemented by the following transitions:

$$\begin{aligned} \delta_2(q_0, \sigma) &= \bigvee_{s \in \text{TP}_0} (s \wedge \bigwedge_{\exists x \varphi(x) \in s} \bigvee_{\substack{t \in \text{TP}_1 \\ s \cup \{\varphi(x)\} \subseteq t}} t^\sharp) \\ \delta_2(s, \sigma) &= \Box s \wedge \bigvee_{t \in \text{TP}_1, s \subseteq t} t \\ \delta_2(t^\sharp, \sigma) &= \langle -1 \rangle t^\sharp \vee \Diamond t^\sharp \vee t^0 \end{aligned}$$

where s ranges over TP_0 . When a state of the form t is assigned to a node w , this is an obligation to prove that there is a $\text{GFO}^2(\Sigma)$ -bisimulation between the element w in $\mathfrak{A}_{(T,L)}$ and an element b of type t in \mathfrak{B} . A state of the form t^0 represents the obligation to verify that there is an A_\perp -delimited $\text{GFO}^2(\Sigma)$ -bisimulation between w and an element of type t in \mathfrak{B} . We first verify that the former obligations are satisfied. This requires to follow all successors of w and to guess types of successors of b to be mapped there, satisfying the back condition of bisimulations. We also need to guess successors of b in \mathfrak{B} (represented as a neighborhood for t) to satisfy the existential demands of t and then select successors of a to which they are mapped, satisfying the “back” condition of bisimulations. Whenever we decide to realize a 1-type t in \mathfrak{B} that does not participate in the bisimulation currently being verified, we also send another copy of the automaton in state t^\sharp to guess an $a \in A_{(T,L)}$ that we can use to satisfy Condition 2 from Theorem 3.3.7:

$$\delta_2(t, (\tau, M)) = t_\uparrow \wedge \Box t_\downarrow \wedge \bigvee_{T \mid t \approx T} \bigwedge_{\lambda \in T} (\Diamond \lambda \vee \lambda_\uparrow) \quad \text{if } \tau_y =_\Sigma t$$

$$\begin{array}{ll}
\delta_2(t, (\tau, M)) & = \text{false} & \text{if } \tau_y \neq_{\Sigma} t \\
\delta_2(t_{\downarrow}, (\tau, M)) & = \text{true} & \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\downarrow}, (\tau, M)) & = \bigvee_{\lambda | t \approx \lambda \wedge \tau =_{\Sigma} \lambda} \lambda_y & \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(t_{\uparrow}, (\tau, M)) & = \text{true} & \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\uparrow}, (\tau, M)) & = \bigvee_{\lambda | t \approx \lambda \wedge \tau =_{\Sigma} \lambda^-} [-1]\lambda_y & \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(\lambda, (\tau, M)) & = \lambda_y & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_{\Sigma} \lambda \\
\delta_2(\lambda, (\tau, M)) & = \text{false} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_{\Sigma} \lambda \\
\delta_2(\lambda, (\tau, M)) & = \lambda_y^? & \text{if } \lambda \text{ is not } \Sigma\text{-guarded} \\
\delta_2(\lambda_{\uparrow}, (\tau, M)) & = \langle -1 \rangle \lambda_y & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_{\Sigma} \lambda^- \\
\delta_2(\lambda_{\uparrow}, (\tau, M)) & = \text{false} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_{\Sigma} \lambda^- \\
\delta_2(\lambda_{\uparrow}, (\tau, M)) & = \lambda_y^? & \text{if } \lambda \text{ is not } \Sigma\text{-guarded}
\end{array}$$

where $\tau_y =_{\Sigma} t$ means that the atoms in τ that mention only y are identical to the Σ -relational atoms in t (up to renaming x to y), $\tau =_{\Sigma} \lambda$ means that the restriction of λ to Σ -atoms is exactly τ , and λ^- is obtained from λ by swapping x and y . We need further transitions to satisfy the obligations represented by states of the form t^0 , which involves checking A_{\perp} -delimited $\text{GFO}^2(\Sigma)$ -bisimulations. Recall that such a bisimulation consists of two relations $\sim_{\Sigma^{\perp,0}}^{A_{\perp},0}$ and $\sim_{\Sigma^{\perp,1}}^{A_{\perp},1}$, each of which behaves essentially like a $\text{GFO}^2(\Sigma)$ -bisimulation except in some special cases that pertain to the A_{\perp} -marking of one of the involved structures, which in this case is the structure $\mathfrak{A}_{(T,L)}$. To deal with $\sim_{\Sigma^{\perp,0}}^{A_{\perp},0}$ and $\sim_{\Sigma^{\perp,1}}^{A_{\perp},1}$, we take copies q^0 and q^1 of every state q that is of the form $t, t_{\downarrow}, t_{\uparrow}, \lambda$, and λ_{\uparrow} , and also copies of the above block of transitions, modified in a suitable way to take care of the special cases. This is implemented for $\sim_{\Sigma^{\perp,0}}^{A_{\perp},0}$ by the following transitions:

$$\begin{array}{ll}
\delta_2(t^0, (\tau, M)) & = t_{\uparrow}^0 \wedge \square t_{\downarrow}^0 \wedge \bigvee_{T | t \approx T} \bigwedge_{\lambda \in T} (\diamond \lambda^0 \vee \lambda_{\uparrow}^0) & \text{if } \tau_y =_{\Sigma} t \\
\delta_2(t^0, (\tau, M)) & = \text{false} & \text{if } \tau_y \neq_{\Sigma} t \\
\delta_2(t_{\downarrow}^0, (\tau, M)) & = \text{true} & \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\downarrow}^0, (\tau, M)) & = \bigvee_{\lambda | t \approx \lambda \wedge \tau =_{\Sigma} \lambda} \lambda_y^0 & \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(t_{\uparrow}^0, (\tau, M)) & = \text{true} & \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\uparrow}^0, (\tau, M)) & = \bigvee_{\lambda | t \approx \lambda \wedge \tau =_{\Sigma} \lambda^-} [-1]\lambda_y^M & \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(\lambda^0, (\tau, M)) & = \lambda_y^0 & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_{\Sigma} \lambda \\
\delta_2(\lambda^0, (\tau, M)) & = \text{false} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_{\Sigma} \lambda \\
\delta_2(\lambda^0, (\tau, M)) & = \lambda_y^? & \text{if } \lambda \text{ is not } \Sigma\text{-guarded} \\
\delta_2(\lambda_{\uparrow}^0, (\tau, M)) & = \langle -1 \rangle (\lambda_y)_{\&} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_{\Sigma} \lambda^- \\
\delta_2(\lambda_{\uparrow}^0, (\tau, M)) & = \text{false} & \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_{\Sigma} \lambda^- \\
\delta_2(\lambda_{\uparrow}^0, (\tau, M)) & = \lambda_y^? & \text{if } \lambda \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\&}, (\tau, M)) & = t^M &
\end{array}$$

The transitions for $\sim_{\Sigma}^{A_{\perp},1}$ are as follows:

$$\begin{aligned}
\delta_2(t^1, (\tau, 1)) &= (t^? \wedge t_{\uparrow}^1 \wedge \square t_{\downarrow}^1 \wedge \bigvee_{T|t \approx T} \bigwedge_{\lambda \in T} (\diamond \lambda^1 \vee \lambda_{\uparrow}^1)) \text{ if } \tau_y =_{\Sigma} t \\
&\quad \vee (t^? \wedge \langle -1 \rangle q_{\perp}) \\
\delta_2(t^1, (\tau, 0)) &= (t^? \wedge t_{\uparrow}^1 \wedge \square t_{\downarrow}^0 \wedge \bigvee_{T|t \approx T} \bigwedge_{\lambda \in T} (\diamond \lambda^0 \vee \lambda_{\uparrow}^1)) \text{ if } \tau_y =_{\Sigma} t \\
&\quad \vee (t^? \wedge \langle -1 \rangle q_{\perp}) \\
\delta_2(t^1, (\tau, M)) &= \langle -1 \rangle q_{\perp} && \text{if } \tau_y \neq_{\Sigma} t \\
\delta_2(q_{\perp}, (\tau, 0)) &= \text{false} \\
\delta_2(q_{\perp}, (\tau, 1)) &= \text{true} \\
\delta_2(t_{\downarrow}^1, (\tau, M)) &= \text{true} && \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\downarrow}^1, (\tau, M)) &= \bigvee_{\lambda | t \approx \lambda \wedge \tau =_{\Sigma} \lambda} \lambda_y^1 && \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(t_{\uparrow}^1, (\tau, M)) &= \text{true} && \text{if } \tau \text{ is not } \Sigma\text{-guarded} \\
\delta_2(t_{\uparrow}^1, (\tau, M)) &= \bigvee_{\lambda | t \approx \lambda \wedge \tau =_{\Sigma} \lambda^-} [-1] \lambda_y^1 && \text{if } \tau \text{ is } \Sigma\text{-guarded} \\
\delta_2(\lambda^1, (\tau, M)) &= \lambda_y^1 && \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_{\Sigma} \lambda \\
\delta_2(\lambda^1, (\tau, M)) &= \text{false} && \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_{\Sigma} \lambda \\
\delta_2(\lambda^1, (\tau, M)) &= \lambda_y^? && \text{if } \lambda \text{ is not } \Sigma\text{-guarded} \\
\delta_2(\lambda_{\uparrow}^1, (\tau, M)) &= \langle -1 \rangle \lambda_y^1 && \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau =_{\Sigma} \lambda^- \\
\delta_2(\lambda_{\uparrow}^1, (\tau, M)) &= \text{false} && \text{if } \lambda \text{ is } \Sigma\text{-guarded and } \tau \neq_{\Sigma} \lambda^- \\
\delta_2(\lambda_{\uparrow}^1, (\tau, M)) &= \lambda_y^? && \text{if } \lambda \text{ is not } \Sigma\text{-guarded}
\end{aligned}$$

Lemma 3.4.6. \mathcal{A}_2 satisfies the condition from Lemma 3.4.5.

Proof. “ \Leftarrow ”. Let (T, L) be a Θ -labeled tree and let \mathfrak{B} be a model of φ_2 such that Conditions 1 and 2 of Theorem 3.3.7 are satisfied when \mathfrak{A} is replaced with $\mathfrak{A}_{(T,L)}$ (and when A_{\perp} is the set described by the second component of the L -labels). We argue that \mathfrak{B} can be used to guide a run of \mathcal{A}_2 on (T, L) so that it is accepting.

In this run, \mathcal{A}_2 starts with choosing the 0-type s realized by \mathfrak{B} . Then, for each $\exists x \varphi(x) \in s$, we guide \mathcal{A}_2 to proceed in state $t^?$, where t is the 1-type of some element $b \in B$ with $\mathfrak{B} \models \varphi(b)$. By Condition 2 of Theorem 3.3.7, there is a $w \in A_{(T,L)}$ such that $\text{tp}_{\mathfrak{B}}(b) = t$ and $(\mathfrak{A}_{(T,L)}, w) \sim^{A_{\perp}} (\mathfrak{B}, b)$. In the search state $t^?$, we guide the run to reach w and switch to state t^0 there. The automaton also sends a copy in state s to each node $w \in A_{(T,L)}$. By Condition 1 of Theorem 3.3.7, there is a $b \in B$ such that $(\mathfrak{A}_{(T,L)}, w) \sim_{\Sigma} (\mathfrak{B}, b)$. We guide the run to proceed in state t , the 1-type of b .

At this point, the automaton needs to satisfy two kinds of obligations:

1. states t true at a node $w \in A_{(T,L)}$ representing the obligation to verify that there is a $b \in B$ with 1-type t and such that $(\mathfrak{A}_{(T,L)}, w) \sim_{\Sigma} (\mathfrak{B}, b)$ and
2. states t^0 true at a node $w \in A_{(T,L)}$ representing the obligation to verify that there is a $b \in B$ with 1-type t and such that $(\mathfrak{A}_{(T,L)}, w) \sim_{\Sigma}^{A_{\perp}} (\mathfrak{B}, b)$.

Note that we have guided the run so that the required bisimulations indeed exist and therefore we can use them to further guide the run. We only consider Case 1 above, thus concentrating on states of the form t , t_{\downarrow} , t_{\uparrow} , λ , and λ_{\uparrow} . Suppose the automaton is in state t at node w . By the way in which we guide the run, there is then a $b \in B$ with 1-type t and such that $(\mathfrak{A}_{(T,L)}, w) \sim_{\Sigma} (\mathfrak{B}, b)$. We guide the run to select as T

the set of all guarded 2-types λ such that $\mathfrak{B} \models (\exists y \lambda(x, y))(b)$. For each such λ , there must be a $b' \in B$ and a $v \in A_{(T,L)}$ with $\mathfrak{B} \models \lambda(b, b')$ and $(\mathfrak{A}_{(T,L)}, v) \sim_{\Sigma} (\mathfrak{B}, b')$ where v is either the predecessor of w or a successor of it. In the former case, we guide the automaton to switch to state λ_{\uparrow} and in the latter, we guide it to execute $\diamond \lambda$. When the automaton was sent in state t_{\downarrow} to a successor v of w , then there must be a $b' \in B$ such that $(\mathfrak{A}_{(T,L)}, v) \sim_{\Sigma} (\mathfrak{B}, b')$ and $\mathfrak{B} \models \lambda(b, b')$ for some guarded 2-type λ . Guide the run to choose λ . The decision to be taken for states t_{\uparrow} is handled very similarly.

“ \Rightarrow ”. Let (T, L) be a Θ -labeled tree that is accepted by \mathcal{A}_2 . Then there is an accepting run (T_r, r) of \mathcal{A}_2 on (T, L) . We show how to use (T_r, r) to construct a model \mathfrak{B} of φ_2 such that Conditions 1 and 2 of Theorem 3.3.7 are satisfied when \mathfrak{A} is replaced with $\mathfrak{A}_{(T,L)}$. Along with \mathfrak{B} , we construct the following objects:

- a $\text{GFO}^2(\Sigma)$ -bisimulation \sim between $\mathfrak{A}_{(T,L)}$ and \mathfrak{B} which witnesses that Condition 1 of Theorem 3.3.7 is satisfied,
- two relations $\sim^{A_{\perp},0}$ and $\sim^{A_{\perp},1}$ that form an A_{\perp} -delimited $\text{GFO}^2(\Sigma)$ -bisimulation between $\mathfrak{A}_{(T,L)}$ and \mathfrak{B} , where $A_{\perp} \subseteq \mathfrak{A}_{(T,L)}$ is the subset defined by the second component of L , and which witness that Condition 2 of Theorem 3.3.7 is satisfied, and
- a function μ that assigns to each element of \mathfrak{B} the 1-type that we aim to realize there.

Throughout the construction, we make sure that the following invariants are satisfied:

1. if $(w, b) \in \sim$, then the label $(w, \mu(b))$ occurs in (T_r, r) ;
2. if $(w, b) \in \sim^{A_{\perp},i}$, $i \in \{0, 1\}$, then the label $(w, \mu(b)^0)$ occurs in (T_r, r) .

The start of the construction is as follows:

- for each label (w, t) that occurs in (T_r, r) , introduce an element b of B , add (w, b) to \sim , and set $\mu(b) = t$;
- for each label (w, t^0) that occurs in (T_r, r) , introduce an element b of B , add (w, b) to $\sim^{A_{\perp},0}$, and set $\mu(b) = t$.

We then iteratively extend \mathfrak{B} , \sim , $\sim^{A_{\perp},0}$, $\sim^{A_{\perp},1}$, and μ , obtaining the desired structure and bisimulations in the limit. In each step, process every $b \in B$ that has not been processed in a previous round. There are three cases.

Case (a). There is a $(w, b) \in \sim$. By Invariant 1, we find a node $x \in T_r$ such that $r(x) = (w, \mu(b))$. We perform two steps:

- For every predecessor or successor v of w in T with $\text{at}_{\mathfrak{A}_{(T,L)}}^{\Sigma}(w, v)$ guarded, there must be a 2-type λ such that $\mu(b) \approx \lambda$, (v, λ_y) occurs as a label in (T_r, r) , and $\text{at}_{\mathfrak{A}_{(T,L)}}^{\Sigma}(w, v) =_{\Sigma} \lambda$. Extend \mathfrak{B} with a new element b' , extend the interpretation of the predicates in \mathfrak{B} such that $\text{at}_{\mathfrak{B}}^{\Sigma}(w, v) =_{\Sigma} \lambda$, set $\mu(b') = \lambda_y$, and extend \sim with (v, b') .
- There must be a set T of guarded 2-types such that $t \approx T$ and for every $\lambda \in T$, there is a predecessor or successor v of w in T such that $\mu(b) \approx \lambda$, (v, λ_y) occurs as a label in (T_r, r) , and $\text{at}_{\mathfrak{A}_{(T,L)}}^{\Sigma}(w, v) =_{\Sigma} \lambda$. Extend \mathfrak{B} with a new element b' (for every λ), extend the interpretation of the predicates in \mathfrak{B} such that $\text{at}_{\mathfrak{B}}^{\Sigma}(w, v) =_{\Sigma} \lambda$, set $\mu(b') = \lambda_y$, and extend \sim with (v, b') .

Case (b). There is a $(w, b) \in \sim^{A_\perp, 0}$. By Invariant 2, we find a node $x \in T_r$ such that $r(x) = (w, \mu(b)^0)$. We can now proceed exactly as in Case (a) except that, in both subcases, we add (v, b') to $\sim^{A_\perp, 1}$ if v is a predecessor of w and $v \in A_\perp$, and to $\sim^{A_\perp, 0}$ otherwise.

Case (c). There is a $(w, b) \in \sim^{A_\perp, 1}$. By Invariant 2, we find a node $x \in T_r$ such that $r(x) = (w, \mu(b)^1)$. If the predecessor of w is not in A_\perp , then we again proceed as in Case (a) except that, in both subcases, we add (v, b') to $\sim^{A_\perp, 0}$ if v is a successor of w and $w \in A_\perp$, and to $\sim^{A_\perp, 1}$ otherwise. If the predecessor of w is in A_\perp , then we also proceed as in Case (a), but do not add (v, b') to any of the constructed bisimulations.

Case (d). None of the above cases applies. Then we proceed as in Case (a), again not adding (v, b') to any of the constructed bisimulations.

It can be verified that, as intended the structure \mathfrak{B} obtained in the limit is a model of φ_2 , that the relation \sim is a $\text{GFO}^2(\Sigma)$ -bisimulation, and that $\sim^{A_\perp, 0}, \sim^{A_\perp, 1}$ form an A_\perp -delimited $\text{GFO}^2(\Sigma)$ -bisimulation. \square

Recall that we define the overall 2ATA \mathcal{A} so that it accepts $L(\mathcal{A}_1) \cap \overline{L(\mathcal{A}_2)} \cap L(\mathcal{A}_3)$. Using Theorem 3.3.7, it can be verified that, as intended, $\varphi_1 \models_{\text{GFO}^2(\Sigma)} \varphi_2$ iff $L(\mathcal{A}) = \emptyset$. Note that for the “only if” direction, we have to show that $L(\mathcal{A}) \neq \emptyset$ implies that there is a *regular* forest model of φ_1 that satisfies the negation of the conditions in Theorem 3.3.7. As is the case for other kinds of tree automata, also for the 2ATA \mathcal{A} it can be shown that $L(\mathcal{A}) \neq \emptyset$ implies that \mathcal{A} accepts a regular Θ -labeled tree (T, L) . The corresponding structure $\mathfrak{A}_{(T, L)}$ must then also be regular.

Theorem 3.4.7. *In GFO^2 , Σ -entailment and conservative extensions can be decided in time $2^{2^{p(|\varphi_2| - \log |\varphi_1|)}}$, for some polynomial p . Moreover, Σ -inseparability is in 2EXPTIME .*

Note that the time bound for conservative extensions given in Theorem 3.4.7 is double exponential only in the size of φ_2 (that is, the extension). In ontology engineering applications, φ_2 will often be small compared with φ_1 .

3.4.3 Lower Bound

We show that Σ -entailment, Σ -inseparability, and conservative extensions in GFO^2 are 2EXPTIME -hard. The proof is by reduction of the word problem for exponentially space bounded alternating Turing machines (ATMs). The construction is inspired by the proof from [GLW06] that conservative extensions in the description logic \mathcal{ALC} are 2EXPTIME -hard, but the lower bound does not transfer directly since we are interested here in witness sentences that are formulated in GFO^2 rather than in \mathcal{ALC} .

Definition 3.4.1. An ATM is of the form $M = (Q, \Theta, \Gamma, q_0, \Delta)$. The set of states $Q = Q_\exists \uplus Q_\forall \uplus \{q_a\} \uplus \{q_r\}$ consists of *existential states* from Q_\exists , *universal states* from Q_\forall , an *accepting state* q_a , and a *rejecting state* q_r ; Θ is the *input alphabet* and $\Gamma \supset \Theta$ the *work alphabet* that contains a *blank symbol* $\square \notin \Theta$; $q_0 \in Q_\exists$ is the *starting state*; and the *transition relation* Δ is of the form $\Delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$. We write $\Delta(q, a)$ for $\{(q', b, M) \mid (q, a, q', b, M) \in \Delta\}$ and assume that $\Delta(q, b) = \emptyset$ for all $q \in \{q_a, q_r\}$ and $b \in \Gamma$.

A *configuration* of an ATM is a word wqw' with $w, w' \in \Gamma^*$ and $q \in Q$. The intended meaning is that the one-side infinite tape contains the word ww' with only blanks behind it, the machine is in state q , and the head is on the symbol just after w . The *successor configurations* of a configuration wqw' are defined in the usual way in terms

of the transition relation Δ . A *halting configuration* (resp. *accepting configuration*) is of the form wqw' with $q \in \{q_a, q_r\}$ (resp. $q = q_a$).

A *computation tree* of an ATM M on input w is a tree whose nodes are labeled with configurations of M on w , such that the descendants of any non-leaf labeled by a universal (resp. existential) configuration include all (resp. one) of the successor configurations of that configuration. A computation tree is *accepting* if the root is labeled with the *initial configuration* q_0w for w and all leaves with accepting configurations. An ATM M accepts input w if there is a computation tree of M on w .

Take an exponentially space bounded ATM M whose word problem is 2EXPTIME-hard [CKS81]. We may w.l.o.g. assume that the length of every computation path of M on $w \in \Theta^n$ is bounded by 2^{2^n} . We can also assume that for each $q \in Q_\forall \cup Q_\exists$ and each $a \in \Gamma$, the set $\Delta(q, a)$ has exactly two elements. We assume that these elements are ordered, i.e., $\Delta(q, a)$ is an ordered pair $((q_L, b_L, M_L), (q_R, b_R, M_R))$. Furthermore, we assume that M never attempts to move left on the left-most tape cell.

Let $w = a_0 \cdots a_{n-1} \in \Theta^*$ be an input to M . In the following, we construct GFO² sentences φ_1 and φ_2 such that $\varphi_1 \wedge \varphi_2$ is a conservative extension of φ_1 if and only if M does not accept w . Informally, the main idea is to construct φ_1 and φ_2 such that models of sentences that witness non-conservativity describe an accepting computation tree of M on w . In such models, each domain element represents a tape cell of a configuration of M , the binary predicate N indicates moving to the next tape cell in the same configuration, and the binary predicates L and R indicate moving to left and right successor configurations in accepting configuration trees. Thus, each node of the computation tree (that is, each configuration) is spread out over a sequence of nodes in the model. We actually assume that every non-halting configuration has two successor configurations, also when its state is existential. This can of course easily be achieved by duplicating subtrees in computation trees. The following predicates are used in φ_1 :

- a unary predicate P to mark the root of computation trees;
- binary predicates N, R, L , as explained above;
- unary predicates C_0, \dots, C_{n-1} that represent the bits of a binary counter which identifies tape positions;
- a unary predicate F that marks the topmost configuration in the configuration tree;
- unary predicates $S_a, a \in \Gamma$, to represent the tape content of cells that are not under the head;
- unary predicates $S_{q,a}, q \in Q$ and $a \in \Gamma$, to represent the state of a configuration, the head position, and the tape content of the cell that is under the head;
- unary predicates S_a^p and $S_{q,a}^p$, with the ranges of q and a as above, to represent the same information, but for the previous configuration in the tree instead of for the current one;
- unary predicates $Y_{L,q,a,M}$ and $Y_{R,q,a,M}, q \in Q, a \in \Gamma, M \in \{L, R\}$, to record the transition to be executed in the subsequent configurations;
- unary predicates $Y_{q,a,M}$, with the ranges of q, a, M as above, to record the transition executed to reach the current configuration.

$$\begin{aligned}
& \forall x (Px \rightarrow \varphi_{C=0}(x)) & (1) \\
& \forall x (\varphi_{C < 2^n - 1}(x) \rightarrow (\exists y Nxy \wedge \forall y (Nxy \rightarrow \varphi_{C++}(x, y))) & (2) \\
& \forall x (\varphi_{C = 2^n - 1}(x) \rightarrow (\exists y Lxy \wedge \forall y (Lxy \rightarrow \varphi_{C=0}(y)) \wedge \exists y Rxy \wedge \forall y (Rxy \rightarrow \varphi_{C=0}(y)))) & (3) \\
& \forall x ((Px \rightarrow Fx) \wedge \forall y (Nxy \rightarrow Fy)) & (4) \\
& \forall x \bigvee_{\alpha \in \Gamma \cup (Q \times \Gamma)} (S_\alpha x \wedge \bigwedge_{\beta \in (\Gamma \cup (Q \times \Gamma)) \setminus \{\alpha\}} \neg S_\beta x) & (5) \\
& \forall x \bigvee_{\alpha \in \Gamma \cup (Q \times \Gamma)} (S_\alpha^p x \wedge \bigwedge_{\beta \in (\Gamma \cup (Q \times \Gamma)) \setminus \{\alpha\}} \neg S_\beta^p x) & (6) \\
& \forall x ((Fx \wedge \varphi_{C=0}(x)) \rightarrow S_{q_0, a_0} x) & (7) \\
& \forall x ((Fx \wedge \varphi_{C=i}(x)) \rightarrow S_{a_i} x) \quad \text{for } 1 \leq i < n & (8) \\
& \forall x ((Fx \wedge \varphi_{C \geq n}(x)) \rightarrow S_\square x) & (9) \\
& \forall x (S_{q,a} x \rightarrow (Y_{L,T_L} x \wedge Y_{R,T_R} x)) \quad \text{if } \Delta(q, a) = (T_L, T_R), q \in Q_\forall & (10) \\
& \forall x (S_{q,a} x \rightarrow ((Y_{L,T_L} x \wedge Y_{R,T_L} x) \vee (Y_{L,T_R} x \wedge Y_{R,T_R} x))) & (11) \\
& \quad \text{if } \Delta(q, a) = (T_L, T_R), q \in Q_\exists \\
& \forall x (Y_{P,T} x \rightarrow \forall y (Nxy \rightarrow Y_{P,Ty})) & (12) \\
& \forall x (Y_{P,T} x \rightarrow \forall y (Pxy \rightarrow Y_T y)) & (13) \\
& \forall x (Y_T x \rightarrow \forall y (Nxy \rightarrow Y_T y)) & (14) \\
& \forall x ((Y_{q,a,M} x \wedge S_{q',b}^p x) \rightarrow S_a x) & (15) \\
& \forall x ((Y_{q,a,L} x \wedge S_b^p x \wedge \exists y (Nxy \wedge S_{q',a'y}^p y)) \rightarrow S_{q,b} x) & (16) \\
& \forall x ((Y_{q,a,R} x \wedge S_b^p x \wedge \exists y (Nxy \wedge S_{q',a'y}^p y)) \rightarrow S_b x) & (17) \\
& \forall x ((Y_{q,a,R} x \wedge S_b^p x \wedge \exists y (Nyx \wedge S_{q',a'y}^p y)) \rightarrow S_{q,b} x) & (18) \\
& \forall x ((Y_{q,a,L} x \wedge S_b^p x \wedge \exists y (Nyx \wedge S_{q',a'y}^p y)) \rightarrow S_b x) & (19) \\
& \forall x ((\exists y (Nxy \wedge S_b^p y)) \wedge S_a^p x \wedge \exists y (Nyx \wedge S_b^p y)) \rightarrow S_a x) & (20) \\
& \forall x \neg S_{q_r, a} x & (21)
\end{aligned}$$

Figure 3.1: The conjuncts of the sentence φ_1 .

The sentence φ_2 uses some additional unary predicates, including C'_0, \dots, C'_{n-1} to implement another counter whose purpose is explained below.

The sentences φ_1 and φ_2 are shown in Figures 3.1 and 3.2, respectively, where q and q' range over Q , a, b, b' over Γ , M and P over $\{L, R\}$, T, T_L, T_R over $Q \times \Gamma \times \{L, R\}$, and α over $\Gamma \cup (Q \times \Gamma)$. The formula $\varphi_{C=i}(x)$, which is easily worked out in detail, expresses that the value of the binary counter implemented by C_0, \dots, C_{n-1} has value exactly i at x , and likewise for $\varphi_{C < i}(x)$ and $\varphi_{C \geq i}(x)$, and for the primed versions in φ_2 which refer to the counter implemented by C'_0, \dots, C'_{n-1} . The formula $\varphi_{C++}(x, y)$ expresses that the counter value at y is obtained from the counter value at x by incrementation modulo 2^n . Again, we omit the details.

Let us walk through φ_1 and φ_2 and give some intuition of what the various conjuncts are good for. In φ_1 , Lines (1) to (4) ensure that at an element that satisfies P , there is

$\exists x Px \rightarrow \exists x Dx$	(22)
$\forall x (Dx \rightarrow (Mx \wedge \varphi_{C'=0}(x)))$	(23)
$\forall x ((Dx \wedge S_\alpha x) \rightarrow Z_\alpha x)$	(24)
$\forall x ((Mx \wedge \varphi_{C < 2^n - 1}(x) \wedge \varphi_{C' < 2^n - 1}(x) \wedge Z_\alpha x) \rightarrow \exists y (Nxy \wedge My \wedge Z_\alpha y \wedge \varphi_{C'++}(x, y)))$	(25)
$\forall x ((Mx \wedge \varphi_{C=2^n-1}(x) \wedge \varphi_{C' < 2^n-1}(x) \wedge Z_\alpha x) \rightarrow$	(26)
$\exists y (Lxy \wedge My \wedge Z_\alpha y \wedge \varphi_{C'++}(x, y)) \vee \exists y (Rxy \wedge My \wedge Z_\alpha y \wedge \varphi_{C'++}(x, y))$	
$\forall x ((Mx \wedge \varphi_{C'=2^n-1}(x) \wedge Z_\alpha x) \rightarrow \exists y (Nxy \wedge \neg S_\alpha^p x))$	(27)

Figure 3.2: The conjuncts of the sentence φ_2 .

an infinite tree of the expected pattern: first $2^n - 1$ N -edges without branching, then a binary branching of an L -edge and an R -edge, then $2^n - 1$ N -edges without branching, and so on, ad infinitum. Of course, a computation tree will be represented using only a finite initial piece of this infinite tree. These conjuncts also set up the counter C so that it identifies the position of tape cells and the marker F so that it identifies the topmost configuration in the tree. Line (5) says that every cell is labeled with exactly one symbol and that the state is unique (locally to one cell; there is no need to express the same globally for the entire configuration), and Line (6) says the same for the representation of the previous configuration. Lines (7) to (9) make sure that the topmost configuration in the infinite tree is the initial configuration of M on input w . Lines (10) and (11) choose transitions to execute and Lines (12) to (14) propagate this choice down to the subsequent configurations. Assume that the predicates of the S_a^p and $S_{q,a}^p$ indeed represent the previous configuration, Lines (15) to (20) then implement the chosen transitions. Line (21) says that we do never see a rejecting halting configuration.

Now for φ_2 . Essentially, we want to achieve that a sentence is a witness for non-conservativity if and only if it expresses that its models contain (a representation of) an accepting computation tree of M on w whose root is labeled with P . This is achieved by designing φ_2 so that, whenever a tree model of φ_1 contains only instances of P that are not the root of such a computation tree, then this model can be extended to a model of φ_2 by assigning an interpretation to the additional predicates in φ_2 . Note that φ_1 already enforces that, below any instance of P , there is a tree that satisfies almost all of the required conditions of an accepting computation tree. In fact, the only way in which that tree cannot be an accepting configuration tree is that the predicates S_a^p and $S_{q,a}^p$ do not behave in the expected way, that is, there is a configuration and a cell in this configuration that is labeled with S_α , $\alpha \in \Gamma \cup (Q \times \Gamma)$, and in one of the two subsequent configurations the same cell is not labeled with S_α^p . We thus design φ_2 so that it can be made true whenever the model contains such a defect. In Line (22), we select the place where the defect is. Line (23) ensures that the counter C' starts counting with value zero at that place, and that the marker M is set there, too. Line (24) memorizes the content α of the cell in the upper configuration involved in the defect. Lines (25) to (27) propagate downwards the memorized content and make sure that, at the corresponding cell of at least one subsequent configuration (which is identified using the counter C'), we do not find S_α^p .

Lemma 3.4.8.

1. If M accepts w , then $\varphi_1 \wedge \varphi_2$ is not a GFO^2 -conservative extension of φ_1 .
2. If there exists a $\text{sig}(\varphi_1)$ -structure that satisfies φ_1 and cannot be extended to a model of $\varphi_1 \wedge \varphi_2$, then M does not accept w .

Proof.(sketch) For Point 1 assume that M accepts w . Then there is an accepting computation tree of M on w . Let $\Sigma = \text{sig}(\varphi_1)$. We can find a $GFO^2(\Sigma)$ -sentence ψ_1 which expresses that the model contains a (homomorphic image of a) finite tree which represents this configuration tree and whose root is labeled with P . We can also find a $GFO^2(\Sigma)$ -sentence ψ_2 which expresses that nowhere in the model there is a defect situation. It can be verified that $\psi_1 \wedge \psi_2$ is satisfiable w.r.t. φ_1 , but not w.r.t. φ_2 because φ_2 requires the existence of a defect situation whenever the extension of P is non-empty.

For Point 2 assume that \mathfrak{A} is a $\text{sig}(\varphi_1)$ -structure that satisfies φ_1 . If $P^{\mathfrak{A}} = \emptyset$, then the desired model \mathfrak{B} is obtained from \mathfrak{A} by interpreting all predicates in $\text{sig}(\varphi_2) \setminus \text{sig}(\varphi_1)$ as empty. Otherwise, take some $a \in P^{\mathfrak{A}}$. We can follow the existential quantifiers in φ_1 to identify a homomorphic image of an infinite tree in \mathfrak{A} with root a whose edges follow the expected pattern and that is labeled in the expected way by the counter C . Since \mathfrak{A} is a model of φ_1 , an initial piece of the identified tree represents an accepting computation tree of M on w provided that the predicates S_α^p behave as expected, that is, if there is no defect of the form described above. Since M does not accept w , there must thus be such a defect, that is a path of length 2^n that links a cell of a configuration with the corresponding cell of a subsequent configuration such that the former is labeled with S_α , but the latter is not labeled with S_α^p . All the elements of the path (with the possible exception of the start point and the end point) are labeled with a different value of the counter C and must thus be distinct. Consequently, we can interpret the counter C' and the other symbols in φ_2 to extend \mathfrak{A} to a model of φ_2 , as desired. \square

The following result is an immediate consequence of Lemma 3.4.8.

Theorem 3.4.9. *In any fragment of FO that contains GFO^2 , the problems Σ -entailment, Σ -inseparability, conservative extensions, and strong Σ -entailment are 2EXPTIME-hard.*

3.5 Concluding Remarks

We have shown that conservative extensions are undecidable in (extensions of) GFO and FO^2 , and that they are decidable and 2EXPTIME-complete in GFO^2 . It thus appears that decidability of conservative extensions is linked even more closely to the tree model property than decidability of the satisfiability problem: apart from cycles of length at most two, GFO^2 enjoys a ‘true’ tree model property while GFO only enjoys a bounded treewidth model property [Grä99] and FO^2 has a rather complex regular model property that is typically not even made explicit [Mor75].

As future work, it would be interesting to investigate whether conservative extensions remain decidable when guarded counting quantifiers, transitive relations, equivalence relations, or fixed points are added, see e.g. [Pra07, Kie06, GW99]. Furthermore, it would be interesting to know where is the limit of decidability: Are conservative extensions decidable in FO^2 when the separating formulas are formulated in \mathcal{ALC} , modal logic, or something weaker? How low do we have to go to make it decidable? Also, Are conservative extensions decidable in GFO when the second formula φ_2 can only use fresh relations that are at most binary? And can we come up with decidable separability

notions for GFO and FO^2 that are not defined in terms of logical consequences, but in terms of natural model-theoretic characterizations? These kind of questions deserve further investigation. Moreover, it would also be interesting to investigate a finite model version of conservative extensions.

Deductive conservative extensions, as studied in this chapter, aim at preserving the logical (non)-consequences of sentences. A crucial difference between ontology-based data access and more traditional ontology reasoning, though, is that this kind of entailment (and related satisfiability) questions play only a peripheral role while query answering becomes the central reasoning task. In these scenarios, the notion of query conservative extensions is of main importance. This is the topic of the next chapter.

Conservative Extensions in Horn Description Logics with Inverse Roles

In the past years, access of incomplete data mediated by description logic (DL) ontologies has gained increasing importance [PLC⁺08, BO15]. The main idea is to specify domain knowledge and semantics of the data in the ontology, resulting in more complete answers to queries. Significant research activity has led to efficient algorithms and tools for a wide range of DLs such as DL-Lite [CDL⁺07], more expressive Horn-DLs [EOŠ⁺12, TSCS15, BHLW16], and “full Boolean” DLs such as \mathcal{ALC} [KG13, ZCN⁺15].

In contrast to query answering, which is by now well-understood, there is a need to develop reasoning services for ontology engineering that are tailored towards query-centric applications and support tasks such as ontology versioning and module extraction from ontologies. For example, if one wants to safely replace an ontology with a new version or with a smaller subset of itself (a module), then the new ontology should preserve the answers to all queries over all ABoxes (which store the data) [KWZ10]. The same guarantee ensures that one can safely replace an ontology with another version in an application [KLWW12]. In both cases, ontologies need to be tested not for their logical equivalence, but for giving the same answers to relevant queries over relevant datasets.

This requirement can be formalized using conservative extensions. In the following, we use the DL term *TBox* instead of *ontology*. A TBox $\mathcal{T}_2 \supseteq \mathcal{T}_1$ is a (Γ, Σ) -*query conservative extension* of a TBox \mathcal{T}_1 , where Γ and Σ are signatures of concept/role names relevant for data and queries, respectively, if all Σ -queries give the same answers w.r.t. \mathcal{T}_1 and \mathcal{T}_2 , for every Γ -ABox. Note that the subset relationship $\mathcal{T}_2 \supseteq \mathcal{T}_1$ is natural when replacing a TBox with a module, but not in versioning, so we might not want to insist on it. In this more general case, \mathcal{T}_1 and \mathcal{T}_2 are called (Γ, Σ) -*query inseparable*. Conservativity and inseparability of *TBoxes*, as defined above, are useful when knowledge is considered static and data changes frequently. Variants of these notions for *knowledge bases (KBs)*, which consist of a TBox and an ABox, can be used for applications with static data [WWT⁺14, ABCR16].

We also consider the basic notion of query entailment: \mathcal{T}_1 (Γ, Σ) -*query entails* \mathcal{T}_2 if all Σ -queries give *at least* the answers w.r.t. \mathcal{T}_1 that they give w.r.t. \mathcal{T}_2 , on any Γ -ABox. Query inseparability and conservativity are special cases of entailment as seen in the previous chapter. As a query language, we concentrate on conjunctive queries (CQs); since we work with Horn-DLs and quantify over the queries, this is equivalent to using

unions of CQs (UCQs) or positive existential queries (PEQs). CQ entailment has been studied for various DLs [KPS⁺09, LW10, KLWW12, BLR⁺16], also in the KB version [BKR⁺16, BLR⁺16] and for OBDA specifications [BR15], see also the survey [BKL⁺16]. Nevertheless, there is still a notable gap in our understanding of this notion: query entailment between TBoxes is poorly understood in Horn DLs with inverse roles, often considered a crucial feature, for which there do not seem to be any available results. Therefore we study the last question of Section 1.2.1:

Q3 How can advanced reasoning support for ontologies be lifted from standard DLs to Horn DLs involving inverse roles, thus facilitating ontology design and maintenance for OBDA applications?

Inverse roles have been found particularly challenging, and this is for a reason: it has been observed in [BKL⁺16, BKR⁺16] that standard techniques for Horn DLs without inverse roles fail when inverse roles are added. In fact, for Horn-DLs without inverse roles query entailment can be characterized by the existence of homomorphisms between universal models [LW10, BKL⁺16]. The resulting characterizations provide an important foundation for decision procedures, often based on tree automata [BKL⁺16]. In the presence of inverse roles, however, such characterizations are only correct if we require the existence of *n*-bounded homomorphisms, for any *n* [BKL⁺16, BKR⁺16]. It is not obvious how the existence of such infinite families of bounded homomorphisms can be verified using tree automata (or related techniques) and, consequently, decidability results for query conservative extensions in Horn-DLs with inverse roles are difficult to obtain. The only result we are aware of concerns inseparability of *KBs*, and it is proved using intricate game-theoretic techniques.

In this chapter, we develop decision procedures for query entailment and related problems in Horn DLs with inverse roles. The main idea is to provide a more refined characterization, mixing unbounded and bounded homomorphisms and using bounded homomorphisms only in places where this is strictly necessary. We can then deal with the “unbounded part” using tree automata while the “bounded part” is addressed by precomputing relevant information using a mosaic technique. In this way, we establish decidability and a 2EXPTIME upper bound for query entailment (and thus inseparability and conservativity) in Horn-*ALCHIF*. Together with lower bounds from [BLR⁺16], we get 2EXPTIME-completeness for all fragments of Horn-*ALCHIF* that contain *ELI* or Horn-*ALC*.

We additionally study the case of deductive entailment between TBoxes, i.e., the question whether \mathcal{T}_1 entails at least the same concept and role inclusions as well as functionality assertions over Σ as \mathcal{T}_2 . This problem too has not previously been studied for Horn DLs with inverse roles. We consider *ELHIF*_⊥-TBoxes and show that deductive entailment is equivalent to a restricted version of query entailment. We obtain a model theoretic characterization, a decision procedure, and a 2EXPTIME upper complexity bound. We also give a CONEXPTIME lower bound.

4.1 Horn-*ALCHIF*

We introduce Horn-*ALCHIF*, a member of the Horn-*ALCQI* family of DLs whose reasoning problems have been widely studied [HMS07, KRH07, EGOŠ08, Kaz09, LW12, ILS14]. Let N_C, N_R, N_I be sets of concept, role, and individual names. A *role* is either a role name *r* or an *inverse role* r^- . As usual, we identify $(r^-)^-$ and *r*, allowing to

switch between roles names and their inverses easily. A *concept inclusion (CI)* is of the form $L \sqsubseteq R$, where L and R are concepts defined by the syntax rules

$$\begin{aligned} R, R' &::= \top \mid \perp \mid A \mid \neg A \mid R \sqcap R' \mid \neg L \sqcup R \mid \exists r.R \mid \forall r.R \\ L, L' &::= \top \mid \perp \mid A \mid L \sqcap L' \mid L \sqcup L' \mid \exists r.L \end{aligned}$$

with A ranging over concept names and r over roles. A *role inclusion (RI)* is of the form $r \sqsubseteq s$ with r, s roles and a *functionality assertion (FA)* is of the form $\text{func}(r)$ with r a role. \mathcal{ELI}_\perp -concepts are expressions that are built according to the syntax rule for L above, but do not use “ \sqcup ”.

A *Horn- \mathcal{ALCHIF} TBox* \mathcal{T} is a set of CIs, RIs, and FAs. An \mathcal{ELHIF}_\perp TBox is a set of \mathcal{ELI}_\perp -CIs, RIs, and FAs. To avoid dealing with rather messy technicalities that do neither seem to be very illuminating from a theoretical viewpoint nor too useful from a practical one,¹ we generally assume that functional roles cannot have any subroles, that is, $r \sqsubseteq s \in \mathcal{T}$ implies $\text{func}(s) \notin \mathcal{T}$. We conjecture that our main results also hold without that restriction.

The semantics is defined as usual in terms of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ complying with the standard name assumption, i.e., $a^{\mathcal{I}} = a$ for all $a \in \mathbf{N}_1$ (see Section 2.3). An interpretation \mathcal{I} is a *model* of a TBox \mathcal{T} if it satisfies all inclusions and assertions in it, and likewise for ABoxes. \mathcal{A} is *consistent* with \mathcal{T} if \mathcal{T} and \mathcal{A} have a common model.

A *signature* Σ is a set of concept and role names. A Σ -ABox is an ABox that uses only concept and role names from Σ , and likewise for Σ - \mathcal{ELI}_\perp -concepts and other syntactic objects.

Generally and without further notice, we work with Horn- \mathcal{ALCHIF} TBoxes that are in a certain nesting-free normal form, that is, they contain only CIs of the form

$$\top \sqsubseteq A, A \sqsubseteq \perp, A_1 \sqcap A_2 \sqsubseteq B, A \sqsubseteq \exists r.B, A \sqsubseteq \forall r.B,$$

where A, B, A_1, A_2 are concept names and r, s are roles. It is well-known that every Horn- \mathcal{ALCHIF} TBox \mathcal{T} can be converted into a TBox \mathcal{T}' in normal form (introducing additional concept names) such that \mathcal{T} is a logical consequence of \mathcal{T}' and every model of \mathcal{T} can be extended to one of \mathcal{T}' by interpreting the additional concept names, see e.g. [BHLW16]. As a consequence, all results obtained in this chapter for TBoxes in normal form lift to the general case.

4.2 Query Conservative Extensions and Entailment

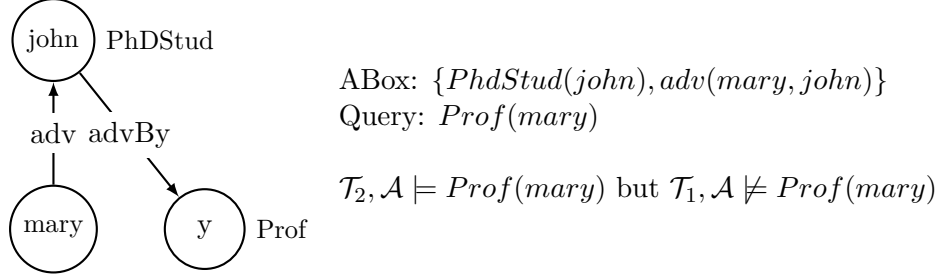
We introduce the central notions studied in this chapter.

Definition 4.2.1. Let Γ, Σ be signatures and $\mathcal{T}_1, \mathcal{T}_2$ Horn- \mathcal{ALCHIF} TBoxes. We say that \mathcal{T}_1 (Γ, Σ)-CQ entails \mathcal{T}_2 , written $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$, if for all Γ -ABoxes \mathcal{A} consistent with \mathcal{T}_1 and \mathcal{T}_2 , all Σ -CQs $q(\mathbf{x})$ and all tuples $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$, $\mathcal{T}_2, \mathcal{A} \models q(\mathbf{a})$ implies $\mathcal{T}_1, \mathcal{A} \models q(\mathbf{a})$. If in addition $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$, we say that \mathcal{T}_2 is a (Γ, Σ)-CQ conservative extension of \mathcal{T}_1 . If $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ and vice versa, then \mathcal{T}_1 and \mathcal{T}_2 are (Γ, Σ)-CQ inseparable.

If $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ because $\mathcal{T}_2, \mathcal{A} \models q(\mathbf{a})$ but $\mathcal{T}_1, \mathcal{A} \not\models q(\mathbf{a})$ for some Γ -ABox \mathcal{A} consistent with both \mathcal{T}_i , Σ -CQ $q(\mathbf{x})$ and \mathbf{a} , we call the triple $(\mathcal{A}, q, \mathbf{a})$ a *witness* to non-entailment.

¹E.g., out of 439 available ontologies in BioPortal [MP17], only 21 ($\leq 4.8\%$) contain the described pattern. A significant fraction of the occurrences of the pattern appear to be due to modeling mistakes.

Example 4.2.1. Let $\mathcal{T}_1 = \{\text{PhDStud} \sqsubseteq \exists \text{advBy}.\text{Prof}, \text{adv} \sqsubseteq \text{advBy}^-\}$ and $\mathcal{T}_2 = \mathcal{T}_1 \cup \{\text{func}(\text{advBy})\}$, $\Sigma = \{\text{Prof}\}$ and $\Gamma = \{\text{PhDStud}, \text{adv}\}$. Then $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ because of the witness $(\{\text{PhDStud}(\text{john}), \text{adv}(\text{mary}, \text{john})\}, \text{Prof}(x), \text{mary})$ as illustrated below.



A CQ q is *tree-shaped* if it does not contain atoms of the form $r(x, x)$ and the undirected graph $(\mathbf{x} \cup \mathbf{y}, \{\{v, w\} \mid r(v, w) \text{ is an atom in } q\})$ is a tree; tree-shaped CQs are thus connected and may contain multi-edges. A tree-shaped CQ q is *strongly tree-shaped* or an *stCQ* if the root is the one and only answer variable and q has no multi-edges, that is, for any distinct variables z, z' in q , there is at most one role atom that contains both z and z' .

In addition to the notions introduced in Definition 4.2.1, we also consider (Γ, Σ) -*stCQ entailment*, denoted $\models_{\Gamma, \Sigma}^{\text{stCQ}}$ and defined in the obvious way by replacing CQs with stCQs.

4.2.1 Query Entailment with Inconsistent ABoxes

If we drop from Definition 4.2.1 the condition that \mathcal{A} must be consistent with both \mathcal{T}_1 and \mathcal{T}_2 , then we obtain an alternative notion of CQ entailment that we call *CQ entailment with inconsistent ABoxes*. While this new notion trivially implies CQ entailment in the original sense, the converse fails.

Example 4.2.2. Let $\mathcal{T}_1 = \emptyset$, $\mathcal{T}_2 = \{A_1 \sqcap A_2 \sqsubseteq \perp\}$ and $\Gamma = \{A_1, A_2\}$, $\Sigma = \{B\}$. Then $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ but \mathcal{T}_1 does not (Γ, Σ) -CQ entail \mathcal{T}_2 with inconsistent ABoxes, witnessed by $(\{A_1(a), A_2(a)\}, \exists y r(y, y), a)$.

We need to deal with the question whether any Γ -ABox that is inconsistent with some TBox \mathcal{T}_2 is inconsistent with another TBox \mathcal{T}_1 . We say that \mathcal{T}_1 Γ -*inconsistency entails* \mathcal{T}_2 , written $\mathcal{T}_1 \models_{\Gamma}^{\perp} \mathcal{T}_2$, if for all Γ -ABoxes \mathcal{A} : if \mathcal{A} is inconsistent with \mathcal{T}_2 , then \mathcal{A} is inconsistent with \mathcal{T}_1 .

The following lemma relates the two notions of CQ entailment.

Lemma 4.2.1. *CQ entailment with inconsistent ABoxes can be decided in polynomial time given access to oracles deciding CQ entailment and CQ evaluation.*

Before proving Lemma 4.2.1 we give some preliminaries. For interpretations $\mathcal{I}_1, \mathcal{I}_2$ and a signature Σ , a Σ -*homomorphism* from \mathcal{I}_1 to \mathcal{I}_2 is a total function $h : \Delta^{\mathcal{I}_1} \rightarrow \Delta^{\mathcal{I}_2}$ such that (1) $h(a) = a$ for all $a \in \mathbf{N}_1$, (2) $h(d) \in A^{\mathcal{I}_2}$ for all $d \in A^{\mathcal{I}_1}$, $A \in \mathbf{N}_C \cap \Sigma$, and (3) $(h(d), h(d')) \in r^{\mathcal{I}_2}$ for all $(d, d') \in r^{\mathcal{I}_1}$, $r \in \mathbf{N}_R \cap \Sigma$. If there is a Σ -homomorphism from \mathcal{I}_1 to \mathcal{I}_2 , we write $\mathcal{I}_1 \rightarrow_{\Sigma} \mathcal{I}_2$.

Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox in normal form and \mathcal{A} an ABox consistent with \mathcal{T} . A *type for \mathcal{T}* is a set $t \subseteq \text{sub}(\mathcal{T}) \cap \mathbf{N}_C$ such that $\mathcal{T} \models \bigcap t \sqsubseteq A$ implies $A \in t$ for all concept names A . For $a \in \text{ind}(\mathcal{A})$, let $\text{tp}_{\mathcal{T}}(a) = \{A \mid \mathcal{T}, \mathcal{A} \models A(a)\}$ be the *type of a relative to \mathcal{T}* . When $a \in \text{ind}(\mathcal{A})$, t, t' are types for \mathcal{T} , and r is a role, we write

- $a \rightsquigarrow_r^{\mathcal{T}, \mathcal{A}} t$ if $\mathcal{T}, \mathcal{A} \models \exists r. \sqcap t(a)$ and t is maximal with this condition, and
- $t \rightsquigarrow_r^{\mathcal{T}} t'$ if $\mathcal{T} \models \sqcap t \sqsubseteq \exists r. \sqcap t'$ and t' is maximal with this condition.

A *path* for \mathcal{A} and \mathcal{T} is a finite sequence $\pi = ar_0t_1 \cdots t_{n-1}r_{n-1}t_n$, $n \geq 0$, with $a \in \text{ind}(\mathcal{A})$, r_0, \dots, r_{n-1} roles, and t_1, \dots, t_n types for \mathcal{T} such that

- $a \rightsquigarrow_{r_0}^{\mathcal{T}, \mathcal{A}} t_1$ and, if $\text{func}(r_0) \in \mathcal{T}$, then there is no $b \in \text{ind}(\mathcal{A})$ such that $\mathcal{T}, \mathcal{A} \models r_0(a, b)$;
- for every $1 \leq i < n$, we have $t_i \rightsquigarrow_{r_i}^{\mathcal{T}} t_{i+1}$ and, if $\text{func}(r_i) \in \mathcal{T}$, then $r_{i-1} \neq r_i^-$.

When $n > 0$, we use $\text{tail}(\pi)$ to denote t_n . Let Paths be the set of all paths for \mathcal{A} and \mathcal{T} . The *universal model* $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ of \mathcal{T} and \mathcal{A} is defined as follows:

$$\begin{aligned} \Delta^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}} &= \text{Paths} \\ A^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}} &= \{a \in \text{ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \cup \\ &\quad \{\pi \in \Delta^{\mathcal{I}} \setminus \text{ind}(\mathcal{A}) \mid \mathcal{T} \models \sqcap \text{tail}(\pi) \sqsubseteq A\} \\ r^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}} &= \{(a, b) \in \text{ind}(\mathcal{A})^2 \mid s(a, b) \in \mathcal{A}, \mathcal{T} \models s \sqsubseteq r\} \cup \\ &\quad \{(\pi, \pi st) \mid \pi st \in \text{Paths} \text{ and } \mathcal{T} \models s \sqsubseteq r\} \cup \\ &\quad \{(\pi st, \pi) \mid \pi st \in \text{Paths} \text{ and } \mathcal{T} \models s^- \sqsubseteq r\} \end{aligned}$$

We also need universal models of a TBox \mathcal{T} and a type t , instead of an ABox. More precisely, we define $\mathcal{I}_{\mathcal{T}, t} = \mathcal{I}_{\mathcal{T}, \mathcal{A}_t}$ where $\mathcal{A}_t = \{A(a) \mid A \in t\}$ for a fixed $a \in \mathbb{N}_I$. If an interpretation \mathcal{I} is a common model of a TBox \mathcal{T} and ABox \mathcal{A} , then we also write $\mathcal{I} \models (\mathcal{T}, \mathcal{A})$ and call \mathcal{I} a *model of* $(\mathcal{T}, \mathcal{A})$. The following is standard to prove [BO15].

Lemma 4.2.2. *For every Horn-ALCHIF TBox \mathcal{T} in normal form and ABox \mathcal{A} consistent with \mathcal{T} , the following hold:*

- (1) $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models (\mathcal{T}, \mathcal{A})$.
- (2) For all models \mathcal{I} of $(\mathcal{T}, \mathcal{A})$, we have $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \rightarrow \mathcal{I}$.
- (3) For all types t, t' for \mathcal{T} with $t \sqsubseteq t'$, we have $\mathcal{I}_{\mathcal{T}, t} \rightarrow \mathcal{I}_{\mathcal{T}, t'}$.
- (4) $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ iff $\mathcal{I}_{\mathcal{T}, \mathcal{A}} \models q(\mathbf{a})$, for all CQs $q(\mathbf{x})$ and tuples \mathbf{a} of individuals.

Now we are ready to prove Lemma 4.2.1. We proceed in two steps: first, we show how to deal with inconsistency entailment; second we show how to use this type of entailment to deal with CQ entailment with inconsistent ABoxes.

For the first step, we can reduce inconsistency entailment to CQ entailment because, if \mathcal{A} is inconsistent with \mathcal{T} , then either (a) $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ contains a B -instance for some B with $B \sqsubseteq \perp \in \mathcal{T}$, or (b) \mathcal{A} contains a “fork” $\{r(a, b), r(a, c)\}$ that is prohibited by $\text{func}(r) \in \mathcal{T}$. We write $\mathcal{T}_1 \models_{\Gamma}^{\text{fork}} \mathcal{T}_2$ if for all Γ -ABoxes $\mathcal{A} = \{r(a, b), r(a, c)\}$: if \mathcal{A} is inconsistent with \mathcal{T}_2 , then also with \mathcal{T}_1 . Thus, if we have a witness ABox \mathcal{A} for $\mathcal{T}_1 \not\models_{\Gamma}^{\perp} \mathcal{T}_2$, then \mathcal{A} is inconsistent with \mathcal{T}_2 either by Case (a) – which we can detect via CQ entailment if we allow a fresh concept name in the CQ and slightly modify the TBoxes – or by Case (b), which implies $\mathcal{T}_1 \not\models_{\Gamma}^{\text{fork}} \mathcal{T}_2$.

Lemma 4.2.3. *Let Γ be a signature and let \mathcal{T}_1 and \mathcal{T}_2 be Horn-ALCHIF TBoxes. Furthermore, let A be a fresh concept name and \mathcal{T}_i^A be obtained from \mathcal{T}_i by replacing each occurrence of \perp with A and adding the axioms $\exists s.A \sqsubseteq A$ and $\exists s^-.A \sqsubseteq A$ for every role s occurring in \mathcal{T}_i , for $i = 1, 2$. Then the following are equivalent.*

- (1) $\mathcal{T}_1 \models_{\Gamma}^{\perp} \mathcal{T}_2$
- (2) $\mathcal{T}_1^A \models_{\Gamma, \{A\}}^{\text{CQ}} \mathcal{T}_2^A$ and $\mathcal{T}_1 \models_{\Gamma}^{\text{fork}} \mathcal{T}_2$
- (3) $\mathcal{T}_1^A \models_{\Gamma, \{A\}}^{\text{stCQ}} \mathcal{T}_2^A$ and $\mathcal{T}_1 \models_{\Gamma}^{\text{fork}} \mathcal{T}_2$

Note that we only need the equivalence between (1) and (2) to prove Lemma 4.2.1. However, we will need (3) later to prove Lemma 4.5.1. Indeed, (2) and (3) are obviously equivalent given the primitive query signature $\{A\}$ and the propagation of A throughout the \mathcal{T}_i^A .

Proof.

“1 \Rightarrow 2”. We prove the contrapositive. Assume $\mathcal{T}_1^A \not\models_{\Gamma, \{A\}}^{\text{CQ}} \mathcal{T}_2^A$ or $\mathcal{T}_1 \not\models_{\Gamma}^{\text{fork}} \mathcal{T}_2$. In case $\mathcal{T}_1 \not\models_{\Gamma}^{\text{fork}} \mathcal{T}_2$, every witness ABox is a witness for $\mathcal{T}_1 \not\models_{\Gamma}^{\perp} \mathcal{T}_2$ too.

In case $\mathcal{T}_1^A \not\models_{\Gamma, \{A\}}^{\text{CQ}} \mathcal{T}_2^A$ is violated, consider a witness $(\mathcal{A}, q, \mathbf{a})$. Since A is the only symbol allowed in q , all atoms of q have the form $A(z)$ for arbitrary variables z . If q consists of several atoms, then it is disconnected and we can omit all but one atom from q and still have a witness (see also proof of Lemma 4.3.2, Property d). Hence we can assume w.l.o.g. that q is of the form (i) $q(x) = A(x)$ or (ii) $q() = \exists y A(y)$ and, furthermore, that \mathcal{A} and thus the universal models $\mathcal{I}_{\mathcal{T}_i, \mathcal{A}}$ are connected. (Due to the “propagation” of A in the \mathcal{T}_i , we can even assume that q is of the form (i) only, but that does not matter in the following argumentation.) We now have:

- \mathcal{A} is inconsistent with \mathcal{T}_2 :

Assume to the contrary that \mathcal{A} is consistent with \mathcal{T}_2 and consider the universal model $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ for \mathcal{T}_2 and \mathcal{A} (Section ??). Clearly, for all domain elements d of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, we have $\mathcal{T}_2 \not\models \prod \text{tp}_{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}}(d) \sqsubseteq \perp$. Since A is fresh and by the definition of \mathcal{T}_2^A we get $\mathcal{T}_2^A \not\models \prod \text{tp}_{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}}(d) \sqsubseteq A$. Now Lemma 4.2.2 (1) for \mathcal{T}_2^A implies that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models (\mathcal{T}_2, \mathcal{A})$; hence $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ satisfies all axioms in \mathcal{T}_2^A that have been taken over from \mathcal{T}_2 without modification, i.e., all axioms that are not of the form $B \sqsubseteq A$. But axioms of the latter form are also satisfied because $\mathcal{T}_2^A \not\models \prod \text{tp}_{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}}(d) \sqsubseteq A$ for every domain element d . Hence $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models (\mathcal{T}_2^A, \mathcal{A})$. Now, since $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ has no A -instance, we cannot have $\mathcal{T}_2^A, \mathcal{A} \models q(\mathbf{a})$ for any $\{A\}$ -query q ; contradicting the assumption that $(\mathcal{A}, q, \mathbf{a})$ is a witness.

- \mathcal{A} is consistent with \mathcal{T}_1 :

Since $(\mathcal{A}, q, \mathbf{a})$ is a witness, we have $\mathcal{I}_{\mathcal{T}_1^A, \mathcal{A}} \not\models q(\mathbf{a})$ by Lemma 4.2.2 (4). Due to the additional axioms in the definition of \mathcal{T}_1^A , which “propagate” A into every domain element of the connected (see above) universal model $\mathcal{I}_{\mathcal{T}_1^A, \mathcal{A}}$, we have $\mathcal{T}_1^A \not\models \prod \text{tp}_{\mathcal{I}_{\mathcal{T}_1^A, \mathcal{A}}}(d) \sqsubseteq A$ for all domain elements d . Since A is fresh, we have $\mathcal{T}_1 \not\models \prod \text{tp}_{\mathcal{I}_{\mathcal{T}_1^A, \mathcal{A}}}(d) \sqsubseteq \perp$. With the same reasoning as above, we get $\mathcal{I}_{\mathcal{T}_1^A, \mathcal{A}} \models (\mathcal{T}_1, \mathcal{A})$; hence \mathcal{A} is consistent with \mathcal{T}_1 .

Consequently $\mathcal{T}_1 \not\models_{\Gamma}^{\perp} \mathcal{T}_2$, as desired.

“2 \Rightarrow 3”. This is immediate because $\mathcal{T}_1^A \models_{\Gamma, \{A\}}^{\text{CQ}} \mathcal{T}_2^A$ implies $\mathcal{T}_1^A \models_{\Gamma, \{A\}}^{\text{stCQ}} \mathcal{T}_2^A$.

“3 \Rightarrow 1”. We prove the contrapositive. Assume $\mathcal{T}_1 \not\models_{\Gamma}^{\perp} \mathcal{T}_2$, i.e., there is a Γ -ABox \mathcal{A} that is inconsistent with \mathcal{T}_2 but consistent with \mathcal{T}_1 . We need to show that $\mathcal{T}_1^A \not\models_{\Gamma, \{A\}}^{\text{stCQ}} \mathcal{T}_2^A$ or $\mathcal{T}_1 \not\models_{\Gamma}^{\text{fork}} \mathcal{T}_2$.

From \mathcal{A} being inconsistent with \mathcal{T}_2 , we first conclude that one of the following two properties must hold.

(i) There is some $d \in B^{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}}$ with $B \sqsubseteq \perp \in \mathcal{T}_2$.

(ii) \mathcal{A} contains a “fork” $\mathcal{A}^- = \{r(a, b), r(a, c)\}$ such that \mathcal{A}^- is inconsistent with \mathcal{T}_2 .

Indeed, if neither (i) nor (ii) holds, then we have $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models (\mathcal{T}_2, \mathcal{A})$, contradicting the inconsistency of \mathcal{A} with \mathcal{T}_2 : First, $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models \mathcal{A}$ follows directly from the construction of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$. Second, $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models \mathcal{T}_2$ can be shown analogously to the (omitted) standard proof of Lemma 4.2.2 (1), via a case distinction over the axioms in \mathcal{T}_2 , using “not (i)” and “not (ii)” instead of the assumption that \mathcal{A} is consistent with \mathcal{T}_2 .

Now first assume that (ii) holds. Since \mathcal{A} is consistent with \mathcal{T}_1 , so is \mathcal{A}^- . Hence $\mathcal{T}_1 \not\models_{\Gamma}^{\text{fork}} \mathcal{T}_2$.

In case (ii) does not hold, (i) must hold. To show that $\mathcal{T}_1^A \not\models_{\Gamma, \{A\}}^{\text{stCQ}} \mathcal{T}_2^A$, consider the stCQ $q = A(x)$ and some $a \in \text{ind}(\mathcal{A})$ to which the element d from (i) is connected in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, i.e., if $d \in \text{ind}(\mathcal{A})$, then choose $a = d$; otherwise choose a such that d is in the subtree $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_a$. We then have:

- \mathcal{A} is consistent with \mathcal{T}_2^A :

Since \mathcal{T}_2^A does not contain \perp and \mathcal{A} does not contain forks as in (ii), \mathcal{A} is consistent with \mathcal{T}_2^A is consistent, as witnessed by the universal model $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}}$ (we again refer to the standard proof of Lemma 4.2.2 (1); except that the FA case in the ABox part of $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}}$ is now due to “not (ii)”).

- \mathcal{A} is consistent with \mathcal{T}_1^A :

It is not difficult to see that $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \models (\mathcal{T}_1^A, \mathcal{A})$: From $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \models (\mathcal{T}_1, \mathcal{A})$, it follows that $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ is a model of \mathcal{A} and satisfies all axioms in \mathcal{T}_1^A that \mathcal{T}_1^A shares with \mathcal{T}_1 . The modified axioms $B \sqsubseteq A$ with $B \sqsubseteq \perp \in \mathcal{T}_1$ are satisfied, too, because $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ cannot have any B -instances. Finally, the additional propagation axioms are satisfied because $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ has no A -instance as A is fresh.

- $\mathcal{T}_2^A, \mathcal{A} \models q(a)$:

Due to (i), we have $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models \exists y B(y)$ for some $B \sqsubseteq \perp \in \mathcal{T}_2$. Hence $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}} \models \exists y B(y)$, which follows from the construction of both universal models (in fact the only difference between $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}}$ and $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ is that some domain elements of $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}}$ may be A -instances). Hence $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}}$ has a B -instance in the subtree $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_a$ and thus, by construction, an A -instance. By the “propagation” of A in \mathcal{T}_2^A , we have that a is an instance of A in $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}}$; hence $\mathcal{I}_{\mathcal{T}_2^A, \mathcal{A}} \models A(a) = q$.

- $\mathcal{T}_1^A, \mathcal{A} \not\models q(a)$:

Follows from $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \models (\mathcal{T}_1^A, \mathcal{A})$ (as shown above) and $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q(a)$ (given the lack of A -instances). □

Proposition 4.2.4. *Fork entailment $\mathcal{T}_1 \models_{\Gamma}^{\text{fork}} \mathcal{T}_2$ can be (Turing) reduced in polynomial time to stCQ evaluation.*

Proof. Perform $2|\Gamma|$ many ABox consistency checks by evaluating the stCQ $A(a)$ on both \mathcal{T}_i , where A is a concept name that does not occur in any of the \mathcal{T}_i . □

For the second step, we can now reduce CQ entailment with inconsistent ABoxes to the disjunction of our original notion of CQ entailment and inconsistency entailment. We need an additional notion: Given a TBox \mathcal{T} and signatures Γ, Σ , we say that \mathcal{T} is (Γ, Σ) -universal if

- (*) for all Γ -ABoxes \mathcal{A} and Σ -CQs $q(\mathbf{x})$ and all tuples $\mathbf{a} \subseteq \text{ind}(\mathcal{A})$ with $|\mathbf{a}| = |\mathbf{x}|$, we have $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$.

Lemma 4.2.5. *Let Γ, Σ be signatures and let \mathcal{T}_1 and \mathcal{T}_2 be Horn- \mathcal{ALCHIF} TBoxes. Then \mathcal{T}_1 (Γ, Σ) -CQ entails \mathcal{T}_2 with inconsistent ABoxes iff one of the two following conditions holds.*

- (1) $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ and $\mathcal{T}_1 \models_{\Gamma}^{\perp} \mathcal{T}_2$
(2) \mathcal{T}_1 is (Γ, Σ) -universal.

Proof. We prove both implications via contraposition.

“ \Rightarrow ”. Assume (1) and (2) are both false, i.e., \mathcal{T}_1 is not (Γ, Σ) -universal and either (a) $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ or (b) $\mathcal{T}_1 \not\models_{\Gamma}^{\perp} \mathcal{T}_2$. In case (a), \mathcal{T}_1 trivially does not (Γ, Σ) -CQ entail \mathcal{T}_2 with inconsistent ABoxes. In case (b), consider a witness Γ -ABox \mathcal{A} . Since \mathcal{T}_1 is not (Γ, Σ) -universal, there is a Γ -ABox \mathcal{A}' , a Σ -CQ $q(\mathbf{x})$ and a tuple $\mathbf{a} \subseteq \text{ind}(\mathcal{A}')$ with $|\mathbf{a}| = |\mathbf{x}|$ such that $\mathcal{T}_1, \mathcal{A}' \not\models q(\mathbf{a})$. We assume w.l.o.g. that \mathcal{A} and \mathcal{A}' use distinct sets of individuals. We set $\mathcal{A}'' = \mathcal{A} \cup \mathcal{A}'$ and have:

- $\mathcal{T}_2, \mathcal{A}'' \models q(\mathbf{a})$ because \mathcal{A} is inconsistent with \mathcal{T}_2 and so is \mathcal{A}'' .
- $\mathcal{T}_1, \mathcal{A}'' \not\models q(\mathbf{a})$: let \mathcal{I} be the disjoint union of the universal model $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ and the model \mathcal{I} witnessing $\mathcal{T}_1, \mathcal{A}' \not\models q(\mathbf{a})$. Clearly $\mathcal{I} \models (\mathcal{T}_1, \mathcal{A}'')$ but $\mathcal{I} \not\models q(\mathbf{a})$.

Hence \mathcal{T}_1 does not (Γ, Σ) -CQ entail \mathcal{T}_2 with inconsistent ABoxes, as desired.

“ \Leftarrow ”. Assume \mathcal{T}_1 does not (Γ, Σ) -CQ entail \mathcal{T}_2 with inconsistent ABoxes and consider a witness $(\mathcal{A}, q, \mathbf{a})$. Then it is immediate that (2) does not hold. Furthermore, if \mathcal{A} is consistent with both \mathcal{T}_1 and \mathcal{T}_2 , then $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$. Otherwise \mathcal{A} must be inconsistent with \mathcal{T}_2 but consistent with \mathcal{T}_1 ; hence $\mathcal{T}_1 \not\models_{\Gamma}^{\perp} \mathcal{T}_2$. Therefore (1) does not hold either. \square

Proposition 4.2.6. *(Γ, Σ) -universality can be (Turing) reduced in polynomial time to stCQ evaluation.*

Proof. It suffices to check Condition (*) above (i) for all singleton Γ -ABoxes $\{A(a)\}$ and all single-atom Σ -CQs $B(x)$ or $r(x, x)$, and (ii) for all two-element Γ -ABoxes $\{r(a, b)\}$ and all Σ -CQs as in (i) but with possibly two distinct answer variables. \square

Lemma 4.2.1 is now a direct consequence of Lemmas 4.2.3 and 4.2.5, and Propositions 4.2.4 and 4.2.6. Consequently and since CQ evaluation is in EXPTIME in Horn- \mathcal{ALCHIF} [EGOŠ08], all complexity results obtained in this chapter also apply to CQ entailment with inconsistent ABoxes.

4.3 Model-theoretic Characterization

We aim to provide a model-theoretic characterization of query entailment that will be the basis for our decision procedure later on. The first step towards this characterization consists in showing that non-entailment is always witnessed by tree-shaped ABoxes and tree-shaped CQs with at most one answer variable. Here, an ABox \mathcal{A} is *tree-shaped* if it does not contain an assertion of the form $r(a, a)$, the undirected graph $G_{\mathcal{A}} = (\text{ind}(\mathcal{A}), \{\{a, b\} \mid r(a, b) \in \mathcal{A}\})$ is a tree, and for any $a, b \in \text{ind}(\mathcal{A})$, \mathcal{A} contains at most one role assertion that involves both a and b .

4.3.1 Unraveling ABoxes

To obtain tree-shaped ABoxes or CQs, we use unraveling, which needs to be more cautious in the presence of inverse roles and functionality. In particular, we need to ensure that, whenever a role is functional in an ABox, then so it is in its unraveling. We define an unraveling for Horn- \mathcal{ALCHIF} similar to the one for Horn- \mathcal{ALCII} in [LW12].

Let \mathcal{A} be an ABox. The unraveling $U_{\mathcal{A}}^a$ of \mathcal{A} at an individual $a \in \text{ind}(\mathcal{A})$ is the following ABox:

- $\text{ind}(U_{\mathcal{A}}^a)$ is the set of sequences $b_0 r_0 b_1 \cdots r_{n-1} b_n$ with $n \geq 0$, where $b_0 = a$, $b_i \in \text{ind}(\mathcal{A})$ for all $0 \leq i \leq n$, $r_i(b_i, b_{i+1}) \in \mathcal{A}$ for all $0 \leq i < n$, and $(b_{i-1}, r_{i-1}^-) \neq (b_{i+1}, r_i)$ (the latter inequality is needed to ensure preservation of functionality).
- The concept assertions in $U_{\mathcal{A}}^a$ are all assertions of the shape $C(\alpha)$ such that $\alpha = b_0 \cdots b_{n-1} r_{n-1} b_n \in \text{ind}(\mathcal{A})$ and $C(b_n) \in \mathcal{A}$. The role assertions in $U_{\mathcal{A}}^a$ are all assertions of the shape $r(b_0 \cdots b_{n-1}, \alpha)$ such that $\alpha = b_0 \cdots b_{n-1} r_{n-1} b_n \in \text{ind}(\mathcal{A})$.

The following is standard to prove [LW17, LW12]:

Proposition 4.3.1. *Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox, \mathcal{A} an ABox, and $a \in \text{ind}(\mathcal{A})$. If \mathcal{A} is consistent with \mathcal{T} , then so is $U_{\mathcal{A}}^a$.*

It is easy to see that $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ if there is a Γ -role r and a Σ -role s with $\mathcal{T}_2 \models r \sqsubseteq s$ but $\mathcal{T}_1 \not\models r \sqsubseteq s$. We write $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$ if there are no such r and s . Clearly, $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$ can be decided via $|\Gamma| \cdot |\Sigma|$ many Horn- \mathcal{ALCHIF} subsumption tests, thus in EXPTIME [Tob01]. It is thus safe to assume $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$ when deciding CQ entailment, which we will generally do from now on to avoid dealing with special cases.

The following result shows that non-entailment is always witnessed by tree-shaped ABoxes and tree-shaped CQs with at most one answer variable.

Lemma 4.3.2. *Let \mathcal{T}_1 and \mathcal{T}_2 be Horn- \mathcal{ALCHIF} TBoxes with $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$. If $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$, then there is a witness $(\mathcal{A}, q, \mathbf{a})$ where \mathcal{A} and q are tree-shaped and $|\mathbf{a}| \leq 1$, i.e., q has at most one answer variable. If $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$, then there is such a witness where additionally q is an stCQ.*

We reformulate the lemma to make its statement more explicit.

Lemma 4.3.2, reformulated equivalently. *Let $\mathcal{T}_1, \mathcal{T}_2$ be Horn- \mathcal{ALCHIF} TBoxes with $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$. If $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$, then there is a tree-shaped Γ -ABox \mathcal{A} consistent with \mathcal{T}_1 and \mathcal{T}_2 , and a tree-shaped Σ -CQ q such that one of the following holds:*

- (1) q has a single answer variable and there is an $a \in \text{ind}(\mathcal{A})$ such that $\mathcal{T}_2, \mathcal{A} \models q(a)$ but $\mathcal{T}_1, \mathcal{A} \not\models q(a)$;

(2) q is Boolean and $\mathcal{T}_2, \mathcal{A} \models q$ but $\mathcal{T}_1, \mathcal{A} \not\models q$.

If $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$, then there is a tree-shaped Γ -ABox \mathcal{A} and a tree-shaped Σ -stCQ q with (1).

Proof. Unrestricted CQs. Assume $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$, i.e., $\mathcal{T}_2, \mathcal{A} \models q(\mathbf{a})$ and $\mathcal{T}_1, \mathcal{A} \not\models q(\mathbf{a})$, for some Γ -ABox \mathcal{A} consistent with both \mathcal{T}_i , some Σ -CQ q and some tuple \mathbf{a} . Lemma 4.2.2 (4) yields $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q(\mathbf{a})$ and $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q(\mathbf{a})$. We first show that the following properties of q and \mathbf{a} are without loss of generality:

- (a) Every match of $q(\mathbf{x})$ into $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ maps every quantified variable into the anonymous part.
- (b) $q(\mathbf{x})$ does not contain atoms of the form $r(x_1, x_2)$ with x_1, x_2 answer variables.
- (c) If $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{a} = (a_1, \dots, a_n)$, then $a_i \neq a_j$ for all i, j with $1 \leq i < j \leq n$.
- (d) $q(\mathbf{x})$ is connected.

For (a), take a match π of q in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ and a quantified variable y such that $\pi(y) = b \in \text{ind}(\mathcal{A})$. Obtain $q'(\mathbf{x}, y)$ from $q(\mathbf{x})$ by removing the quantification over y , thus making y an answer variable. Clearly, we have $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q'(\mathbf{a}, b)$ and $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q'(\mathbf{a}, b)$, and thus $\mathcal{T}_2, \mathcal{A} \models q'(\mathbf{a}, b)$ and $\mathcal{T}_1, \mathcal{A} \not\models q'(\mathbf{a}, b)$.

For (b), observe that such atoms can always be dropped, since they cannot be inferred via \mathcal{T}_1 or \mathcal{T}_2 : Let $q(\mathbf{x}) = \exists \mathbf{y} (r(x_1, x_2) \wedge \varphi(\mathbf{x}', \mathbf{y}))$ with $x_1, x_2 \in \mathbf{x}$, and let $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q(\mathbf{a})$ be witnessed by the match π with $\pi(x_i) = a_i$, $i = 1, 2$. Construct the CQ $q(\mathbf{x}') = \exists \mathbf{y} \varphi(\mathbf{x}', \mathbf{y})$ by dropping the atom $r(x_1, x_2)$ (and thus possibly removing x_1 and/or x_2 from the free variables). It is clear that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q'(\mathbf{a}')$ for the corresponding restriction \mathbf{a}' of the tuple \mathbf{a} ; thus it suffices to show that $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q'(\mathbf{a}')$.

From $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q(\mathbf{a})$ we can conclude that $(a_1, a_2) \in r^{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}}$. By construction of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ there is some Γ -role r' with $r'(a_1, a_2) \in \mathcal{A}$ and $\mathcal{T}_2 \models r' \sqsubseteq r$ (which includes the possibility $r' = r$, i.e., $r(a_1, a_2) \in \mathcal{A}$). Due to $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$, we also have $\mathcal{T}_1 \models r' \sqsubseteq r$ and hence $(a_1, a_2) \in r^{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}}$. This implies the desired $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q'(\mathbf{a}')$ because, otherwise, any match π of q' in $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ with $\pi(x_i) = a_i$, $i = 1, 2$, could be extended to a match of q .

This construction does not introduce any violations of (a).

For (c), observe that, whenever $a_i = a_j$ for some i, j with $1 \leq i < j \leq n$, we can always drop x_j and a_j : Let \mathbf{x}' and \mathbf{a}' be \mathbf{x} and \mathbf{a} with x_j and a_j removed, and transform $q(\mathbf{x})$ into $q'(\mathbf{x}')$ by replacing every occurrence of x_j with x_i . Now $\mathcal{T}_2, \mathcal{A} \models q'(\mathbf{a}')$ and $\mathcal{T}_1, \mathcal{A} \not\models q'(\mathbf{a}')$. This construction does not introduce any violations of (a) or (b).

For (d), observe that $\mathcal{T}_2, \mathcal{A} \models q(\mathbf{a})$ and $\mathcal{T}_1, \mathcal{A} \not\models q(\mathbf{a})$ implies $\mathcal{T}_2, \mathcal{A} \models q'(\mathbf{a})$ and $\mathcal{T}_1, \mathcal{A} \not\models q'(\mathbf{a})$ for some connected component q' of q . This construction does not introduce any violations of (a), (b), or (c). While this is easy to see for (b) and (c), Property (a) requires a closer look: If the possibly disconnected CQ q satisfies (a) and has at least one match π in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, then every match of any connected component q' in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ can be extended to a match of q in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ via π restricted to the remaining connected components. Since the match of q satisfies (a), so does the match of q' .

Thus, as long as q violates any of the above properties, we apply the corresponding modification as described and in the order given. From now on, we assume that q satisfies properties (a) to (d). Furthermore, they imply:

- (e) $q(\mathbf{x})$ does *not* contain a *proper path* between any two answer variables, which is a non-empty sequence of atoms $r_1(z_1, z_2), r_2(z_2, z_3), \dots, r_n(z_n, z_{n+1})$ with variables $z_1, z_{n+1} \in \mathbf{x}$ and $z_i \in \mathbf{y}$ for $1 < i \leq n$, and with roles r_i such that $z_{i+1} \neq z_{i-1}$ for every $1 < i \leq n$.

To show this, assume the opposite, i.e., $q(\mathbf{x})$ contains a proper path as above between two answer variables x, x' . By (b) we have $n > 1$. By (a) and (c), π maps all z_i with $1 < i \leq n$ to the anonymous part of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$. However, there is no corresponding proper path between any two ABox individuals in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$; a contradiction.

Assume now that $q(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ is not tree-shaped, i.e., assume there is a cycle $r_1(z_1, z_2), r_2(z_2, z_3), \dots, r_n(z_n, z_{n+1})$ with variables $z_i \in \mathbf{x} \cup \mathbf{y}$, $z_1 = z_{n+1}$, and roles r_i such that $z_{i+1} \neq z_{i-1}$ for every $1 < i \leq n$ and $z_2 \neq z_n$. By (e), we have $z_i \in \mathbf{y}$ for all $1 \leq i \leq n+1$. Let π be a match of $q(\mathbf{x})$ in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$. By (a), π maps all variables to the anonymous part of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ which, by construction, is acyclic. Hence π cannot satisfy the properties of a match; contradiction.

Assume now that \mathbf{x} in $q(\mathbf{x})$ contains more than one answer variable, say $x \neq x'$, matched by a and a' in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, with $a \neq a'$ due to (c). By (d), q is connected, and thus, there is a path from x to x' in q . Since $x \neq x'$, there is even a proper path; contradicting (e).

Thus, we now have that q is tree-shaped and behaves as required by (1) or (2). It remains to transform \mathcal{A} into a tree-shaped ABox: In case q is Boolean, we get from (a) and (c) that every match of q in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ is into the anonymous subtree rooted at some ABox individual a ; in case q has one answer variable, let $\mathbf{a} = a$. Consider the unraveling $U_{\mathcal{A}}^a$ of \mathcal{A} at a . Clearly, $\mathcal{T}_2, U_{\mathcal{A}}^a \models q(a)$ and $\mathcal{T}_1, U_{\mathcal{A}}^a \not\models q(a)$, which is still consistent with both \mathcal{T}_i , due to Proposition 4.3.1. By compactness, there is a finite subset $\mathcal{B} \subseteq U_{\mathcal{A}}^a$ with $\mathcal{T}_1, \mathcal{B} \models q(a)$ and $\mathcal{T}_2, \mathcal{B} \not\models q(a)$. Clearly, we can also assume that \mathcal{B} is connected.

stCQs. Since stCQs are already tree-shaped and have exactly one answer variable, the previous argument for unrestricted CQs reduces to observing Properties (a) and (c) and unraveling the witness ABox as described. \square

4.3.2 Characterization of Query Entailment

Our goal is to characterize query entailment in terms of homomorphisms between universal models. Homomorphisms are natural because answers to CQs are preserved under homomorphisms (both on interpretations and on ABoxes). In fact, they are preserved even under bounded homomorphisms if the bound is not smaller than the number of variables in the CQ.

Let $\mathcal{I}_1, \mathcal{I}_2$ be interpretations, $d \in \Delta^{\mathcal{I}_1}$, and $n \geq 0$. We say that there is an *n -bounded Σ -homomorphism* from \mathcal{I}_1 to \mathcal{I}_2 , written $\mathcal{I}_1 \rightarrow_{\Sigma}^n \mathcal{I}_2$, if for any subinterpretation \mathcal{I}'_1 of \mathcal{I}_1 with $|\Delta^{\mathcal{I}'_1}| \leq n$, we have $\mathcal{I}'_1 \rightarrow_{\Sigma} \mathcal{I}_2$. Moreover, we write $\mathcal{I}_1 \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_2$ if $\mathcal{I}_1 \rightarrow_{\Sigma}^n \mathcal{I}_2$ for any n . The following characterization follows from the definition of CQ entailment, Lemma 4.3.2, and the connection between CQs and suitably bounded homomorphisms.

Lemma 4.3.3. *Let \mathcal{T}_1 and \mathcal{T}_2 be Horn-ALCHIF TBoxes with $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$. Then $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ iff for all tree-shaped Γ -ABoxes \mathcal{A} consistent with \mathcal{T}_1 and \mathcal{T}_2 , $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.*

Proof. We prove both implications via contraposition.

“ \Leftarrow ”. Assume $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ and consider a witness $(\mathcal{A}, q, \mathbf{a})$. By Lemma 4.3.2, we can assume that \mathcal{A} is tree-shaped. From Lemma 4.2.2 (4) we get $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q(\mathbf{a})$ and $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q(\mathbf{a})$. If we take the finite subinterpretation \mathcal{I} of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ given by a match of q in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, then we must have $\mathcal{I} \not\rightarrow_{\Sigma} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ because of $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \not\models q(\mathbf{a})$. Hence $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

“ \Rightarrow ”. Assume $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$, i.e., there is a finite subinterpretation \mathcal{I} of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ with $\mathcal{I} \not\rightarrow_{\Sigma} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$. Let \mathbf{a} be the ABox individuals in \mathcal{I} and let $q_{\mathcal{I}}$ be \mathcal{I} viewed as a CQ whose variables correspond to the domain elements of \mathcal{I} and the ABox individuals are represented by answer variables. Then it can be verified that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q_{\mathcal{I}}(\mathbf{a})$ and $\mathcal{I}_1 \not\models q_{\mathcal{I}}(\mathbf{a})$. \square

Ideally, we would like to use Lemma 4.3.3 as a basis for a decision procedure based on tree automata. To this end, it is useful that the ABox \mathcal{A} and models $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ and $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ in the lemma are tree-shaped. What is problematic is that Lemma 4.3.3 speaks about bounded homomorphisms, for *any* bound (corresponding to the bounded size of CQs), since it does not seem possible to verify such a condition using automata. We would thus like to replace bounded homomorphisms with unbounded ones, which does not compromise the characterization in the case of Horn-DLs without inverse roles [LW10, BLR⁺16]. However, this is not true already for \mathcal{ELI} TBoxes [BKL⁺16]:

Example 4.3.1. Let $\mathcal{T}_1 = \{A \sqsubseteq \exists s.B, B \sqsubseteq \exists r^-.B\}$, $\mathcal{T}_2 = \{A \sqsubseteq \exists s.B, B \sqsubseteq \exists r.B\}$, $\Gamma = \{A\}$, and $\Sigma = \{r\}$. The universal models are shown below.

$$\begin{array}{c} \mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \quad A \xrightarrow{s} a \bullet \xrightarrow{r} \bullet \xrightarrow{r} \bullet \xrightarrow{r} \bullet \xrightarrow{r} \bullet \dots \\ \mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \quad A \xrightarrow{s} a \bullet \xleftarrow{r} \bullet \xleftarrow{r} \bullet \xleftarrow{r} \bullet \xleftarrow{r} \bullet \dots \end{array}$$

Then both $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ and $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ contain an infinite r -path; the r -path in $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ has a final element while the one in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ does not. Hence $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$, but $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ (see Theorem 4.3.4 below).

We now show that it is possible to refine Lemma 4.3.3 so that it makes a much more careful statement in which bounded homomorphisms are *partly* replaced by unbounded ones. It is then possible to check the unbounded homomorphism part of the characterization using tree automata as desired, and to deal with bounded homomorphisms using a mosaic technique that “precompiles” relevant information about unbounded homomorphisms to be used in the automaton construction.

We start with introducing relevant notation. For a signature Σ , we use $\mathcal{I}|_{\Sigma}^{\text{con}}$ to denote the restriction of the interpretation \mathcal{I} to those elements that can be reached from an ABox individual by traveling along Σ -roles (forwards or backwards). *Tree-shaped interpretations* are defined analogously to tree-shaped CQs (thus multi-edges are allowed). For a TBox \mathcal{T} , an ABox \mathcal{A} , and $a \in \text{ind}(\mathcal{A})$, we use $\mathcal{I}_{\mathcal{T}, \mathcal{A}}|_a$ to denote the subtree interpretation in the universal model $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ rooted at a . A Σ -subtree in $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ is a maximal tree-shaped, Σ -connected sub-interpretation \mathcal{I} of $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ that does not comprise any ABox individuals. The *root* of \mathcal{I} is the (unique) element of $\Delta^{\mathcal{I}}$ that can be reached from an ABox individual on a shortest path among all element of $\Delta^{\mathcal{I}}$. The refined characterization uses simulations instead of homomorphisms for the stCQ case because they are insensitive to multi-edges. Given a signature Σ and two interpretations \mathcal{I}, \mathcal{J} , a Σ -simulation of \mathcal{I} in \mathcal{J} is a relation $\sigma \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ such that: (1) $(a, a) \in \sigma$ for all

$a \in \mathbf{N}_I$, (2) if $d \in A^{\mathcal{I}}$ with $A \in \Sigma$ and $(d, e) \in \sigma$, then $e \in A^{\mathcal{J}}$, and (3) if $(d, d') \in r^{\mathcal{I}}$ with r a Σ -role and $(d, e) \in \sigma$, then there is some e' with $(e, e') \in r^{\mathcal{J}}$ and $(d', e') \in \sigma$. We write $\mathcal{I} \preceq_{\Sigma} \mathcal{J}$ if there is a Σ -simulation of \mathcal{I} in \mathcal{J} .

Theorem 4.3.4. *Let \mathcal{T}_1 and \mathcal{T}_2 be Horn-ALCHIF TBoxes with $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{RI}} \mathcal{T}_2$. Then $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ iff for all tree-shaped Γ -ABoxes \mathcal{A} consistent with \mathcal{T}_1 and \mathcal{T}_2 , and for all tree-shaped, finitely branching models \mathcal{I}_1 of \mathcal{A} and \mathcal{T}_1 , the following hold:*

- (1) $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_{\Sigma}^{\text{con}} \rightarrow_{\Sigma} \mathcal{I}_1$;
- (2) for all Σ -subtrees \mathcal{I} in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, one of the following holds:
 - (a) $\mathcal{I} \rightarrow_{\Sigma} \mathcal{I}_1$;
 - (b) $\mathcal{I} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \text{tp}_{\mathcal{I}_1}(a)}$ for some $a \in \text{ind}(\mathcal{A})$.

Furthermore, $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$ iff $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_{\Sigma}^{\text{con}} \preceq_{\Sigma} \mathcal{I}_1$ for all \mathcal{A} and \mathcal{I}_1 as above iff $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_{\Sigma}^{\text{con}} \preceq_{\Sigma} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

To prove the second part of Theorem 4.3.4 (the stCQ case), we need a bounded variant of simulations, analogously to bounded homomorphisms. We write $\mathcal{I}_1 \preceq_{\Sigma}^n \mathcal{I}_2$ if for any subinterpretation \mathcal{I}'_1 of \mathcal{I}_1 with $|\Delta^{\mathcal{I}'_1}| \leq n$, we have $\mathcal{I}'_1 \preceq_{\Sigma} \mathcal{I}_2$. Moreover, we write $\mathcal{I}_1 \preceq_{\Sigma}^{\text{fin}} \mathcal{I}_2$ if $\mathcal{I}_1 \preceq_{\Sigma}^n \mathcal{I}_2$ for any n .

We begin with two useful insights about bounded homomorphisms (and simulations) and their connection to unbounded ones. We use $\mathcal{I}_1|_n^d$ to denote the restriction of \mathcal{I}_1 to elements that can be reached by starting at d and traveling along at most n role edges (forwards or backwards).

The first insight is straightforward.

Fact 1. Let Σ be a signature and $\mathcal{I}_1, \mathcal{I}_2$ be interpretations such that \mathcal{I}_1 is finitely branching.

- (1) The following are equivalent.
 - (a) $\mathcal{I}_1 \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_2$
 - (b) For every $d \in \Delta^{\mathcal{I}_1}$ and every $i \geq 0$: $\mathcal{I}_1|_i^d \rightarrow_{\Sigma} \mathcal{I}_2$
- (2) The following are equivalent.
 - (a) $\mathcal{I}_1 \preceq_{\Sigma}^{\text{fin}} \mathcal{I}_2$
 - (b) For every $d \in \Delta^{\mathcal{I}_1}$ and every $i \geq 0$: $\mathcal{I}_1|_i^d \preceq_{\Sigma} \mathcal{I}_2$

We will thus use Conditions (1b) and (2b) as alternative characterizations of bounded homomorphisms and simulations.

The second insight shows that, under additional conditions, we can extract an unbounded homomorphism from a suitable family of bounded ones.

Lemma 4.3.5. *Let $\mathcal{I}_1, \mathcal{I}_2$ be finitely branching interpretations and let \mathcal{I}_1 be Σ -connected.*

- (1) *If there are $d_0 \in \Delta^{\mathcal{I}_1}$ and $e_0 \in \Delta^{\mathcal{I}_2}$ such that for each $i \geq 0$ there is a Σ -homomorphism h_i from $\mathcal{I}_1|_i^{d_0}$ to \mathcal{I}_2 with $h_i(d_0) = e_0$, then $\mathcal{I}_1 \rightarrow_{\Sigma} \mathcal{I}_2$.*
- (2) *If there are $d_0 \in \Delta^{\mathcal{I}_1}$ and $e_0 \in \Delta^{\mathcal{I}_2}$ such that for each $i \geq 0$ there is a Σ -simulation ρ_i of $\mathcal{I}_1|_i^{d_0}$ in \mathcal{I}_2 with $(d_0, e_0) \in \rho_i$, then $\mathcal{I}_1 \preceq_{\Sigma} \mathcal{I}_2$.*

Proof. We only show (1); Part (2) is analogous. We are going to construct a Σ -homomorphism h from \mathcal{I}_1 to \mathcal{I}_2 step by step, obtaining the desired homomorphism in the limit. We will take care that, at all times, the domain of h is finite and

- (*) there is a sequence h_0, h_1, \dots with h_i a Σ -homomorphism from $\mathcal{I}_1|_i^{d_0}$ to \mathcal{I}_2 such that whenever $h(d)$ is already defined, then $h_i(d) = h(d)$ for all $i \geq 0$.

Start with setting $h(d_0) = e_0$. The original sequence h_0, h_1 from the lemma witnesses (*). Now consider the set Λ that consists of all elements $d \in \Delta^{\mathcal{I}_1}$ such that $h(d)$ is undefined and there is an $e \in \Delta^{\mathcal{I}_1}$ with $h(e)$ defined and such that d is reachable from e along a Σ -role edge. Since the domain of h is finite and \mathcal{I}_1 is finitely branching, Λ is finite. By (*), since every $d \in \Lambda$ is reachable in one step from an element e such that $h(e)$ is defined, and since \mathcal{I}_2 is finitely branching, for each $d \in \Lambda$ there are only finitely many e' such that $h_i(d) = e'$ for some i . Thus there must be a function $\delta : \Lambda \rightarrow \Delta^{\mathcal{I}_2}$ such that, for infinitely many i , we have $h_i(d) = \delta(d)$ for all $d \in \Lambda$. Extend h accordingly, that is, set $h(d) = \delta(d)$ for all $d \in \Lambda$. Clearly, the sequence h_0, h_1, \dots from (*) before the extension is no longer sufficient to witness (*) after the extension. We fix this by skipping homomorphisms that do not respect δ , that is, define a new sequence h'_0, h'_1, \dots by using as h'_i the restriction of h_j to the domain of $\mathcal{I}_1|_i^{d_0}$ where $j \geq i$ is smallest such that $h_j(d) = \delta(d)$ for all $d \in \Lambda$. This finishes the construction. Note that we will automatically have $h(a) = a$ for all individual names a (as required), no matter whether d_0 is an individual name or not. \square

We are now ready to prove Theorem 4.3.4.

Proof. Unrestricted CQs, “if”. We show the contrapositive. Thus first assume that $\mathcal{T}_1 \not\models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$. By Lemma 4.3.2, there is a tree-shaped Γ -ABox \mathcal{A} consistent with both \mathcal{T}_i , and a tree-shaped Σ -CQ q such that either

- (1') q has a single answer variable and there is an element $a \in \text{ind}(\mathcal{A})$ such that $\mathcal{T}_2, \mathcal{A} \models q(a)$ but $\mathcal{T}_1, \mathcal{A} \not\models q(a)$ or
- (2') q is Boolean and $\mathcal{T}_2, \mathcal{A} \models q$ but $\mathcal{T}_1, \mathcal{A} \not\models q$.

In case (1') holds, q is connected. Let h be a match of q in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$; in particular $h(x) = a$. Since q contains an answer variable, we must have $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_{\Sigma}^{\text{con}} \not\rightarrow_{\Sigma} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ as otherwise the composition of h and the witnessing homomorphism shows $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \models q(a)$, which is not the case. Thus Condition (1) is violated for $\mathcal{I}_1 = \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

In case (2') holds, consider again a match h of q in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$. Let $\mathcal{I}'_{\mathcal{T}_2, \mathcal{A}}$ be the restriction of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ to the elements in the range of h . Clearly, we have $\mathcal{I}'_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$. Consequently, $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma}^n \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ where n is the number of variables in q , implying that Conditions (2a) and (2b) are both false.

Unrestricted CQs, “only if”. Assume that $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ and let \mathcal{A} be a tree-shaped Γ -ABox consistent with both \mathcal{T}_i . We first show the following:

Claim. For all models \mathcal{I}_1 of $(\mathcal{T}_1, \mathcal{A})$, we have $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_1$.

Proof of claim: Assume to the contrary that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_1$. Then $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \not\rightarrow_{\Sigma}^n \mathcal{I}_1$ for some n , that is, there is a subinterpretation \mathcal{I} of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ with $|\Delta^{\mathcal{I}}| \leq n$ such that $\mathcal{I} \not\rightarrow_{\Sigma} \mathcal{I}_1$. Let \mathbf{a} be the ABox individuals in \mathcal{I} and let $q_{\mathcal{I}}$ be \mathcal{I} viewed as a CQ whose variables correspond to the domain elements of \mathcal{I} and the ABox individuals are represented by answer variables. Then it can be verified that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q_{\mathcal{I}}(\mathbf{a})$ and $\mathcal{I}_1 \not\models q_{\mathcal{I}}(\mathbf{a})$.

Condition 1 is a consequence of Lemma 4.3.5: Fix a tree-shaped, finitely branching model $\mathcal{I}_1 \models (\mathcal{T}_1, \mathcal{A})$ and let $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}}$ be the disjoint union of the connected interpretations $\mathcal{I}_1, \dots, \mathcal{I}_k$. In each \mathcal{I}_i , we find at least one individual a_i from $\text{ind}(\mathcal{A})$. Let $\ell \in \{1, \dots, k\}$. By the claim above and Fact 1, we find a sequence h_0, h_1, \dots such that h_i is a Σ -homomorphism from $\mathcal{I}_\ell|_i^{a_\ell}$ to \mathcal{I}_1 . Note that we must have $h_i(a_\ell) = a_\ell$ for all i . Thus, Lemma 4.3.5 yields $\mathcal{I}_\ell \rightarrow_\Sigma \mathcal{I}_1$ and, in summary, $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}} \rightarrow_\Sigma \mathcal{I}_1$.

Now for Condition 2. Let \mathcal{I} be a Σ -subtree in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ with root d_0 . By the claim above and Fact 1, there is a sequence h_0, h_1, \dots such that h_i is a Σ -homomorphism from $\mathcal{I}|_i^{d_0}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

First assume that there is an $e_0 \in \Delta^{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}}$ such that $h_i(d_0) = e_0$ for infinitely many i . Construct a new sequence h'_0, h'_1, \dots with h'_i a Σ -homomorphism from $\mathcal{I}|_i^{d_0}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ by skipping homomorphisms that do not map d_0 to e_0 , that is, h'_i is the restriction of h_j to the domain of $\mathcal{I}|_i^{d_0}$ where $j \geq i$ is smallest such that $h_j(d_0) = e_0$. Clearly, $h'_i(d_0) = e_0$ for all i . Thus, Lemma 4.3.5 yields $\mathcal{I} \rightarrow_\Sigma \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ and thus, by Lemma 4.2.2 (2) $\mathcal{I} \rightarrow_\Sigma \mathcal{I}_1$ for every tree-shaped, finitely branching model $\mathcal{I}_1 \models (\mathcal{T}_1, \mathcal{A})$.

It remains to deal with the case that there is no $e_0 \in \Delta^{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}}$ such that $h_i(d_0) = e_0$ for infinitely many i . We can assume that there is an $a_0 \in \text{ind}(\mathcal{A})$ such that $h_i(d_0) \in \Delta^{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}|_{a_0}}$ for all i ; in fact, there must be an a_0 such that $h_i(d_0) \in \Delta^{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}|_{a_0}}$ for infinitely many i and we can again skip homomorphisms to achieve this for all i . It is important to note that the remaining homomorphisms do not necessarily map all ancestors of d_0 in \mathcal{I} to elements in $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}|_{a_0}$ due to the presence of inverse roles. Now, since $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ is finitely branching, for all $i, n \geq 0$ we must find a $j \geq i$ such that $h_j(d_0)$ is a domain element whose distance from a_0 exceeds n (otherwise the previous case would apply). We can use this fact to construct a sequence h'_0, h'_1, \dots with h'_i a Σ -homomorphism from $\mathcal{I}|_i^{d_0}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}|_{a_0}$. It is easy to verify that this implies $\mathcal{I} \rightarrow_\Sigma^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}|_{a_0}$; in fact, h'_0, h'_1, \dots can again be found by skipping homomorphisms.

If we now fix an arbitrary (tree-shaped, finitely branching) model $\mathcal{I}_1 \models (\mathcal{T}_1, \mathcal{A})$, by Lemma 4.2.2 (2) and (3) we have $\text{tp}_{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}}(a_0) \subseteq \text{tp}_{\mathcal{I}_1}(a_0)$ and thus $\mathcal{I}_{\mathcal{T}_1, \text{tp}_{\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}}(a_0)} \rightarrow_\Sigma \mathcal{I}_{\mathcal{T}_1, \text{tp}_{\mathcal{I}_1}(a_0)}$. Hence $\mathcal{I} \rightarrow_\Sigma^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \text{tp}_{\mathcal{I}_1}(a_0)}$ as required.

stCQs. We need to show that the following three conditions are equivalent.

- (i) $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$
- (ii) $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}} \preceq_\Sigma \mathcal{I}_1$ for all tree-shaped Γ -ABoxes \mathcal{A} consistent with \mathcal{T}_1 and \mathcal{T}_2 , and for all tree-shaped, finitely branching models \mathcal{I}_1 of $(\mathcal{T}_1, \mathcal{A})$.
- (iii) $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}} \preceq_\Sigma \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ for all \mathcal{A} as above.

(ii) \Leftrightarrow (iii). The “only if” direction follows from Lemma 4.2.2 (1); the “if” direction follows from $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}} \preceq_\Sigma \mathcal{I}_1$, which is a direct consequence of Lemma 4.2.2 (2).

(ii) \Rightarrow (i). This implication is analogous to the “if” direction of the case for unrestricted CQs above, except that the witness stCQ is rooted and connected, which rules out Case (2') and thus Condition (2).

(i) \Rightarrow (ii). Assume that $\mathcal{T}_1 \models_{\Gamma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$ and let \mathcal{A} be a tree-shaped Σ -ABox consistent with both \mathcal{T}_i . We first show the following:

Claim. For all models \mathcal{I}_1 of $(\mathcal{T}_1, \mathcal{A})$: $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}} \preceq_\Sigma^{\text{fin}} \mathcal{I}_1$.

Proof of claim: Assume to the contrary that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}} \not\preceq_\Sigma^{\text{fin}} \mathcal{I}_1$. Then $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}^{\text{con}} \not\preceq_\Sigma^n \mathcal{I}_1$ for some n , that is, there is a subinterpretation \mathcal{I} of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ with $|\Delta^{\mathcal{I}}| \leq n$ such that $\mathcal{I} \not\preceq_\Sigma \mathcal{I}_1$. We can assume w.l.o.g. that \mathcal{I} is connected and contains at least one ABox individual

(otherwise we just extend \mathcal{I} and increase n accordingly). Let \mathbf{a} be the ABox individuals in \mathcal{I} and let $q_{\mathcal{I}}$ be \mathcal{I} viewed as a tree-shaped CQ whose variables correspond to the domain elements of \mathcal{I} and the ABox individuals are represented by answer variables. Clearly $\mathcal{I} \models q(\mathbf{a})$ and thus $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q(\mathbf{a})$; let π be a match of q in \mathcal{I} . To transform q into an stCQ, perform the following operations.

- Remove all binary atoms involving only answer variables (see Condition (b) in the proof of Lemma 4.3.2).
- Restrict the resulting CQ to one connected component, with exactly one answer variable x (see Condition (d) in the proof of Lemma 4.3.2); then x is the root of the tree q . Let $a = \pi(x)$.
- “Split” multi-edges along the tree structure of q : if there are n binary atoms involving variables z_1, z_2 of q with z_2 being a child of z_1 in the tree q , introduce n copies of z_2 and its subtree, and redirect each of the n original atoms to its corresponding copy. Apply this step exhaustively.

The result of this transformation is an stCQ q' , which still satisfies $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}} \models q'(a)$. On the other hand, $\mathcal{I}_1 \not\models q'(a)$ because, otherwise, a match π' of $q'(x)$ in \mathcal{I}_1 would give rise to a simulation of \mathcal{I} in \mathcal{I}_1 .

Having established the claim, we proceed as follows: Let a be an ABox individual in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_{\Sigma}^{\text{con}}$. By the claim and Fact 1, there is a sequence h_0, h_1, \dots such that h_i is a Σ -homomorphism from $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_i^a$ to \mathcal{I}_1 . Obviously $h_i(a) = a$ for all i . From Lemma 4.3.5 we obtain $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}|_{\Sigma}^{\text{con}} \preceq_{\Sigma} \mathcal{I}_1$ as desired. \square

We can now use the characterization in Theorem 4.3.4 to provide a decision procedure for CQ entailment.

4.4 Decidability and Complexity

We prove that, in Horn- \mathcal{ALCHIF} , CQ entailment can be decided in 2EXPTIME. By existing lower bounds, the former is thus 2EXPTIME-complete in all fragments of Horn- \mathcal{ALCHIF} that contain \mathcal{ELI} or Horn- \mathcal{ALC} . Moreover, stCQ entailment in Horn- \mathcal{ALCHIF} can also be decided in 2EXPTIME.

To obtain the upper bounds, we use a combination of tree automata and mosaics to implement the characterization in Theorem 4.3.4.

4.4.1 Mosaic Technique

We start with a mosaic-based decision procedure for Condition (2b). Note that a Σ -subtree \mathcal{I} in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ can be uniquely identified by the type t_2 of its root. It therefore suffices to show the following.

Theorem 4.4.1. *Given two Horn- \mathcal{ALCHIF} TBoxes \mathcal{T}_1 and \mathcal{T}_2 and types t_i for \mathcal{T}_i , $i \in \{1, 2\}$, it can be decided in time $2^{2^{p(|\mathcal{T}_2| \log |\mathcal{T}_1|)}}$ whether $\mathcal{I}_{\mathcal{T}_2, t_2}|_{\Sigma}^{\text{con}} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, t_1}$, p a polynomial.*

Although we cannot get rid of bounded homomorphisms in Theorem 4.3.4, a central idea for applying a mosaic approach to prove Theorem 4.4.1 is to first replace bounded homomorphisms with unbounded ones. To make this possible, we also replace $\mathcal{I}_{\mathcal{T}_1, t_1}$ with a suitable class of interpretations used as targets for the unbounded homomorphisms.

To illustrate, consider Example 4.3.1 and let $t_1 = t_2 = \{B\}$. The difference between $\mathcal{I}_{\mathcal{T}_2, t_2} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, t_1}$ and $\mathcal{I}_{\mathcal{T}_2, t_2} \rightarrow_{\Sigma} \mathcal{I}_{\mathcal{T}_1, t_1}$ is that unbounded homomorphisms fail once they “reach the root” of $\mathcal{I}_{\mathcal{T}_1, t_1}$ while bounded homomorphisms can, depending on the bound, map the root of $\mathcal{I}_{\mathcal{T}_2, t_2}$ deeper and deeper into $\mathcal{I}_{\mathcal{T}_1, t_1}$, thus never reaching its root. The latter is possible because $\mathcal{I}_{\mathcal{T}_1, t_1}$ is regular in the sense that any two elements which have the same type root isomorphic subtrees. This is of course not only true in this example, but by construction in any universal model. To transition back from bounded to unbounded homomorphisms, we replace $\mathcal{I}_{\mathcal{T}_1, t_1}$ with a class of (finite and infinite) interpretations that can be seen as a “backwards regularization” of $\mathcal{I}_{\mathcal{T}_1, t_1}$. In our concrete example, we would include an interpretation where a predecessor is added to the root of $\mathcal{I}_{\mathcal{T}_1, t_1}$ because $\mathcal{I}_{\mathcal{T}_1, t_1}$ contains an element of the same type as the root that has such a predecessor, an interpretation where that predecessor has a predecessor, and so on, even ad infinitum. We will now make this precise.

An interpretation \mathcal{I} is *quasi tree-shaped* if:

1. $\Delta^{\mathcal{I}} \subseteq (\{-1\} \cup \mathbb{N})^*$;
2. $(d, e) \in r^{\mathcal{I}}$ implies that $e = d \cdot c$ or $d = e \cdot c$ for some $c \in \{-1\} \cup \mathbb{N}$.

For $d, e \in \Delta^{\mathcal{I}}$, we say that e is a *successor* of d if $e = d \cdot c$ for some $c \in \mathbb{N}$ or $d = e \cdot -1$. By this convention, quasi tree-shaped interpretations can be viewed as directed graphs. The directedness does not correspond to the distinction between roles and inverse roles; in particular, there can be several role edges in both directions between the same d and e . Quasi tree-shaped interpretations can be viewed as a finite or infinite trees that need not have a root as they can extend indefinitely not only downwards but also upwards.

Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox and let $\text{tp}(\mathcal{T})$ be the set of all types for \mathcal{T} consistent with \mathcal{T} . For every $t_0 \in \text{tp}(\mathcal{T})$, we use $\text{tp}(\mathcal{T}, t_0)$ to denote the set of all $t \in \text{tp}(\mathcal{T})$ that occur in the universal model $\mathcal{I}_{\mathcal{T}, t_0}$ of t_0 and \mathcal{T} . Furthermore, given a quasi tree-shaped interpretation \mathcal{I} and an element $d \in \Delta^{\mathcal{I}}$, the *1-neighborhood of d in \mathcal{I}* is a tuple $n_1^{\mathcal{I}}(d) = (t^-, \rho, t, S)$ such that (a) $t = \text{tp}_{\mathcal{I}}(d)$; (b) if there is a predecessor $d_0 \in \Delta^{\mathcal{I}}$ of d , then $t^- = \text{tp}_{\mathcal{I}}(d_0)$ and $\rho = \{r \mid (d_0, d) \in r^{\mathcal{I}}\}$, otherwise $\rho = t^- = \perp$; (c) S is the set of all pairs (ρ', t') such that there is a successor d' of d such that $t' = \text{tp}_{\mathcal{I}}(d')$ and $\rho' = \{r \mid (d, d') \in r^{\mathcal{I}}\}$. We write $(t_1^-, \rho_1, t_1, S_1) \sqsubseteq (t_2^-, \rho_2, t_2, S_2)$ if $t_1 = t_2$, $S_1 \subseteq S_2$ and, if $\rho_1 \neq \perp$, then $\rho_1 = \rho_2$ and $t_1^- = t_2^-$.

In the following, we define a class $\text{can}_{\omega}(\mathcal{T}, t_0)$ of quasi tree-shaped models of \mathcal{T} . To construct a model from this class, choose a type $t \in \text{tp}(\mathcal{T}, t_0)$ and define $\mathcal{I} = (\{d_0\}, \cdot^{\mathcal{I}})$ such that $\text{tp}_{\mathcal{I}}(d_0) = t$. Then extend \mathcal{I} by applying the following rule exhaustively in a fair way:

- (R) Let $d \in \Delta^{\mathcal{I}}$. Choose some $e \in \Delta^{\mathcal{I}_{\mathcal{T}, t_0}}$ such that $n_1^{\mathcal{I}}(d) \sqsubseteq n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(e)$, and add to d the predecessor and/or successors required to achieve $n_1^{\mathcal{I}}(d) = n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(e)$.

The potentially infinite class $\text{can}_{\omega}(\mathcal{T}, t_0)$ is the set of all interpretations \mathcal{I} obtained as a limit of this construction.

Lemma 4.4.2. *Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox, $t_0 \in \text{tp}(\mathcal{T})$, and \mathcal{I} a tree-shaped interpretation. Then $\mathcal{I} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}, t_0}$ iff there is a $\mathcal{J} \in \text{can}_{\omega}(\mathcal{T}, t_0)$ with $\mathcal{I} \rightarrow_{\Sigma} \mathcal{J}$.*

Proof. “ \Rightarrow ”. Let d_0 be the root of \mathcal{I} . By Fact 1, there is a sequence h_0, h_1, \dots such that h_i is a Σ -homomorphism from $\mathcal{I}|_i^{d_0}$ to $\mathcal{I}_{\mathcal{T}, t_0}$. Note that the set $\text{tp}(\mathcal{T})$ is finite, and that $\mathcal{I}_{\mathcal{T}, t_0}$ is finitely branching. By skipping homomorphisms, we can thus construct a new sequence h'_0, h'_1, \dots such that h'_i is a Σ -homomorphism from $\mathcal{I}|_i^{d_0}$ to $\mathcal{I}_{\mathcal{T}, t_0}$ and, additionally, for every $0 \leq i \leq j$ and $d \in \Delta^{\mathcal{I}|_i^{d_0}}$ the following properties hold:

- (i) $n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(h'_i(d)) = n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(h'_j(d))$, and
- (ii) If e is a successor of d in \mathcal{I} , then $h'_i(e)$ is a successor of $h'_i(d)$ in $\mathcal{I}_{\mathcal{T}, t_0}$ iff $h'_j(e)$ is a successor of $h'_j(d)$.

Guided by h'_i , we construct a sequence of interpretations $\mathcal{J}_0, \mathcal{J}_1, \dots$ and a sequence g_0, g_1, \dots with g_i a Σ -homomorphism from $\mathcal{I}|_i^{d_0}$ to \mathcal{J}_i such that for all $0 \leq i \leq j$ and d in the domain of $\mathcal{I}|_i^{d_0}$, we have $g_i(d) = g_j(d)$. Throughout the construction, we maintain the invariant

$$n_1^{\mathcal{J}_i}(g_i(d)) \sqsubseteq n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(h'_i(d)) \quad (*)$$

for all i, d such that $g_i(d)$ is defined.

We start with $\mathcal{J}_0 = (\{e_0\}, \cdot^{\mathcal{J}_0})$ such that $\text{tp}_{\mathcal{J}_0}(e_0) = \text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(h'_0(d_0))$ and $g_0(d_0) = e_0$. Clearly (*) is satisfied. Assuming that \mathcal{J}_i and g_i are already defined, we extend them to \mathcal{J}_{i+1} and g_{i+1} by doing the following for every $(d, d') \in \rho^{\mathcal{I}}$ with $d \in \Delta^{\mathcal{I}|_i^{d_0}}$ and $d' \notin \Delta^{\mathcal{I}|_i^{d_0}}$. By invariant (*) and Item (i), we have $n_1^{\mathcal{J}_i}(g_i(d)) \sqsubseteq n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(h'_j(d))$ for all $j \geq i$; thus, we can apply **(R)** to $g_i(d)$ and $h'_i(d)$. More precisely, we obtain \mathcal{J}_{i+1} by adding a predecessor and/or successors to achieve

$$n_1^{\mathcal{J}_{i+1}}(g_i(d)) = n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(h'_i(d)). \quad (**)$$

To define $g_{i+1}(d')$, we distinguish two cases according to Item (ii):

- $h'_j(d')$ is a successor of $h'_j(d)$ for all $j \geq i$. Then there is some (ρ', t') in component S of $n_1^{\mathcal{I}_{\mathcal{T}, t_0}}(h'_i(d))$ such that $(h'_i(d), h'_i(d')) \in \rho^{\mathcal{I}_{\mathcal{T}, t_0}}$ (ρ maximal) and $\text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(h'_i(d')) = t'$. By (**), that pair is also in component S of $n_1^{\mathcal{J}_{i+1}}(g_i(d))$. Take a corresponding ρ' -successor e' of e in \mathcal{J}_{i+1} and set $g_{i+1}(d') = e'$. Clearly (*) is satisfied.
- $h'_j(d)$ is a successor of $h'_j(d')$ for all $j \geq i$. Then $t^- = \text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(h'_i(d'))$ and ρ is maximal with $(h'_i(d'), h'_i(d)) \in \rho^{\mathcal{I}_{\mathcal{T}, t_0}}$. By (**), the t^- - and ρ -component in $n_1^{\mathcal{J}_{i+1}}(g_i(d))$ are identical. Take a corresponding ρ -predecessor e' of e in \mathcal{J}_{i+1} and set $g_{i+1}(d') = e'$. Clearly (*) is satisfied.

The construction of \mathcal{J} and h is finished by setting $h = \bigcup_{i \geq 0} g_i$ and $\mathcal{J}' = \bigcup_{i \geq 0} \mathcal{J}_i$, and defining \mathcal{J} as the result of exhaustive application of rule **(R)** to \mathcal{J}' .

“ \Leftarrow ”. It suffices to show $\mathcal{J} \rightarrow^{\text{fin}} \mathcal{I}_{\mathcal{T}, t_0}$.² To this end, denote with \mathcal{J}_i , $i \geq 0$, the finite submodel of \mathcal{J} obtained after i rule applications, and with d_i the root of \mathcal{J}_i . We verify the following claim, which implies $\mathcal{J} \rightarrow^{\text{fin}} \mathcal{I}_{\mathcal{T}, t_0}$.

Claim. For all $i \geq 0$, we have:

- (i) there is an $e_0 \in \Delta^{\mathcal{I}_{\mathcal{T}, t_0}}$ with $\text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(e_0) = \text{tp}_{\mathcal{J}_i}(d_i)$;

²We write $\mathcal{I} \rightarrow^{\text{fin}} \mathcal{J}$ to denote that, for every $n \geq 0$, there are n -bounded homomorphisms from \mathcal{I} to \mathcal{J} , without restricting the signature.

(ii) for all $e_0 \in \Delta^{\mathcal{I}_{\mathcal{T}, t_0}}$ with $\text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(e_0) \supseteq \text{tp}_{\mathcal{J}_i}(d_i)$, we have $(\mathcal{J}_i, d_i) \rightarrow (\mathcal{I}_{\mathcal{T}, t_0}, e_0)$.

We prove the claim by induction on i . For $i = 0$, Points (i) and (ii) are clear by definition of \mathcal{J}_0 . For the inductive step, consider \mathcal{J}_{i+1} and suppose **(R)** has been applied to some $d \in \Delta^{\mathcal{J}_i}$ and $e \in \Delta^{\mathcal{I}_{\mathcal{T}, t_0}}$.

Observe that Point (i) is trivially preserved when d is not the root of \mathcal{J}_i . In case $d = d_i$, it is preserved by the condition on the choice of e in **(R)**: e has the same type as d_i and, by construction, the predecessor e' of e (if it exists) has the same type as d_{i+1} .

For Point (ii), we distinguish two cases:

- The extension of \mathcal{J}_i to \mathcal{J}_{i+1} has not added any predecessors to d . In particular, we then have $d_{i+1} = d_i$. Let e_0 be as in (ii), i.e., $\text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(e_0) \supseteq \text{tp}_{\mathcal{J}_i}(d_{i+1}) = \text{tp}_{\mathcal{J}_i}(d_i)$. By induction hypothesis, there is a homomorphism $h : (\mathcal{J}_i, d_{i+1}) \rightarrow (\mathcal{I}_{\mathcal{T}, t_0}, e_0)$. We extend h to the domain of \mathcal{J}_{i+1} by doing the following for each newly added successor d' of d .

Let $\text{tp}_{\mathcal{J}_{i+1}}(d) = t$ and $\text{tp}_{\mathcal{J}_{i+1}}(d') = t'$ and ρ maximal with $(d, d') \in \rho^{\mathcal{J}_{i+1}}$. By the choice of e in **(R)**, e is of type t and has a ρ -successor of type t' . By construction of the universal model, there is some $r \in \rho$ with $t \rightsquigarrow_r^{\mathcal{T}} t'$ and $\rho = \{s \mid \mathcal{T} \models r \sqsubseteq s\}$. Denote with $\hat{t} = \text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(h(d))$. The definition of a homomorphism yields $t \sqsubseteq \hat{t}$. Thus, there is $\hat{t}' \supseteq t'$ such that $\hat{t} \rightsquigarrow_r^{\mathcal{T}} \hat{t}'$. By definition of the universal model, $h(d)$ has a ρ -successor of type \hat{t}' or a ρ -predecessor of type \hat{t}'' , for $\hat{t}' \supseteq \hat{t}''$. We extend h by setting $h(d')$ to that predecessor or successor, respectively.

- The extension of \mathcal{J}_i to \mathcal{J}_{i+1} has added a ρ -predecessor d' to d . Then $d = d_i$ and $d' = d_{i+1}$. Let $\text{tp}_{\mathcal{J}_{i+1}}(d) = t$ and $\text{tp}_{\mathcal{J}_{i+1}}(d') = t'$. By construction of the universal model, there is $r \in \rho$ with $t' \rightsquigarrow_r^{\mathcal{T}} t$ and $\rho = \{s \mid \mathcal{T} \models r \sqsubseteq s\}$. Let e_0 be as in (ii), that is, $\hat{t}' := \text{tp}_{\mathcal{I}_{\mathcal{T}, t_0}}(e_0) \supseteq t'$. We then have that $\hat{t}' \rightsquigarrow_r \hat{t}$ for some $\hat{t} \supseteq t$. By definition of the universal model, e_0 has a ρ -successor of type \hat{t} or a ρ -predecessor of type $\hat{t}'' \supseteq t$. Let this element be \bar{e}_0 . By induction hypothesis, there is a homomorphism $h : (\mathcal{J}_i, d) \rightarrow (\mathcal{I}_{\mathcal{T}, t_0}, \bar{e}_0)$. We extend h by first setting $h(d') = e_0$ and then extending h to all successors of d as in the previous case.

It should be clear that h , updated as above, witnesses $(\mathcal{J}_{i+1}, d_{i+1}) \rightarrow (\mathcal{I}_{\mathcal{T}, t_0}, e_0)$. \square

We can now use Lemma 4.4.2 to devise the mosaic-based procedure for deciding the existence of a bounded homomorphism. Let $\mathcal{T}_1, \mathcal{T}_2$ be as in Theorem 4.3.4. We denote with $\text{rol}(\mathcal{T}_i)$ the set of all roles r, r^- such that the (possibly inverse) role r occurs in \mathcal{T}_i . Moreover, for a set of roles ρ , denote with $\rho|_{\Sigma}$ the restriction of ρ to Σ -roles.

Fix now some $t_1 \in \text{tp}(\mathcal{T}_1)$. Intuitively, a mosaic for t_1 represents a possible 1-neighborhood of some element in $\mathcal{I}_{\mathcal{T}_1, t_1}$ together with a decoration with sets of types for \mathcal{T}_2 that can be homomorphically embedded into the neighborhood. Formally, a *mosaic for t_1* is a tuple $M = (t^-, \rho, t, S, \ell)$ such that $(t^-, \rho, t, S) = n_1^{\mathcal{I}_{\mathcal{T}_1, t_1}}(d)$ for some $d \in \Delta^{\mathcal{I}_{\mathcal{T}_1, t_1}}$ and $\ell : \{t^-, t\} \cup S \rightarrow 2^{\text{tp}(\mathcal{T}_2)}$ satisfies the following condition:

(M) For all $\hat{t} \in \ell(t)$ we have $\hat{t} \cap \Sigma \subseteq t$ and, for all $\hat{t}' \in \text{tp}(\mathcal{T}_2)$, $r \in \text{rol}(\mathcal{T}_2)$ with $\hat{t} \rightsquigarrow_r^{\mathcal{T}_2} \hat{t}'$, one of the following holds for $\sigma = \{s \in \text{rol}(\mathcal{T}_2) \mid \mathcal{T}_2 \models r \sqsubseteq s\}$:

- $\sigma|_{\Sigma} = \emptyset$;
- $t^- \neq \perp$, $\sigma|_{\Sigma} \subseteq \rho^-$, and $\hat{t}' \in \ell(t^-)$;

(c) there is $(\rho', t') \in S$ with $\widehat{t'} \in \ell(\rho', t')$ and $\sigma|_{\Sigma} \subseteq \rho'$.

To ease notation, we use t_M^- to denote t^- , ρ_M to denote ρ , and likewise for the other components of a mosaic M . Let \mathcal{M} be the set of all mosaics for t_1 and $\mathcal{M}' \subseteq \mathcal{M}$. An $M \in \mathcal{M}'$ is *good in \mathcal{M}'* if the following conditions are satisfied:

1. for each $(\rho, t) \in S_M$, there is an $N \in \mathcal{M}'$ such that $(t_M, \rho, t) = (t_N^-, \rho_N, t_N)$, $\ell_M(\rho, t) = \ell_N(t_N)$, and $\ell_M(t_M) = \ell_N(t_N^-)$.
2. if $t_M^- \neq \perp$, there is $N \in \mathcal{M}'$ with $(\rho_M, t_M) \in S_N$, $t_M^- = t_N$, $\ell_M(t_M^-) = \ell_N(t_N)$, and $\ell_M(t_M) = \ell_N(\rho_M, t_M)$.

Let $\mathcal{M}_0, \mathcal{M}_1, \dots$ be the sequence obtained by starting with $\mathcal{M}_0 = \mathcal{M}$ and defining \mathcal{M}_{i+1} to be \mathcal{M}_i when all mosaics that are not good in \mathcal{M}_i have been removed. Assume that \mathcal{M}_p is where the sequence stabilizes.

Lemma 4.4.3. *Let $t_i \in \text{tp}(\mathcal{T}_i)$ for $i \in \{1, 2\}$. Then there is a $\mathcal{J} \in \text{can}_\omega(\mathcal{T}_1, t_1)$ such that $\mathcal{I}_{\mathcal{T}_2, t_2}|_{\Sigma}^{\text{con}} \rightarrow_{\Sigma} \mathcal{J}$ iff \mathcal{M}_p contains a mosaic M with $t_2 \in \ell_M(t_M)$.*

Proof. “ \Rightarrow ”. Let h be a Σ -homomorphism from $\mathcal{I}_{\mathcal{T}_2, t_2}|_{\Sigma}^{\text{con}}$ to some $\mathcal{J} \in \text{can}_\omega(\mathcal{T}_1, t_1)$. For every $d \in \Delta^{\mathcal{J}}$, denote with $T_h(d)$ the set of all types mapped to d by h , that is,

$$T_h(d) = \{\text{tp}_{\mathcal{I}_{\mathcal{T}_2, t_2}}(e) \mid h(e) = d, e \in \Delta^{\mathcal{I}_{\mathcal{T}_2, t_2}|_{\Sigma}^{\text{con}}}\}.$$

For every element $d \in \Delta^{\mathcal{J}}$, we define a tuple $M(d) = (t^-, \rho, t, S, \ell)$ by taking:

- $(t^-, \rho, t, S) = n_1^{\mathcal{J}}(d)$;
- $\ell(t) = T_h(d)$;
- If there is a predecessor d' of d , then $\ell(t^-) = T_h(d')$; otherwise, set $\ell(t^-) = \emptyset$ (not important);
- For every successor d' of d with $\text{tp}_{\mathcal{J}}(d') = t'$ and $\rho' = \{r \mid (d, d') \in r^{\mathcal{J}}\}$ add $(\rho', t') \in S$ and set $\ell(\rho', t') = T_h(d')$;

It is easy to verify that every $M(d) = (t^-, \rho, t, S, \ell)$ obtained in this way is actually a mosaic: By definition of \mathcal{J} , we know that $(t^-, \rho, t, S) = n_1^{\mathcal{T}_1, t_1}(d')$ for some $d' \in \Delta^{\mathcal{I}_{\mathcal{T}_1, t_1}}$. Moreover, by definition of the universal model $\mathcal{I}_{\mathcal{T}_2, t_2}$ and the fact that h is a homomorphism, Condition **(M)** is satisfied.

Let $\mathcal{M}(\mathcal{J}) = \{M(d) \mid d \in \Delta^{\mathcal{J}}\}$. It follows from the construction that all mosaics in $\mathcal{M}(\mathcal{J})$ are good in $\mathcal{M}(\mathcal{J})$; hence $\mathcal{M}(\mathcal{J}) \subseteq \mathcal{M}_p$. Finally, let d_0 be the root of $\mathcal{I}_{\mathcal{T}_2, t_2}$. By definition of $M := M(h(d_0))$, we have $t_2 \in \ell_M(t_M)$.

“ \Leftarrow ”. Assume \mathcal{M}_p contains a mosaic M with $t_2 \in \ell_M(t_M)$. We define the interpretation \mathcal{J} as the limit of the following process. We maintain a partial function $q : \Delta^{\mathcal{J}} \rightarrow \mathcal{M}_p$, intuitively mapping each domain element of \mathcal{J} to the mosaic that gave rise to it. Throughout the construction, the following invariant is preserved:

$$\text{If } q(d) = (t^-, \rho, t, S, \ell), \text{ then } n_1^{\mathcal{J}}(d) = (t^-, \rho, t, S). \quad (*)$$

We start with defining \mathcal{J} as the interpretation corresponding to the 1-neighborhood represented by M , and define $q(e_0) = M$, where e_0 is the “center” of that 1-neighborhood. By definition, the invariant $(*)$ is satisfied. Then extend \mathcal{J} by applying the following step exhaustively in a fair way: Choose some $d \in \mathcal{J}$ such that $q(d)$ is undefined, and:

- If d has a predecessor d' such that $q(d') = M'$ then, due to $(*)$, there is $(\rho, t) \in S_{M'}$ such that $(d', d) \in \rho^{\mathcal{J}}$ and $\text{tp}_{\mathcal{J}}(d) = t$. Let $N \in \mathcal{M}_p$ be the mosaic that exists according to Condition 1 of being good for $(\rho, t) \in S_{M'}$. Then extend \mathcal{J} such that $n_1^{\mathcal{J}}(d) = (t_N^-, \rho_N, t_N, S_N)$ and set $q(d) = N$.
- If d has a successor d' such that $q(d') = M'$ then, due to $(*)$, we know that $t_{M'}^- = \text{tp}_{\mathcal{J}}(d) \neq \perp$. Let $N \in \mathcal{M}_p$ be the mosaic that exists according to Condition 2 of being good. Then extend \mathcal{J} such that $n_1^{\mathcal{J}}(d) = (t_N^-, \rho_N, t_N, S_N)$ and set $q(d) = N$.

It is immediate from the construction that these steps preserve $(*)$, and that always one of the cases applies. Moreover, by construction, any interpretation \mathcal{J} obtained in the limit of such a process is an element of $\text{can}_{\omega}(\mathcal{T}_1, t_1)$. It thus remains to construct a Σ -homomorphism h witnessing $\mathcal{I}_{\mathcal{T}_2, t_2} |_{\Sigma}^{\text{con}} \rightarrow_{\Sigma} \mathcal{J}$. We proceed again inductively, maintaining the invariant:

$$\text{If } h(d) \text{ is defined, then } \text{tp}_{\mathcal{I}_{\mathcal{T}_2, t_2}}(d) \in \ell_{q(h(d))}(t_{q(h(d))}). \quad (\dagger)$$

Let d_0 be the root of $\mathcal{I}_{\mathcal{T}_2, t_2}$. We start with setting $h(d_0) = e_0$, where e_0 is as above. By the assumption that $t_2 \in \ell_M(t_M)$, invariant (\dagger) is satisfied. Now, exhaustively apply the following step. Choose $d \in \Delta^{\mathcal{I}_{\mathcal{T}_2, t_2} |_{\Sigma}^{\text{con}}}$ such that $h(d)$ is not defined but $h(d') = e$ is defined for the predecessor d' of d . Let $t = \text{tp}_{\mathcal{I}_{\mathcal{T}_2, t_2}}(d)$, $t' = \text{tp}_{\mathcal{I}_{\mathcal{T}_2, t_2}}(d')$, and $M' = q(d')$. By definition of $\mathcal{I}_{\mathcal{T}_2, t_2}$, we know that $t' \rightsquigarrow_r^{\mathcal{T}_2} t$ for some $r \in \text{rol}(\mathcal{T}_2)$. Let $\sigma = \{s \mid \mathcal{T} \models r \sqsubseteq s\}$. By invariant (\dagger) , we know that $t' \in \ell_{M'}(t_{M'})$. Thus, one of (a)–(c) in Condition **(M)** applies. Since $d, d' \in \Delta^{\mathcal{I}_{\mathcal{T}_2, t_2} |_{\Sigma}^{\text{con}}}$, we know that $\sigma |_{\Sigma} \neq \emptyset$, thus only (b) or (c) are possible. In case of (b), we extend h by setting $h(d)$ to the predecessor of $h(d')$. In case of (c), we extend h by setting $h(d)$ to the according successor of $h(d')$. Note that h extended like this satisfies the homomorphism conditions and preserves (\dagger) due to the conditions in (b) and (c). \square

We are now in a position to prove Theorem 4.4.1.

Proof.[Theorem 4.4.1] By Lemma 4.4.2, we can decide $\mathcal{I}_{\mathcal{T}_2, t_2} |_{\Sigma}^{\text{con}} \rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, t_1}$ by checking whether there is a $\mathcal{J} \in \text{can}_{\omega}(\mathcal{T}_1, t_1)$ with $\mathcal{I}_{\mathcal{T}_2, t_2} |_{\Sigma}^{\text{con}} \rightarrow_{\Sigma} \mathcal{J}$. By Lemma 4.4.3, this can be done by constructing the corresponding set \mathcal{M} of mosaics for t_1 , removing all mosaics that are not good, and checking whether the remaining set \mathcal{M}_p contains a mosaic M with $t_2 \in T_M$.

The desired upper time bound is now a consequence of the following observations:

- The size of each 1-neighborhood in $\mathcal{I}_{\mathcal{T}_1, t_1}$ is bounded by $q(|\mathcal{T}_1|)$, for a polynomial q .
- The number of mosaics for t_1 is bounded by $2^{q'(|\mathcal{T}_1|)2^{|\mathcal{T}_2|}}$ for a polynomial q' : there are at most $2^{|\mathcal{T}_1|^2}$ many 1-neighborhoods in $\mathcal{I}_{\mathcal{T}_1, t_1}$, and each such neighborhood admits at most $2^{|\mathcal{T}_1|q(|\mathcal{T}_1|)2^{|\mathcal{T}_2|}}$ many decorations with sets of types.
- Given a tuple (t^-, ρ, t, S, ℓ) , one can decide in time $2^{\hat{q}(|\mathcal{T}_1|)}$, \hat{q} a polynomial, whether (t^-, ρ, t, S) is a 1-neighborhood. Moreover, we can decide in time $2^{\hat{q}'(|\mathcal{T}_1| \cdot |\mathcal{T}_2|)}$, \hat{q}' a polynomial, whether **(M)** is satisfied.
- Conditions 1 and 2 of a mosaic being good can be checked in the desired time. \square

4.4.2 Automata-Based Technique

We now develop the decision procedure for CQ and stCQ entailment in Horn- \mathcal{ALCHIF} , based on Theorems 4.3.4 and 4.4.1. Our main tool are *alternating two-way tree automata with counting* ($2ATA_c$), an extension of alternating tree automata over *unranked trees* [GW99] with the ability to count.

$2ATA_c$ and Their Emptiness Problem

A $2ATA_c$ is a tuple $\mathfrak{A} = (Q, \Theta, q_0, \delta, \Omega)$ where Q is a finite set of *states*, Θ is the *input alphabet*, $q_0 \in Q$ is the *initial state*, δ is a *transition function*, and $\Omega : Q \rightarrow \mathbb{N}$ is a *priority function*. The transition function δ maps every state q and input letter $a \in \Theta$ to a positive Boolean formula $\delta(q, a)$ over the truth constants **true** and **false** and *transition atoms* of the form q , $\langle - \rangle q$, $[-]q$, $\diamond_n q$ and $\square_n q$. Informally, a transition q expresses that a copy of \mathfrak{A} is sent to the current node in state q ; $\langle - \rangle q$ means that a copy is sent in state q to the predecessor node, which is required to exist; $[-]q$ means the same except that the predecessor node is not required to exist; $\diamond_n q$ (resp., $\square_n q$) means that a copy of q is sent to n (resp., to all but n) successors. The semantics of $2ATA_c$ is given in terms of runs. Let (T, L) be a Θ -labeled tree and $\mathfrak{A} = (Q, \Theta, q_0, \delta, \Omega)$ a $2ATA_c$. A *run of \mathfrak{A} over (T, L)* is a $T \times Q$ -labeled tree (T_r, r) such that $\varepsilon \in T_r$, $r(\varepsilon) = (\varepsilon, q_0)$, and for all $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$, there is an assignment v of truth values to the transition atoms in θ such that v satisfies θ and:

- if $v(q') = 1$, then $r(y') = (x, q')$ for some successor y' of y in T_r ;
- if $v(\langle - \rangle q') = 1$, then $x \neq \varepsilon$ and $r(y') = (x \cdot -1, q')$ for some successor y' of y in T_r ;
- if $v([-]q') = 1$, then $x = \varepsilon$ or $r(y') = (x \cdot -1, q')$ for some successor y' of y in T_r ;
- if $v(\diamond_n q') = 1$, then there are pairwise different i_1, \dots, i_n such that, for each j , there is some successor y' of y in T_r with $r(y') = (x \cdot i_j, q')$;
- if $v(\square_n q') = 1$, then for all but n successors x' of x , there is a successor y' of y in T_r with $r(y') = (x', q')$.

Let $\gamma = i_0 i_1 \dots$ be an infinite path in T_r and denote, for all $j \geq 0$, with q_j the state such that $r(i_j) = (x, q_j)$. The path γ is *accepting* if the largest number m such that $\Omega(q_j) = m$ for infinitely many j is even. A run (T_r, r) is *accepting*, if all infinite paths in T_r are accepting. \mathfrak{A} accepts a tree if \mathfrak{A} has an accepting run over it.

We use $L(\mathfrak{A})$ to denote the set of trees accepted by \mathfrak{A} . It is standard to verify closure of $2ATA_c$ under intersection. The following is obtained via reduction to standard alternating parity tree automata [Var98].

Theorem 4.4.4. *The emptiness problem for $2ATA_c$ can be solved in time exponential in the number of states.*

The proof is by reduction to the emptiness problem of standard two-way alternating tree automata on trees of some fixed outdegree [Var98]. We need to introduce strategy trees similar to [Var98, Section 4]. A *strategy tree for \mathfrak{A}* is a tree (T, τ) where τ labels every node in T with a subset $\tau(x) \subseteq 2^{Q \times \mathbb{N} \cup \{-1\} \times Q}$, that is, with a graph with nodes from Q and edges labeled with natural numbers or -1 . Intuitively, $(q, i, p) \in \tau(x)$ expresses that, if we reached node x in state q , then we should send a copy of the

automaton in state p to $x \cdot i$. For each label ζ , we define $\text{state}(\zeta) = \{q \mid (q, i, q') \in \zeta\}$, that is, the set of sources in the graph ζ . A strategy tree is *on an input tree* (T', L) if $T = T'$, $q_0 \in \text{state}(\tau(\varepsilon))$, and for every $x \in T$, the following conditions are satisfied:

- (i) if $(q, i, p) \in \tau(x)$, then $x \cdot i \in T$;
- (ii) if $(q, i, p) \in \tau(x)$, then $p \in \text{state}(\tau(x \cdot i))$;
- (iii) if $q \in \text{state}(\tau(x))$, then the truth assignment $v_{q,x}$ defined below satisfies $\delta(q, L(x))$:
 - $v_{q,x}(p) = 1$ iff $(q, 0, p) \in \tau(x)$;
 - $v_{q,x}(\langle - \rangle p) = 1$ iff $(q, -1, p) \in \tau(x)$;
 - $v_{q,x}([-]p) = 1$ iff $x = \varepsilon$ or $(q, -1, p) \in \tau(x)$;
 - $v_{q,x}(\diamond_n p) = 1$ iff $(q, i, p) \in \tau(x)$ for n pairwise distinct $i \geq 1$;
 - $v(\Box_n p) = 1$ iff for all but at most n values $i \geq 1$ with $x \cdot i \in T$, we have $(q, i, p) \in \tau(x)$.

A *path* β in a strategy tree (T, τ) is a sequence $\beta = (u_1, q_1)(u_2, q_2) \cdots$ of pairs from $T \times Q$ such that for all $i > 0$, there is some c_i such that $(q_i, c_i, q_{i+1}) \in \tau(u_i)$ and $u_{i+1} = u_i \cdot c_i$. Thus, β is obtained by moves prescribed in the strategy tree. We say that β is *accepting* if the largest number m such that $\Omega(q_i) = m$, for infinitely many i , is even. A strategy tree (T, τ) is *accepting* if all infinite paths in (T, τ) are accepting.

Lemma 4.4.5. *A $2ATA_c$ accepts an input tree iff there is an accepting strategy tree on the input tree.*

Proof. The “if”-direction is immediate: just read off an accepting run from the accepting strategy tree.

For the “only if”-direction, we observe that acceptance of an input tree can be defined in terms of a parity game between Player 1 (trying to show that the input is accepted) and Player 2 (trying to challenge that). The initial configuration is (ε, q_0) and Player 1 begins. Consider a configuration (x, q) . Player 1 chooses a satisfying truth assignment v of $\delta(q, L(x))$. Player 2 chooses an atom α with $v_{q,x}(\alpha) = 1$ and determines the next configuration as follows:

- if $\alpha = p$, then the next configuration is (x, p) ,
- if $\alpha = \langle - \rangle p$, then the next configuration is $(x \cdot -1, p)$ unless $x = \varepsilon$; in this case, Player 1 loses immediately;
- if $\alpha = [-]p$, then the next configuration is $(x \cdot -1, p)$ unless $x = \varepsilon$; in this case, Player 2 loses immediately;
- if $\alpha = \diamond_n p$, then Player 1 selects pairwise distinct i_1, \dots, i_n with $x \cdot i_j \in T$, for all j (and loses if she cannot); Player 2 then chooses some i_j and the next configuration is $(x \cdot i_j, p)$;
- if $\alpha = \Box_n p$, then Player 1 selects n values i_1, \dots, i_n ; Player 2 then chooses some $\ell \notin \{i_1, \dots, i_n\}$ such that $x \cdot \ell \in T$ (and loses if he cannot) and the next configuration is $(x \cdot \ell, p)$.

Player 1 wins an infinite play $(x_0, q_0)(x_1, q_1) \cdots$ if the largest number m such that $\Omega(q_i) = m$, for infinitely many i , is even. It is not difficult to see that Player 1 has a winning strategy on an input tree iff \mathfrak{A} accepts the input tree.

Observe now that the defined game is a parity game and thus Player 1 has a winning strategy iff she has a *memoryless* winning strategy [EJ91]. It remains to observe that a memoryless winning strategy is nothing else than an accepting strategy tree. \square

Lemma 4.4.6. *If $L(\mathfrak{A}) \neq \emptyset$, then there is some $(T, L) \in L(\mathfrak{A})$ such that T has outdegree at most $n \cdot C$, where n is the number of states in \mathfrak{A} and C is the largest number in (some transition \diamond_{mp} or \square_{mp} in) δ .*

Proof. Let (T, L) be an input tree and τ an accepting strategy tree on T , and let C be the largest number appearing in δ . We inductively construct a tree (T', L') with $T' \subseteq T$ and L' the restriction of L to T' and an accepting strategy tree τ' on (T', L') . For the induction base, we start with $T' = \{\varepsilon\}$ and τ' the empty mapping. For the inductive step, assume that $\tau'(x)$ is still undefined for some $x \in T'$, and proceed as follows:

1. For every $(q, i, p) \in \tau(x)$ with $i \in \{-1, 0\}$, add $(q, i, p) \in \tau'(x)$.
2. For every $p \in Q$, define $N_p = \{i \geq 1 \mid (q, i, p) \in \tau(x), x \cdot i \in T\}$ and let $N'_p \subseteq N_p$ be a subset of N_p with precisely $\min(C, |N_p|)$ elements. Then:
 - (a) for all $i \in N'_p$, add $x \cdot i \in T'$;
 - (b) for all $(q, i, p) \in \tau(x)$ with $i \in N'_p$, add $(q, i, p) \in \tau'(x)$;
 - (c) for all $q \in \text{state}(x)$ and $i \in N'_p$, add $(q, i, p) \in \tau'(x)$.

By Step 2 above, T' has outdegree bounded by $|Q| \cdot C$. It remains to show that τ' is an accepting strategy tree on T' . Observe first that, by construction, $q_0 \in \text{state}(\tau'(\varepsilon))$.

We verify Conditions (i)–(iii) of a strategy tree being on an input tree. Conditions 1 follows directly from the construction. For (ii), assume that $(q, i, p) \in \tau'(x)$. By construction, there is some q' with $(q', i, p) \in \tau(x)$, and, by Condition (ii) $p \in \text{state}(\tau(x \cdot i))$. Hence, there is some $(p, j, p') \in \text{state}(\tau(x \cdot i))$. By construction, there is also some $(p, j', p') \in \text{state}(\tau'(x \cdot i))$, thus $p \in \text{state}(x \cdot i)$. For Condition (iii), take any $x \in T'$ and $q \in \text{state}(\tau'(x))$. As $q \in \text{state}(\tau(x))$, we know that the truth assignment $v_{q,x}$ defined for τ in Condition (iii) satisfies $\delta(q, L(x))$. We show that for all transitions α with $v_{q,x}(\alpha) = 1$, we also have $v'_{q,x}(\alpha) = 1$, where $v'_{q,x}$ is the truth assignment defined for τ' . By Step 1 of the construction, this is true for all α of the shape $p, \langle - \rangle p$, and $[-]p$. Let now be $\alpha = \diamond_k p$, that is, there are k pairwise distinct $i \geq 1$ such that $(q, i, p) \in \tau(x)$. By the choice of C , we have $|N'_p| \geq k$. By Step 2(c), we know that there are k pairwise distinct i such that $(q, i, p) \in \tau'(x)$, hence $v'_{q,x}(\alpha) = 1$. Consider now $\alpha = \square_k p$, that is, for all but at most k values $i \geq 1$ with $x \cdot i \in T$, we have $(q, i, p) \in \tau(x)$. By Step 2(b), this remains true for τ' , hence $v'_{q,x}(\alpha) = 1$.

We finally argue that τ' is also accepting. Let $\beta = (u_1, q_1)(u_2, q_2) \cdots$ be an infinite path in (T', τ') . We construct an infinite path $\beta' = (u'_1, q_1)(u'_2, q_2)(u'_3, q_3) \cdots$ in (T, τ) as follows:

$$- u'_1 = u_1;$$

- Let $u_{i+1} = u_i \cdot \ell$ for some ℓ with $(q_i, \ell, q_{i+1}) \in \tau'(x)$. If $\ell \in \{0, 1\}$, we have $(q_i, \ell, q_{i+1}) \in \tau(x)$, by Step 1. We set $u'_{i+1} = u'_i \cdot \ell$. If $\ell \geq 0$ then, by Step 2(c), there is some ℓ' with $(q_i, \ell', q_{i+1}) \in \tau(x)$ and $x \cdot \ell' \in T'$. Set $u'_{i+1} = u'_i \cdot \ell'$.

Since every infinite path in (T, τ) is accepting, so is β' , and thus β . □

We are now ready to reduce the emptiness problem of $2ATA_c$ to the emptiness of alternating automata running on trees of fixed outdegree, which can be solved in time exponential in the number of states [Var98].

Theorem 4.4.4 *The emptiness problem for $2ATA_c$ can be solved in time exponential in the number of states.*

Proof. Let $\mathfrak{A} = (Q, \Theta, q_0, \delta, \Omega)$ be an $2ATA_c$ with n states and C the largest number in δ . We translate \mathfrak{A} to a $2ATA^k$ $\mathfrak{A}' = (Q', \Theta', q'_0, \delta', \Omega)$ with $k = n \cdot C$, the bound from Lemma 4.4.6. Set $Q' = Q \cup \{q'_0, q_1, q_r, q_\perp\}$ and $\Theta' = (\Theta \cup \{d_\perp\}) \times \{0, 1\}$. The extended alphabet and the extra states are used to simulate transitions of the form $[-]p$ and to allow for input trees of outdegree less than k .

We obtain δ' from δ by replacing q with $(0, 1)$, $\langle - \rangle q$ with $(-1, q)$ and $[-]q$ with $(0, q_r) \vee (-1, q)$. Moreover, we replace

- $\diamond_n q$ with $\bigvee_{X \in \binom{\{1, \dots, N\}}{n}} \bigwedge_{i \in X} (i, q)$, and
- $\square_n q$ with $\bigvee_{X \in \binom{\{1, \dots, N\}}{n}} \bigwedge_{i \in \{1, \dots, N\} \setminus X} (i, q)$,

where, as usual, $\binom{M}{m}$ denotes the set of all m -elementary subsets of a set M . To deal with the case of smaller outdegree, we use the fresh symbol d_\perp as follows:

- For all $q \in Q'$: $\delta(q, (d_\perp, b)) = \begin{cases} \text{true} & \text{if } b = 0 \\ \text{false} & \text{if } b = 1 \end{cases}$

To enforce the intended labeling in the second component and the correct behaviour for q_r , we set:

$$\begin{aligned} \delta'(q'_0, (\theta, b)) &= \begin{cases} \text{false} & \text{if } b = 0 \\ q_0 \wedge \bigwedge_{i=1}^k (i, q_1) & \text{otherwise} \end{cases} \\ \delta'(q_1, (\theta, b)) &= \begin{cases} \bigwedge_{i=1}^k (i, q_1) & \text{if } b = 0 \\ \text{false} & \text{otherwise} \end{cases} \\ \delta'(q_r, (\theta, b)) &= \begin{cases} \text{true} & \text{if } b = 1 \\ \text{false} & \text{otherwise} \end{cases} \end{aligned}$$

Using Lemma 4.4.6, it is easy to verify that $L(\mathfrak{A})$ is empty iff $L(\mathfrak{A}')$ is empty. Moreover, since emptiness of $2ATA^k$ s can be checked in exponential time in the number of states, this finishes the proof of Theorem 4.4.4. □

Upper Bound

Let $\mathcal{T}_1, \mathcal{T}_2$ be Horn- \mathcal{ALCHIF} TBoxes and Γ, Σ signatures. We aim to show that one can construct a $2ATA_c$ \mathfrak{A} such that $L(\mathfrak{A}) = \emptyset$ iff $\mathcal{T}_1 \not\equiv_{\Gamma, \Sigma}^{CQ} \mathcal{T}_2$. In fact, \mathfrak{A} is the intersection of four $2ATA_c$ $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4$. They run over Θ -labeled trees with $\Theta = 2^{\Theta_0} \times 2^{\Theta_1} \times 2^{\Theta_2}$, where $\Theta_0 = \Gamma \cup \{r^- \mid r \in \Gamma\}$ and $\Theta_i = \text{sig}(\mathcal{T}_i) \cup \{r^- \mid r \in \text{sig}(\mathcal{T}_i)\}$ for $i = 1, 2$. For a Θ -labeled tree (T, L) , we use $L_i, i \in \{0, 1, 2\}$ to refer to the i -th component of L ,

that is, $L(n) = (L_0(n), L_1(n), L_2(n))$, for all $n \in T$. The component L_0 represents a (possibly infinite) ABox $\mathcal{A} = \{A(n) \mid A \in L_0(n)\} \cup \{r(n \cdot -1, n) \mid n \neq \varepsilon, r \in L_0(n)\}$, where $r^-(a, b)$ is identified with $r(b, a)$. The 2ATA_c \mathfrak{A}_1 accepts a Θ -labeled tree (T, L) iff \mathcal{A} is finite, tree-shaped (and thus connected) and includes the root of T , and it is straightforward to construct.

Components L_1, L_2 give rise to interpretations $\mathcal{I}_1 = (T, \cdot^{\mathcal{I}_1})$ and $\mathcal{I}_2 = (\text{ind}(\mathcal{A}), \cdot^{\mathcal{I}_2})$, where for $i \in \{1, 2\}$:

$$\begin{aligned} A^{\mathcal{I}_i} &= \{n \mid A \in L_i(n)\} \\ r^{\mathcal{I}_i} &= \{(n, n \cdot -1) \mid r^- \in L_i(n)\} \cup \{(n \cdot -1, n) \mid r \in L_i(n)\} \end{aligned}$$

\mathfrak{A}_2 verifies that \mathcal{I}_1 is a model of \mathcal{A} and \mathcal{T}_1 , which is standard, too. \mathfrak{A}_3 verifies that \mathcal{A} is consistent with \mathcal{T}_2 , and \mathcal{I}_2 is $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ restricted to $\text{ind}(\mathcal{A})$. This involves computing the type of an ABox element without having access to the anonymous (that is: non-ABox) part of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, using a characterization of ABox entailments [BLW13] in terms of derivation trees. Finally, \mathfrak{A}_4 verifies that either (1) or (2) from Theorem 4.3.4 is *not* satisfied. For (1), \mathfrak{A}_4 sends a copy of itself to every tree \mathcal{I} starting at an ABox element in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$, and attempts to show that \mathcal{I} cannot be homomorphically embedded into a corresponding tree in \mathcal{I}_1 . This attempt is successful if either incompatible types are found in the root or, recursively, there is some successor of the current type in $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ that cannot be mapped to any neighbor in \mathcal{I}_1 . Since the anonymous part of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ is not explicit in the input, the current type is stored in the states, and the generating relation $t \rightsquigarrow_r^{\mathcal{T}_2} t'$ is “hard-coded” into the transition function. For Condition (2a), \mathfrak{A}_4 non-deterministically guesses a Σ -subtree \mathcal{I} and proceeds as in (1); Condition (2b) is verified based on Theorem 4.4.1 by pre-computing $\rightarrow_{\Sigma}^{\text{fin}}$. Thus the number of states of \mathfrak{A}_4 is exponential in \mathcal{T}_2 (because of the types) but only polynomial in $|\mathcal{T}_1|$. Automata $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3$ have polynomially many states.

In the special case of stCQ entailment, we simply replace \mathfrak{A}_4 with a 2ATA_c \mathfrak{A}'_4 that refutes the simulation condition of Theorem 4.3.4 analogously to how \mathfrak{A}_4 refutes Condition (1).

To obtain the desired upper complexity bounds for CQ and stCQ entailment, we observe that, in both cases, \mathfrak{A} can be constructed in time polynomial in $|\mathcal{T}_1|$ and exponential in $|\mathcal{T}_2|$, and the emptiness check adds an exponential blowup (Theorem 4.4.4).

Theorem 4.4.7. *In Horn-ALCHIF, the following problems can be decided in time $2^{2^{p(|\mathcal{T}_2| \log |\mathcal{T}_1|)}}$, p a polynomial: (Γ, Σ) -CQ entailment, (Γ, Σ) -CQ inseparability, and (Γ, Σ) -CQ conservative extensions. The same holds for (Γ, Σ) -stCQ entailment, (Γ, Σ) -stCQ inseparability, and (Γ, Σ) -stCQ conservative extensions.*

We show the following lemma which together with Theorem 4.4.4 implies Theorem 4.4.7.

Lemma 4.4.8. *There are 2ATA_c $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4, \mathfrak{A}'_4$ such that:*

- \mathfrak{A}_1 accepts (T, L) iff \mathcal{A} is finite, tree-shaped, and contains ε ;
- \mathfrak{A}_2 accepts (T, L) iff \mathcal{I}_1 is a model of \mathcal{A} and \mathcal{T}_1 ;
- \mathfrak{A}_3 accepts (T, L) iff \mathcal{A} is consistent with \mathcal{T}_2 , and \mathcal{I}_2 is $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ restricted to $\text{ind}(\mathcal{A})$;
- \mathfrak{A}_4 accepts (T, L) iff either (1) or (2) from Theorem 4.3.4 is not satisfied, when $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}}$ is replaced with \mathcal{I}_2 .

– \mathfrak{A}'_4 accepts (T, L) iff $\mathcal{I}_2|_{\Sigma}^{\text{con}} \not\leq_{\Sigma} \mathcal{I}_1$.

The number of states of \mathfrak{A}_1 and \mathfrak{A}_2 is polynomial in $|\mathcal{T}_1|$ (and independent of \mathcal{T}_2); the number of states of \mathfrak{A}_3 is polynomial in $|\mathcal{T}_2|$ (and independent of \mathcal{T}_1), and the number of states of $\mathfrak{A}_4, \mathfrak{A}'_4$ is exponential in $|\mathcal{T}_2|$ (and independent of \mathcal{T}_1). All automata can be constructed in time polynomial in $|\mathcal{T}_1|$ and double-exponential in $|\mathcal{T}_2|$.

The construction of the automaton \mathfrak{A}_1 is straightforward, so we concentrate on $\mathfrak{A}_2, \mathfrak{A}_3$, and \mathfrak{A}_4 .

In what follows, we use $\diamond q$ and $\square q$ to abbreviate $\diamond_1 q$ and $\square_0 q$, respectively. We define $\mathfrak{A}_2 = (Q_2, \Theta, q_0, \delta_2, \Omega_2)$ where

$$Q_2 = \{q_0, q_A\} \cup \{q_\alpha \mid \alpha \in \mathcal{T}_1\} \cup \{q_\rho, \bar{q}_\rho \mid \rho \in \Theta_1\} \cup \\ \{q_{r,B}, q_{r,B}^\downarrow, \bar{q}_{r,B}, \bar{q}_{r,B}^\downarrow \mid \exists r.B \in \text{cl}(\mathcal{T}_1)\},$$

and Ω_2 assigns 0 to all states. The idea of \mathfrak{A}_2 is to check that the ABox is satisfied, realized in state q_A , and that every axiom TBox axiom in \mathcal{T}_1 is satisfied everywhere, realized using states q_α below. Formally, the transition function δ_2 is given as follows, for $\sigma = (L_0, L_1, L_2)$:

$$\begin{aligned} \delta_2(q_0, \sigma) &= \square q_0 \wedge q_A \wedge \bigwedge_{\alpha \in \mathcal{T}_1} q_\alpha \\ \delta_2(q_A, \sigma) &= \bigwedge_{\rho \in L_0} q_\rho \\ \delta_2(q_{\text{func}(r)}, \sigma) &= (q_{r-} \wedge \square \bar{q}_r) \vee (\bar{q}_{r-} \wedge \square_1 \bar{q}_r) \\ \delta_2(q_{r \sqsubseteq s}, \sigma) &= \bar{q}_r \vee q_s \\ \delta_2(q_{A_1 \sqcap A_2 \sqsubseteq B}, \sigma) &= \bar{q}_{A_1} \vee \bar{q}_{A_2} \vee q_B \\ \delta_2(q_{A \sqsubseteq \perp}, \sigma) &= \bar{q}_A \\ \delta_2(q_{\top \sqsubseteq A}, \sigma) &= q_A \\ \delta_2(q_{A \sqsubseteq \exists r.B}, \sigma) &= \bar{q}_A \vee q_{r,B} \\ \delta_2(q_{\exists r.A \sqsubseteq B}, \sigma) &= \bar{q}_{r,A} \vee q_B \\ \delta_2(q_{r,B}, \sigma) &= \diamond q_{r,B}^\downarrow \vee (q_{r-} \wedge \langle - \rangle q_B) \\ \delta_2(\bar{q}_{r,B}, \sigma) &= \square \bar{q}_{r,B}^\downarrow \wedge (\bar{q}_{r-} \vee [-] \bar{q}_B) \\ \delta_2(q_{r,B}^\downarrow, \sigma) &= q_r \wedge q_B \\ \delta_2(\bar{q}_{r,B}^\downarrow, \sigma) &= \bar{q}_r \vee \bar{q}_B \end{aligned}$$

Finally, we set for all $\rho \in \Theta_1$:

$$\begin{aligned} \delta_2(q_\rho, \sigma) &= \begin{cases} \text{true} & \text{if } \rho \in L_1 \\ \text{false} & \text{if } \rho \notin L_1 \end{cases} \\ \delta_2(\bar{q}_\rho, \sigma) &= \begin{cases} \text{true} & \text{if } \rho \notin L_1 \\ \text{false} & \text{if } \rho \in L_1 \end{cases} \end{aligned}$$

Automaton \mathfrak{A}_3 relies on a syntactic characterization of ABox entailment [BLW13], which we introduce first.

Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox and \mathcal{A} a tree-shaped ABox. A *derivation tree* for an assertion $A_0(a_0)$ in \mathcal{A} w.r.t. \mathcal{T} with $A_0 \in \mathbf{N}_C$ is a finite $\text{ind}(\mathcal{A}) \times \mathbf{N}_C$ -labeled tree (T, V) that satisfies the following conditions:

1. $V(\varepsilon) = (a_0, A_0)$;
2. if $V(x) = (a, A)$ and neither $A(a) \notin \mathcal{A}$ nor $\top \sqsubseteq A \in \mathcal{T}$, then one of the following holds:
 - (i) x has successors y_1, \dots, y_k , $k \geq 1$ with $V(y_i) = (a, A_i)$ for $1 \leq i \leq k$ and $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_k \sqsubseteq A$;
 - (ii) x has a single successor y with $V(y) = (b, B)$ and there is an $\exists r.B \sqsubseteq A \in \mathcal{T}$ and an $s(a, b) \in \mathcal{A}$ such that $\mathcal{T} \models s \sqsubseteq r$;
 - (iii) x has a single successor y with $V(y) = (b, B)$ and there is a $B \sqsubseteq \exists r.A \in \mathcal{T}$ such that $r(b, a) \in \mathcal{A}$ and $\text{func}(r) \in \mathcal{T}$.

Note that the first item of Point 2 above requires $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A$ instead of $A_1 \sqcap A_2 \sqsubseteq A \in \mathcal{T}$ to ‘shortcut’ anonymous parts of the universal model. In fact, the derivation of A from $A_1 \sqcap \dots \sqcap A_n$ by \mathcal{T} can involve the introduction of anonymous elements.

The main property of derivation trees is the following.

Lemma 4.4.9. *Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox and \mathcal{A} an ABox consistent with \mathcal{T} . Then for all assertions $A(a)$ with $A \in \mathbf{N}_C$, and $a \in \text{ind}(\mathcal{A})$ we have $\mathcal{T}, \mathcal{A} \models A(a)$ iff there is a derivation tree for $A(a)$ in \mathcal{A} w.r.t. \mathcal{T} .*

Proof. The “if”-direction is immediate, so we concentrate on the “only if”-direction. We construct a sequence of interpretations $\mathcal{I}_0, \mathcal{I}_1, \dots$ by the following procedure. We start with setting:

$$\begin{aligned} \Delta^{\mathcal{I}_0} &= \text{ind}(\mathcal{A}) \\ A^{\mathcal{I}_0} &= \{a \mid A(a) \in \mathcal{A}\} \\ r^{\mathcal{I}_0} &= \{(a, b) \mid r(a, b) \in \mathcal{A}\} \end{aligned}$$

For every $i \geq 0$, we obtain \mathcal{I}_{i+1} from \mathcal{I}_i by setting $\mathcal{I}_{i+1} = \mathcal{I}_i$ and applying the following rules to all $d, e \in \Delta^{\mathcal{I}_i}$:

1. If $d \in (A_1 \sqcap A_2)^{\mathcal{I}_i}$, but $d \notin A^{\mathcal{I}_i}$ for some $A_1 \sqcap A_2 \sqsubseteq A \in \mathcal{T}$, then add $d \in A^{\mathcal{I}_{i+1}}$;
2. If $d \in (\exists r.B)^{\mathcal{I}_i}$, but $d \notin A^{\mathcal{I}_i}$ for some $\exists r.B \sqsubseteq A \in \mathcal{T}$, then add $d \in A^{\mathcal{I}_{i+1}}$;
3. If $(d, e) \in r^{\mathcal{I}_i}$ but $(d, e) \notin s^{\mathcal{I}_i}$, for some s with $\mathcal{T} \models r \sqsubseteq s$, then add $(d, e) \in s^{\mathcal{I}_{i+1}}$;
4. If $d \in A^{\mathcal{I}_i}$, but $d \notin (\exists r.B)^{\mathcal{I}_i}$ for some $A \sqsubseteq \exists r.B \in \mathcal{T}$, then:
 - (a) if there is e with $(d, e) \in r^{\mathcal{I}_i}$ and $\text{func}(r) \in \mathcal{T}$ then add $e \in B^{\mathcal{I}_{i+1}}$;
 - (b) otherwise add a fresh domain element e with $(d, e) \in r^{\mathcal{I}_{i+1}}$ and $e \in B^{\mathcal{I}_{i+1}}$.

Let \mathcal{I} be defined as $\Delta^{\mathcal{I}} = \bigcup_{i \geq 0} \Delta^{\mathcal{I}_i}$, $A^{\mathcal{I}} = \bigcup_{i \geq 0} A^{\mathcal{I}_i}$, and $r^{\mathcal{I}} = \bigcup_{i \geq 0} r^{\mathcal{I}_i}$. It is standard to verify the following:

Claim 1. $\mathcal{I} \rightarrow \mathcal{J}$ for all models \mathcal{J} of \mathcal{T} and \mathcal{A} .

By definition of \mathcal{I}_0 , we have $\mathcal{I} \models \mathcal{A}$. Moreover, we have $\mathcal{I} \models \mathcal{T}'$ where $\mathcal{T}' \sqsubseteq \mathcal{T}$ is obtained from \mathcal{T} by dropping all CIs of the form $A \sqsubseteq \perp$ and all FAs. Since \mathcal{A} is consistent with \mathcal{T} , there is a model \mathcal{J} of \mathcal{A} and \mathcal{T} ; in particular, $A^{\mathcal{J}} = \emptyset$ for all $A \sqsubseteq \perp \in \mathcal{T}$. By Claim 1, we have $\mathcal{I} \rightarrow \mathcal{J}$, and thus $A^{\mathcal{I}} = \emptyset$. For the FAs $\text{func}(s)$, observe that they are obeyed by \mathcal{A} (because of consistency with \mathcal{T}) and that they are preserved, by rule 4(a). Thus, \mathcal{I} is a model of \mathcal{T} .

Claim 2. For all $i \geq 0$, we have:

- (a) For all $a \in \text{ind}(\mathcal{A})$: if $a \in A^{\mathcal{I}_i}$, then there is a derivation tree for $A(a)$ in \mathcal{A} w.r.t. \mathcal{T} .
- (b) If e was created because of d in Rule 4(b), then we have $\mathcal{T} \models \Box\{A \mid d \in A^{\mathcal{I}_i}\} \sqsubseteq \exists r. \Box\{A \mid e \in A^{\mathcal{I}_i}\}$ for all r with $(d, e) \in r^{\mathcal{I}_i}$.

Proof of Claim 2. It is standard to show Part (b) of the Claim. We show Part (a) by induction on i . By construction of \mathcal{I}_0 , it is true for $i = 0$. Consider \mathcal{I}_{i+1} . If $a \in A^{\mathcal{I}_{i+1}}$ because of Rule 1, construct a derivation tree of type (i) from the derivation trees for $A_1(a)$ and $A_2(a)$ which exist due to the induction hypothesis. If $a \in A^{\mathcal{I}_{i+1}}$ because of Rule 2, there is some $d \in B^{\mathcal{I}_i}$ with $(a, d) \in r^{\mathcal{I}_i}$ and $\exists r. B \sqsubseteq A \in \mathcal{T}$. If $d \in \text{ind}(\mathcal{A})$, then there is some $s(a, d) \in \mathcal{A}$ with $\mathcal{T} \models s \sqsubseteq r$, by Rule 3. We can thus construct a derivation of type (ii) from the derivation tree of $B(d)$, which exists due to induction hypothesis. If $d \notin \text{ind}(\mathcal{A})$, then d was created because of a in Rule 4(b). By Part (b) of the Claim, we have $\mathcal{T} \models \Box\{A' \mid a \in A'^{\mathcal{I}_i}\} \sqsubseteq \exists r. B$. Hence, $\mathcal{T} \models \Box\{A' \mid a \in A'^{\mathcal{I}_i}\} \sqsubseteq A$, and we can construct a derivation tree of type (i) for $A(a)$. If $a \in A^{\mathcal{I}_{i+1}}$ because of Rule 4(a), there is $(d, a) \in r_i^{\mathcal{I}_i}$ and $d \in B^{\mathcal{I}_i}$, and $B \sqsubseteq \exists r. A, \text{func}(r) \in \mathcal{T}$. If $d \in \text{ind}(\mathcal{A})$, we can construct a derivation tree of type (iii) for $A(a)$ from the derivation tree of $B(d)$ which exists by induction. If $d \notin \text{ind}(\mathcal{A})$, then d was created because of a in Rule 4(b). By Part (b) of the Claim, we have $\mathcal{T} \models \Box\{A' \mid a \in A'^{\mathcal{I}_i}\} \sqsubseteq \exists r^-. B$. Hence, $\mathcal{T} \models \Box\{A' \mid a \in A'^{\mathcal{I}_i}\} \sqsubseteq A$, and construct a derivation tree of type (i) for $A(a)$ based on this. This finishes the proof of Claim 2 and the Lemma. \square

In the following Lemma, we characterize consistency of ABoxes with TBoxes.

Lemma 4.4.10. *Let \mathcal{T} be a Horn- \mathcal{ALCHIF} TBox and \mathcal{A} an ABox. Then \mathcal{A} is consistent with \mathcal{T} iff the following points are satisfied for all $a \in \text{ind}(\mathcal{A})$:*

1. the following ABox \mathcal{A}_a is consistent with \mathcal{T} :

$$\mathcal{A}_a = \{B(a) \mid B(a) \text{ has a derivation tree in } \mathcal{A} \text{ w.r.t. } \mathcal{T}\}$$

2. for all $\text{func}(s) \in \mathcal{T}$, there is at most one $b \in \text{ind}(\mathcal{A})$ with $s(a, b) \in \mathcal{A}$.

Proof. The “only if”-direction is immediate, so we concentrate on the “if”-direction. Assume that all $a \in \text{ind}(\mathcal{A})$ satisfy both items above. By the first item, there is a model \mathcal{I}_a of \mathcal{A}_a and \mathcal{T} . Since we are considering Horn- \mathcal{ALCHIF} , there is also a tree-model \mathcal{I}_a with root $d_a \in \Delta^{\mathcal{I}_a}$ satisfying, for all concept names $B \in \text{NC}$:

$$(*) \quad d_a \in B^{\mathcal{I}_a} \text{ iff } \mathcal{T}, \mathcal{A}_a \models B(a).$$

We construct an interpretation \mathcal{I} as follows. Start with \mathcal{I}_0 by taking

$$\begin{aligned} \Delta^{\mathcal{I}_0} &= \text{ind}(\mathcal{A}) \\ A^{\mathcal{I}_0} &= \{a \mid A(a) \text{ has a derivation tree in } \mathcal{A} \text{ w.r.t. } \mathcal{T}\} \\ r^{\mathcal{I}_0} &= \{(a, b) \mid s(a, b) \in \mathcal{A}, \mathcal{T} \models s \sqsubseteq r\} \end{aligned}$$

Now, obtain \mathcal{I} from \mathcal{I}_0 by performing the following operation for every $a \in \text{ind}(\mathcal{A})$ and $b \in \Delta^{\mathcal{I}_a}$ such that $(d_a, b) \in \rho^{\mathcal{I}}$ for some set of roles ρ which contains no role r such that there is a' with $r(a, a') \in \mathcal{A}$. Extend \mathcal{I} by adding the sub-interpretation of \mathcal{I}_a rooted at b as a ρ -successor of a .

Based on (*) and the assumptions, it is straightforward to show that \mathcal{I} is a model of \mathcal{A} and \mathcal{T} . \square

We are now ready to give the automaton \mathfrak{A}_3 . We take $\mathfrak{A}_3 = (Q_3, \Theta, q_0, \delta_3, \Omega_3)$ where

$$\begin{aligned} Q_3 = & \{q_0, q_{0r}\} \cup \{q_A, \bar{q}_A \mid A \in \Theta_2 \cap \mathbf{N}_C\} \cup \\ & \{q_r, \bar{q}_r, q_r^A, \bar{q}_r^A, q_r^f, q_{\neg r} \mid r \in \Theta_2 \setminus \mathbf{N}_C\} \cup \\ & \{q_{r,B}, \bar{q}_{r,B} \mid r \in \Theta_2 \cap \mathbf{N}_R, B \in \Theta_2 \cap \mathbf{N}_C\} \end{aligned}$$

and Ω_3 assigns zero to all states, except for states of the form q_A , to which it assigns 1. The automaton \mathfrak{A}_3 ensures that, for all $n \in \text{ind}(\mathcal{A})$ we have:

- (i) $A \in L_2(n)$ iff there is a derivation tree for $A(n)$ in \mathcal{A} ,
- (ii) for all $n \neq \varepsilon$, $r \in L_2(n)$ iff there is some s such that $s(n \cdot -1, n) \in \mathcal{A}$ and $\mathcal{T}_2 \models s \sqsubseteq r$.

Intuitively, these points ensure that the represented interpretation \mathcal{I}_2 is the universal model of \mathcal{T}_2 and \mathcal{A} , in case \mathcal{A} is consistent with \mathcal{T}_2 . Having (i) and (ii), we can check inconsistency of \mathcal{A} with \mathcal{T}_2 based on Lemma 4.4.10, that is, we verify the following conditions for all $n \in \text{ind}(\mathcal{A})$:

- (iii) the set $L_2(n) \cap \mathbf{N}_C$ is consistent with \mathcal{T}_2 ;
- (iv) for each s with $\text{func}(s) \in \mathcal{T}$, there are no $n_1 \neq n_2$ such that both $s(n, n_1) \in \mathcal{A}$ and $s(n, n_2) \in \mathcal{A}$.

For Point (i), we use states q_A for the “if” part, and states \bar{q}_A for the “only if” part; for Point (ii), we use states q_r and \bar{q}_r , respectively. Intuitively, a state q_A assigned to some node n is an obligation to verify the existence of a derivation tree for $A(n)$. Conversely, \bar{q}_A is the obligation that there is *no* such derivation tree. Similar obligations hold for q_r and \bar{q}_r . For Point (iii), we precompute the set of consistent types and check (iii) while visiting all $n \in \text{ind}(\mathcal{A})$. Point (iv) can be checked directly on \mathcal{A} , that is, independent from \mathcal{T}_2 . The automaton starts with the following transitions, where we assume $\sigma = (L_0, L_1, L_2)$:

- $\delta_3(q_0, \sigma) = \text{true}$ if $L_0 = \emptyset$;
- $\delta_3(q_0, \sigma) = \text{false}$ if $L_0 \neq \emptyset$ and $L_2 \cap \mathbf{N}_C$ inconsistent with \mathcal{T}_2 , c.f. Point (iii);
- if $L_0 \neq \emptyset$ and $L_2 \cap \mathbf{N}_C$ consistent with \mathcal{T}_2 , then

$$\delta_3(q_0, \sigma) = \Box q_0 \wedge \Box q_{0r} \wedge \bigwedge_{A \in L_2 \cap \mathbf{N}_C} q_A \wedge \bigwedge_{A \in (\Theta_2 \cap \mathbf{N}_C) \setminus L_2} \bar{q}_A.$$

- $\delta_3(q_{0r}, \sigma) = \text{true}$ if $L_0 = \emptyset$;
- if $L_0 \neq \emptyset$, then

$$\delta_3(q_r, \sigma) = \bigwedge_{\text{func}(r) \in \mathcal{T}_2} q_r^f \wedge \bigwedge_{r \in L_2 \cap \mathbf{N}_R} q_r \wedge \bigwedge_{r \in (\Theta_2 \cap \mathbf{N}_R) \setminus L_2} \bar{q}_r$$

$$- \delta_3(q_r^f, \sigma) = \begin{cases} \Box q_{\neg r} & \text{if } r^- \in L_0 \\ \Box 1 q_{\neg r} & \text{if } r^- \notin L_0 \end{cases}$$

$$- \delta_3(q_{\neg r}, \sigma) = \begin{cases} \text{true} & \text{if } r \notin L_0 \\ \text{false} & \text{otherwise} \end{cases}$$

Now, for states q_A , we directly implement the conditions of a derivation tree. Finiteness of the derivation is ensured by the priority of states of the form q_A . The relevant transitions are as follows:

- $\delta_3(q_A, \sigma) = \text{false}$ if $L_0 = \emptyset$;
- $\delta_3(q_A, \sigma) = \text{true}$ if $A \in L_0$;
- if $A \notin L_0$ and $L_0 \neq \emptyset$, then

$$\begin{aligned} \delta_3(q_A, \sigma) = & \bigvee_{\mathcal{T}_2 \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A} (q_{A_1} \wedge \dots \wedge q_{A_n}) \vee \\ & \bigvee_{\exists r. B \sqsubseteq A \in \mathcal{T}_2, \mathcal{T}_2 \models s \sqsubseteq r} (q_{s^-}^A \wedge \langle - \rangle q_B) \vee \diamond q_{s, B} \vee \\ & \bigvee_{B \sqsubseteq \exists r. A \in \mathcal{T}_2, \text{func}(r) \in \mathcal{T}_2} (q_s^A \wedge \langle - \rangle q_B) \vee \diamond q_{s^-, B} \end{aligned}$$

- $\delta_3(q_r^A, \sigma) = \begin{cases} \text{true} & \text{if } r \in L_0; \\ \text{false} & \text{otherwise;} \end{cases}$
- $\delta_3(q_{s, B}, \sigma) = q_s^A \wedge q_B$.

The transitions for \bar{q}_A are obtained by taking the “complement” of the ones for q_A . More precisely, we define $\delta_3(\bar{q}, \sigma) = \overline{\delta_3(q, \sigma)}$, where $\bar{\varphi}$ is obtained from φ by exchanging \wedge and \vee , \diamond and \square , $\langle - \rangle$ and $[-]$, and **true** and **false**, and replacing every state p with \bar{p} ; see the following set of transitions.

- $\delta_3(\bar{q}_A, \sigma) = \text{true}$ if $L_1 = \emptyset$;
- $\delta_3(\bar{q}_A, \sigma) = \text{false}$ if $A \in L_1$;
- if $A \notin L_1$ and $L_1 \neq \emptyset$, then

$$\begin{aligned} \delta_3(\bar{q}_A, \sigma) = & \bigwedge_{\mathcal{T}_2 \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A} (\bar{q}_{A_1} \vee \dots \vee \bar{q}_{A_n}) \wedge \\ & \bigwedge_{\exists r. B \sqsubseteq A \in \mathcal{T}_2, \mathcal{T}_2 \models s \sqsubseteq r} (\bar{q}_{s^-}^A \wedge [-] \bar{q}_B) \wedge \square \bar{q}_{s, B} \wedge \\ & \bigwedge_{B \sqsubseteq \exists r. A \in \mathcal{T}_2, \text{func}(r) \in \mathcal{T}_2} (\bar{q}_s^A \vee [-] \bar{q}_B) \wedge \square \bar{q}_{s^-, B} \end{aligned}$$

- $\delta_3(\bar{q}_r^A, \sigma) = \begin{cases} \text{false} & \text{if } r \in L_0; \\ \text{true} & \text{otherwise;} \end{cases}$
- $\delta_3(\bar{q}_{s, B}, \sigma) = \bar{q}_s^A \vee \bar{q}_B$.

Finally, states q_r and \bar{q}_r at some node n represent the obligation to verify that the role atom $r(n \cdot -1, n)$ follows, respectively does not follow, from \mathcal{T} and \mathcal{A}_2 . This is realized by the following transitions which implement Point (ii) above.

$$\delta_3(q_r, \sigma) = \bigvee_{\mathcal{T}_2 \models s \sqsubseteq r} q_s^A \quad \delta_3(\bar{q}_r, \sigma) = \bigwedge_{\mathcal{T}_2 \models s \sqsubseteq r} \bar{q}_s^A$$

For the automaton \mathfrak{A}_4 , we take $\mathfrak{A}_4 = (Q_4, \Theta, q_0, \delta_4, \Omega_4)$ where

$$Q_4 = \{q_0, q_1, q_r\} \cup \{q_t, q_t^3, q_t^{3b} \mid t \in \text{tp}(\mathcal{T}_2)\} \cup \\ \{q_{\rho,t}, q_{\rho,t}^\downarrow \mid t \in \text{tp}(\mathcal{T}_2), \rho \text{ set of } \text{sig}(\mathcal{T}_2)\text{-roles}\},$$

and Ω_4 assigns zero to all states, except for states of the form q_t , $t \in \text{tp}(\mathcal{T}_2)$, to which it assigns one. For some $n \in \text{ind}(\mathcal{A})$, denote with \mathcal{J}_n the universal model of the type $\{A(n) \mid A \in L_2(n)\}$ and \mathcal{T}_2 . The automaton ensures that indeed (1) or (2) from Theorem 4.3.4 is not satisfied, by verifying that there is some $n \in \text{ind}(\mathcal{A})$ such that one of the following conditions holds:

1. there is $r \in \Sigma$ and $n' \in \text{ind}(\mathcal{A})$ such that $(n, n') \in r^{\mathcal{I}_2}$, but $(n, n') \notin r^{\mathcal{I}_1}$;
2. $\mathcal{J}_n \not\rightarrow_{\Sigma} \mathcal{I}_1$;
3. there is a Σ -subtree \mathcal{J} of \mathcal{J}_n such that
 - (a) $\mathcal{J} \not\rightarrow_{\Sigma} \mathcal{I}_1$, and
 - (b) $\mathcal{J} \not\rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, \text{tp}_{\mathcal{I}_1}(m)}$, for all m with $L_0(m) \neq \emptyset$.

Condition 1 is straightforward (realized in state q_r). For Condition 2, we use states q_t with $t \in \text{tp}(\mathcal{T}_2)$. A state q_t assigned to a node n represents the obligation to verify that there is *no* Σ -homomorphism from the universal model of t and \mathcal{T}_2 to \mathcal{I}_1 that maps the root to n . This is the case if either the root cannot be mapped to n , or, recursively, there is some ρ -successor t' of t in the universal model such that the universal model of t' and \mathcal{T}_2 cannot be mapped to any ρ -neighbor of n . This process is finite because of priority 1 for all q_t with $t \in \text{tp}(\mathcal{T}_2)$. For Condition 3, we precompute the set $R_{\Sigma}(t)$ of all types of roots of Σ -subtrees which appear in the universal model of t and \mathcal{T}_2 , and the relation $\rightarrow_{\Sigma}^{\text{fin}}$ according to Theorem 4.4.1. Thus, the sets $R_{\Sigma}(t)$ and the test for finite homomorphisms can be used directly in the transition condition, see states q_1 and q_t^{3b} , respectively. Using states q_t^3 , the automaton ensures that a given root t of a Σ -subtree satisfies 3(a) and 3(b).

Let $t|_{\Sigma}$ and $\rho|_{\Sigma}$ denote the restriction of t and ρ , respectively, to symbols from Σ . For $\sigma = (L_1, L_2, L_3)$, we take the following transitions:

$$\begin{aligned} \delta_3(q_0, \sigma) &= \begin{cases} \diamond q_0 \vee q_1 & \text{if } L_0 \neq \emptyset \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_3(q_1, \sigma) &= q_r \vee q_t \vee \bigvee_{t' \in R_{\Sigma}(t)} q_t^3 \quad \text{for } t = L_2 \cap \mathbf{N}_{\mathbf{C}} \\ \delta_3(q_t, \sigma) &= \begin{cases} \text{true} & \text{if } t|_{\Sigma} \not\subseteq L_1 \\ \bigvee_{t' | t \rightsquigarrow_{\rho}^{\mathcal{T}_2} t'} q_{\rho, t'} & \text{otherwise} \end{cases} \\ \delta_3(q_{\rho, t}, \sigma) &= \begin{cases} \Box q_{\rho, t}^\downarrow & \text{if } \rho^-|_{\Sigma} \not\subseteq L_1 \\ \Box q_{\rho, t}^\downarrow \wedge \langle - \rangle q_t & \text{if } \rho^-|_{\Sigma} \subseteq L_1 \end{cases} \\ \delta_3(q_{\rho, t}^\downarrow, \sigma) &= \begin{cases} \text{true} & \text{if } \rho|_{\Sigma} \not\subseteq L_1 \\ q_t & \text{if } \rho|_{\Sigma} \subseteq L_1 \end{cases} \\ \delta_3(q_t^3, \sigma) &= \Box q_t^3 \wedge [-] q_t^3 \wedge q_t \wedge q_t^{3b} \\ \delta_3(q_t^{3b}, \sigma) &= \begin{cases} \text{true} & \text{if } L_0 = \emptyset \text{ or } \mathcal{I}_{\mathcal{T}_2, t}|_{\Sigma}^{\text{con}} \not\rightarrow_{\Sigma}^{\text{fin}} \mathcal{I}_{\mathcal{T}_1, L_1 \cap \mathbf{N}_{\mathbf{C}}} \\ \text{false} & \text{otherwise} \end{cases} \\ \delta_3(q_r, \sigma) &= \begin{cases} \text{true} & \text{if there is } \Sigma\text{-role } s \in L_2 \setminus L_1 \\ \text{false} & \text{otherwise} \end{cases} \end{aligned}$$

The automaton \mathfrak{A}'_4 is a variant of \mathfrak{A}_4 which drops states q_t^3 , q_t^{3b} , and all $q_{\rho,t}$, $q_{\rho,t}^\downarrow$ with $|\rho| > 1$ (and all according transitions), and replaces the transitions for q_1 and q_t as follows:

$$\delta_3(q_1, \sigma) = q_r \vee q_t \quad \text{for } t = L_2 \cap N_C$$

$$\delta_3(q_t, \sigma) = \begin{cases} \text{true} & \text{if } t|_\Sigma \not\subseteq L_1 \\ \bigvee_{t'|t \rightsquigarrow_{\rho} t'} \bigvee_{r \in \rho} q_{\{r\},t'} & \text{otherwise} \end{cases}$$

In this way it verifies that either Condition 1 above is satisfied or the variant 2' of Condition 2 is satisfied, for some $n \in \text{ind}(\mathcal{A})$:

1. there is $r \in \Sigma$ and $n' \in \text{ind}(\mathcal{A})$ such that $(n, n') \in r^{\mathcal{I}_2}$, but $(n, n') \notin r^{\mathcal{I}_1}$;
- 2' $\mathcal{J}_n \not\subseteq_{\Sigma} \mathcal{I}_1$.

This finishes the proof of Lemma 4.4.8.

Matching lower bounds for all problems except stCQ entailment are provided by [BLR⁺16]. They hold even in the case where $\Gamma = \Sigma$.

Corollary. *In any fragment of Horn- \mathcal{ALCHIF} that contains \mathcal{ELI} or Horn- \mathcal{ALC} , the following problems are 2EXPTIME-complete: (Γ, Σ) -CQ entailment, (Γ, Σ) -CQ inseparability, and (Γ, Σ) -CQ conservative extensions.*

4.5 Deductive Conservative Extensions

Another natural notion of entailment is deductive entailment, which generalizes the notion of deductive conservative extensions [GLW06, LWW07, KLWW09, LW10], and which separates two TBoxes in terms of concept and role inclusions and functionality assertions, instead of ABoxes and queries.

Definition 4.5.1. Let Σ be a signature and let \mathcal{T}_1 and \mathcal{T}_2 be \mathcal{ELHIF}_\perp TBoxes. We say that \mathcal{T}_1 Σ -deductively entails \mathcal{T}_2 , written $\mathcal{T}_1 \models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$, if for all Σ - \mathcal{ELI}_\perp -concept inclusions α and all Σ -RIs and Σ -FAs α : $\mathcal{T}_2 \models \alpha$ implies $\mathcal{T}_1 \models \alpha$. If additionally $\mathcal{T}_1 \subseteq \mathcal{T}_2$, then we say that \mathcal{T}_2 is a Σ -deductive conservative extension of \mathcal{T}_1 . If $\mathcal{T}_1 \models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$ and vice versa, then \mathcal{T}_1 and \mathcal{T}_2 are Σ -deductively inseparable.

Although closely related, it is not difficult to see that deductive and query entailment are orthogonal.

Example 4.5.1. (1) Let $\mathcal{T}_1, \mathcal{T}_2$ be as in Example 4.2.2 and $\Sigma = \{A_1, A_2, B\}$. Then $\mathcal{T}_1 \models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$, but $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$.

(2) Let $\mathcal{T}_1 = \emptyset$ and $\mathcal{T}_2 = \{A \sqsubseteq \exists r.B\}$, and $\Sigma = \{A, B\}$. Then $\mathcal{T}_1 \models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$, but $\mathcal{T}_1 \not\models_{\Sigma, \Sigma}^{\text{CQ}} \mathcal{T}_2$ as witnessed by $(\{A(a)\}, \exists x B(x), a)$. However, $\mathcal{T}_1 \models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$.

Nevertheless, the two notions are sufficiently closely related so that we have the following.

Lemma 4.5.1. *In \mathcal{ELHIF}_\perp , deductive entailment can be decided in polynomial time given access to oracles for stCQ entailment and stCQ evaluation.*

Lemma 4.5.1 is an immediate consequence of the following lemma because the additional $\mathcal{T}_1 \models_{\Sigma}^{\perp} \mathcal{T}_2$ can be reduced to stCQ entailment and stCQ evaluation via Lemma 4.2.3.

Lemma 4.5.2. *Let Σ be a signature and $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{ELHIF}_\perp$ TBoxes such that $\mathcal{T}_1 \models_{\Sigma, \Sigma}^{\text{RI}} \mathcal{T}_2$. Then*

$$\mathcal{T}_1 \models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2 \quad \text{iff} \quad \mathcal{T}_1 \models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2 \quad \text{and} \quad \mathcal{T}_1 \models_{\Sigma}^{\perp} \mathcal{T}_2.$$

Proof. We prove both implications via contraposition.

“ \Leftarrow ”. We assume that $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$. In case this is witnessed by a Σ -FA $\text{func}(r)$, we immediately get a witness Σ -ABox $= \{r(a, b), r(a, c)\}$ for $\mathcal{T}_1 \not\models_{\Sigma}^{\perp} \mathcal{T}_2$ and are done.

Otherwise, \mathcal{T}_1 contains all Σ -FAs from \mathcal{T}_2 , and there is a witness Σ -CI $C \sqsubseteq D$ (witness RIs are excluded by the assumption $\mathcal{T}_1 \models_{\Sigma, \Sigma}^{\text{RI}} \mathcal{T}_2$). Since \mathcal{ELI}_\perp -concepts that contain \perp are equivalent to \perp , the left-hand side C cannot contain \perp (i.e., is an \mathcal{ELI} concept) and, if D does, then $C \sqsubseteq \perp$ is a witness. We show that such witnesses give rise to either a witness \mathcal{A}_C for $\mathcal{T}_1 \not\models_{\Sigma}^{\perp} \mathcal{T}_2$ or a witness (\mathcal{A}_C, q_D, a) for $\mathcal{T}_1 \not\models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$ with $q_D(x)$ an stCQ .

We first consider the case that there is a witness $C \sqsubseteq \perp$ with C an \mathcal{ELI} concept. We can construct from C in the obvious way a tree-shaped Σ -ABox \mathcal{A}_C and root a : \mathcal{A} reflects the tree structure of C ; however, to respect the Σ -FAs in \mathcal{T}_1 (and thus those in \mathcal{T}_2), we need to merge the subtrees of all nodes that are r -neighbors of the same node, whenever $\text{func}(r) \in \mathcal{T}_1$. Consider the universal model $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}_C}$ ³ and observe that $a \in C^{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}_C}}$ from the construction of $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}_C}$. Since $\mathcal{T}_2 \models C \sqsubseteq \perp$, we have that $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}_C}$ is not a model of \mathcal{T}_2 . Hence, by the contrapositive of Lemma 4.2.2 (1), \mathcal{A}_C is inconsistent with \mathcal{T}_2 . On the other hand, since $\mathcal{T}_1 \not\models C \sqsubseteq \perp$, there is a model $\mathcal{I} \models \mathcal{T}_1$ and an instance $d \in C^{\mathcal{I}}$. We can turn \mathcal{I} into a model of \mathcal{A}_C by interpreting the ABox individuals accordingly (“partial” unraveling might be necessary to ensure that the standard name assumption is respected), witnessing the consistency of \mathcal{A} with \mathcal{T}_1 . We thus have $\mathcal{T}_1 \not\models_{\Sigma}^{\perp} \mathcal{T}_2$ and are done.

In the second case, *all* witnesses $C \sqsubseteq D$ consist solely of \mathcal{ELI} concepts C, D . We construct the same ABox \mathcal{A}_C with root a from C and transform D into a Σ -stCQ $q_D(x)$ with a single answer variable that represents the tree shape of D . Now (\mathcal{A}_C, q_D, a) is a witness to $\mathcal{T}_1 \not\models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$ for the following reasons.

- \mathcal{A}_C is consistent with \mathcal{T}_1 : a model can be obtained in the obvious way from the model witnessing $\mathcal{T}_1 \not\models C \sqsubseteq D$ (possibly involving “partial” unraveling as above).
- \mathcal{A}_C is consistent with \mathcal{T}_2 : since $C \sqsubseteq \perp$ is not a witness to $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$, there must be a model $\mathcal{I} \models \mathcal{T}_2$ with $d \in C^{\mathcal{I}}$. We claim that we can turn \mathcal{I} into a model of \mathcal{A}_C by interpreting the ABox individuals without violating the standard name assumption. If we assume to the contrary that this is not possible, then there are subconcepts C_1, \dots, C_n of C corresponding to subtrees that have been merged in the construction of \mathcal{A}_C , such that $\mathcal{T}_2 \models C_1 \sqcap \dots \sqcap C_n \sqsubseteq \perp$. However, $\mathcal{T}_1 \not\models C_1 \sqcap \dots \sqcap C_n \sqsubseteq \perp$ because \mathcal{A}_C is consistent with \mathcal{T}_1 , as shown previously. Hence $C_1 \sqcap \dots \sqcap C_n \sqsubseteq \perp$ would be a witness to $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_\perp} \mathcal{T}_2$, which we have ruled out – a contradiction.
- $\mathcal{T}_2, \mathcal{A}_C \models q_D(a)$, witnessed by $\mathcal{I}_{\mathcal{T}_2, \mathcal{A}_C}$, together with $a \in C^{\mathcal{I}_{\mathcal{T}_2, \mathcal{A}_C}}$ and $\mathcal{T}_2 \models C \sqsubseteq D$.
- $\mathcal{T}_1, \mathcal{A}_C \not\models q_D(a)$: take a model \mathcal{I} witnessing $\mathcal{T}_1 \not\models C \sqsubseteq D$ and an element $d \in C^{\mathcal{I}} \setminus D^{\mathcal{I}}$. As in the previous case, we can turn \mathcal{I} into a model \mathcal{J} of \mathcal{A}_C by interpreting the ABox individuals (again involving unraveling if necessary), obtaining $\mathcal{J} \not\models q_D(a)$.

³The assumption that \mathcal{A} is consistent with \mathcal{T} is not needed for the construction of $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$, only for the proof of Lemma 4.2.2 (1).

“ \Rightarrow ”. Assume $\mathcal{T}_1 \not\models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$ or $\mathcal{T}_1 \not\models_{\Sigma}^{\perp} \mathcal{T}_2$.

In case $\mathcal{T}_1 \not\models_{\Sigma}^{\perp} \mathcal{T}_2$, consider a witness Σ -Box \mathcal{A} and assume w.l.o.g. that \mathcal{A} is tree-shaped. Let $a \in \text{ind}(\mathcal{A})$ be its root. We can assume that \mathcal{T}_1 contains all Σ -FAs from \mathcal{T}_2 (otherwise $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_{\perp}} \mathcal{T}_2$ and we are done). We turn \mathcal{A} into a Σ - \mathcal{ELI} concept $C_{\mathcal{A}}$ in the obvious way. Then $C_{\mathcal{A}} \sqsubseteq \perp$ is a witness to $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_{\perp}} \mathcal{T}_2$:

- $\mathcal{T}_2 \models C_{\mathcal{A}} \sqsubseteq \perp$ because, if there were a model \mathcal{I} of \mathcal{T}_2 with $d \in C_{\mathcal{A}}^{\mathcal{I}}$, we could turn it into a model of $(\mathcal{T}_2, \mathcal{A})$ by interpreting the ABox individuals accordingly (possibly involving partial unraveling as above), which would contradict the assumption that \mathcal{A} is a witness to $\mathcal{T}_1 \not\models_{\Sigma}^{\perp} \mathcal{T}_2$.
- $\mathcal{T}_1 \not\models C_{\mathcal{A}} \sqsubseteq \perp$, witnessed by $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

In case $\mathcal{T}_1 \not\models_{\Sigma, \Sigma}^{\text{stCQ}} \mathcal{T}_2$, by Lemma 4.3.2 there is a witness (\mathcal{A}, q, a) with \mathcal{A} tree-shaped and q a Σ -stCQ with exactly one answer variable. We construct $C_{\mathcal{A}}$ as above and another Σ - \mathcal{ELI} concept D_q from q in the obvious way. It can be shown analogously to the previous case that $C_{\mathcal{A}} \sqsubseteq D_q$ is a witness to $\mathcal{T}_1 \not\models_{\Sigma}^{\mathcal{ELHIF}_{\perp}} \mathcal{T}_2$. \square

The following theorem follows from Lemma 4.5.1 and Theorem 4.4.7.

Theorem 4.5.3. *In \mathcal{ELHIF}_{\perp} , the following problems can be decided in time $2^{2^p(|\mathcal{T}_2| \log |\mathcal{T}_1|)}$, p a polynomial: deductive Σ -entailment, deductive Σ -inseparability, and deductive conservative extensions.*

We establish a CONEXPTIME lower bound and leave the precise complexity open.

4.5.1 Lower Bound

In the description logic \mathcal{EL} , which is \mathcal{ELI} without inverse roles, deductive conservative extensions and deductive Σ -entailment are EXPTIME-complete [LW10]. This raises the question whether the upper bound for deductive entailment reported in Theorem 4.5.3 is tight. While we leave this question open, we observe that the transition from \mathcal{EL} to \mathcal{ELI} does increase the complexity of deductive conservative extensions and related problems to at least CONEXPTIME. We consider this a surprising result since in reasoning problems that are not defined in terms of conjunctive queries, adding inverse roles does typically not result in an increase of complexity. The following is established by a non-trivial reduction of a tiling problem.

Theorem 4.5.4. *In any DL between \mathcal{ELI} and \mathcal{ELHIF}_{\perp} , deductive conservative extensions, deductive Σ -entailment, and deductive Σ -inseparability are CONEXPTIME-hard.*

The proof is by reduction of a NEXPTIME-complete tiling problem, where the aim is to tile a $2^n \times 2^n$ -grid, to the complement of stCQ-conservative extensions. This tiling problem was introduced as a special case of the *origin constrained domino problem* by Grädel [Grä89], and its NEXPTIME-hardness follows from Grädel’s Theorem 3.3. An instance is given by a tuple $P = (\mathfrak{T}, \mathfrak{T}_0, H, V)$, where \mathfrak{T} is a finite set of *tile types*, $\mathfrak{T}_0 \subseteq \mathfrak{T}$ is a set of *distinguished tiles* to be placed on position $(0, 0)$ of the grid, and H and V are horizontal and vertical matching conditions. Let $|\mathfrak{T}| = n$. A *solution* to P is a function $\tau : 2^n \times 2^n \rightarrow \mathfrak{T}$ such that

- if $\tau(i, j) = t$ and $\tau(i + 1, j) = t'$ then $(t, t') \in H$, for all $i < 2^n - 1, j < 2^n$,
- if $\tau(i, j) = t$ and $\tau(i, j + 1) = t'$ then $(t, t') \in V$, for all $i < 2^n, j < 2^n - 1$,

- $\tau(0, 0) \in \mathfrak{T}_0$.

We can assume w.l.o.g. that for every tile $t \in \mathfrak{T}$, there is a t' with $(t', t) \in V$.

Let $P = (\mathfrak{T}, \mathfrak{T}_0, H, V)$. We show how to construct \mathcal{ELI} TBoxes \mathcal{T}_1 and \mathcal{T}_2 such that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a $(\text{sig}(\mathcal{T}_1), \text{sig}(\mathcal{T}_1))$ -stCQ-conservative extension of \mathcal{T}_1 iff there is no solution for P . Hence, stCQ-conservative extensions, (Γ, Σ) -stCQ entailment, and (Γ, Σ) -stCQ inseparability are CONEXPTIME-hard in \mathcal{ELI} (and any DL that contains it as a fragment). Since \mathcal{T}_1 and \mathcal{T}_2 are formulated in \mathcal{ELI} , we trivially have $\mathcal{T}_1 \models_{\text{sig}(\mathcal{T}_1)}^\perp \mathcal{T}_1 \cup \mathcal{T}_2$. Thus, hardness of deductive conservative extensions follows from Lemma 4.5.2, in all DLs between \mathcal{ELI} and \mathcal{ELHIF}_\perp since \mathcal{T}_1 and \mathcal{T}_2 are formulated in \mathcal{ELI} and Lemma 4.5.2 covers deductive conservative extensions in \mathcal{ELHIF}_\perp . This also implies hardness of deductive Σ -entailment and of deductive Σ -inseparability in the mentioned DLs.

The intuitions and correctness proofs are based on the following characterization of stCQ-conservative extensions.

Lemma 4.5.5. *Let \mathcal{T}_1 and \mathcal{T}_2 be \mathcal{ELI} TBoxes such that all role names in \mathcal{T}_2 are in $\text{sig}(\mathcal{T}_1)$. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a $(\text{sig}(\mathcal{T}_1), \text{sig}(\mathcal{T}_1))$ -stCQ-conservative extension of \mathcal{T}_1 iff $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}} \rightarrow_{\text{sig}(\mathcal{T}_1)} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ for all tree-shaped $\text{sig}(\mathcal{T}_1)$ -ABoxes \mathcal{A} .*

Proof. An interpretation is *strongly tree-shaped* if it is tree-shaped and does not contain multi-edges, that is, any $d, d' \in \Delta^{\mathcal{I}}$ are involved in at most one role edge. Since \mathcal{T}_1 and \mathcal{T}_2 are formulated in \mathcal{ELI} (and thus do not contain role inclusions), for any tree-shaped ABox \mathcal{A} the universal models $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ and $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ are strongly tree-shaped. The assumption on role names in \mathcal{T}_2 made in the lemma implies that every element in $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ can be reached from an ABox individual by traveling only along $\text{sig}(\mathcal{T}_1)$ -roles. Together, this implies the following:

- (*) there is a $\text{sig}(\mathcal{T}_1)$ -simulation from $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$ iff there is a $\text{sig}(\mathcal{T}_1)$ -homomorphism from $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

From Theorem 4.3.4, we get that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a $(\text{sig}(\mathcal{T}_1), \text{sig}(\mathcal{T}_1))$ -stCQ-conservative extension of \mathcal{T}_1 iff $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}} \Big|_{\text{sig}(\mathcal{T}_1)}^{\text{con}} \preceq_{\text{sig}(\mathcal{T}_1)} \mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$. But $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}} \Big|_{\text{sig}(\mathcal{T}_1)}^{\text{con}} = \mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ by the assumption on role names in \mathcal{T}_2 made in the lemma and simulations can be replaced with homomorphisms by (*). \square

We will build \mathcal{T}_1 and \mathcal{T}_2 such that the same single role name r is used in \mathcal{T}_1 and \mathcal{T}_2 , thus the assumption in Lemma 4.5.5 will be satisfied.

For a clearer presentation, we proceed in two steps. We first define \mathcal{T}_1 and \mathcal{T}_2 to be an \mathcal{ELIU} -TBox, i.e., on both sides of CIs we allow concepts of the following form:

$$L, L' ::= \top \mid A \mid L \sqcap L' \mid L \sqcup L' \mid \exists r.L.$$

The only non-trivial use of disjunction will be on the right-hand side of a CI in \mathcal{T}_2 . In a second step, we show how to remove disjunction.

We use S to abbreviate the role composition $r; r^-$, writing for example $\exists S.C$ for $\exists r.\exists r^-.C$. Note that S behaves like a reflexive-symmetric role.⁴ Ideally, we would like \mathcal{T}_1 to be empty (except introducing the required symbols) and \mathcal{T}_2 to verify the existence of an S -path in the input ABox whose individuals represent the grid positions

⁴We will make sure that all ‘relevant domain elements’ have an r -successor, which guarantees reflexivity.

along with a tiling, row by row from left to right, starting at the lower left corner and ending at the upper right corner. The positions in the grid are represented in binary by the concept names X_1, \dots, X_{2n} in the ABox where X_1, \dots, X_n indicate the horizontal position and X_{n+1}, \dots, X_{2n} the vertical position. The tiling is represented by concept names $T_t, t \in \mathfrak{T}$. The verification is done by propagating a concept name as a marker bottom up and while doing this, verifying the horizontal matching condition. Under the assumption that an additional labeling with concept names $T'_t, t \in \mathfrak{T}$, is such that

- (*) every point in the path is labeled with T'_t if its descendant at distance exactly 2^n (that is, the grid position immediately below it) is labeled with T_t ,

the vertical matching condition is also verified.

For several reasons, this program cannot quite be implemented in the desired way. First, we still have to make sure that (*) actually holds. This is done as follows. We install yet another labeling with concept names $\overline{T}_t, t \in \mathfrak{T}$, such that a node is labeled with \overline{T}_t if it is not labeled with T_t . Then \mathcal{T}_2 checks for a violation of (*) in the following way: when the propagation reaches the final individual the verified path, the generation of a finite anonymous S -path is triggered. That path homomorphically embeds into the S -path in the ABox in many different ways since S is reflexiv-symmetric. In fact, for any individual on the ABox path we can find a homomorphism such that the endpoint of the anonymous path maps to that individual because (a) the anonymous path is long enough to reach the first individual on the ABox path and (b) the homomorphism can always ‘fold’ the reflexive-symmetric role S in a suitable way. At the end of the anonymous path, we then guess (using disjunction) a tile t , make T'_t true, continue building the anonymous path for another 2^n steps (in a way such that it cannot fold), and finally make \overline{T}_t true. Let us pretend for a second that our TBoxes are formulated in \mathcal{ELI} . If (*) is violated, then the guess can be made such that the anonymous path homomorphically maps into the ABox path. Otherwise, this is not the case. Clearly, the latter can occur only if P has a solution.

The fact that S is reflexive-symmetric allows the mentioned folding of the existential path. However, it poses some complications in the verification of the S -path in the ABox because we must be careful not to confuse successors with predecessors. To this end, every grid position is actually represented by three consecutive individuals labeled with the concept names B_0, B_1, B_2 , respectively. All these individuals are labeled identically regarding the X -counter and the concept names T_t . We are going to enforce (*) for the B_2 -individuals and only these individuals also receive T'_t and \overline{T}_t labels (any other B_i would work as well). Another problem is that \mathcal{T}_2 cannot check all possible kinds of defects. In particular, it cannot detect the defect that an element is labeled with more than one tile or that there are multiple successors in the ABox that have an incompatible labeling with the counter concept names. We thus use \mathcal{T}_1 to check for such defects. If found, it will generate a defect of the kind that \mathcal{T}_2 can verify, that is, a violation of (*).

We start with assembling \mathcal{T}_2 , which uses a single role name r via the abbreviation S introduced above and the following concept names.

- jointly with \mathcal{T}_1 :
 - $X_1, \dots, X_{2n}, \overline{X}_1, \dots, \overline{X}_{2n}$ for the binary representation of the horizontal and vertical grid positions on the ABox path
 - B_0, B_1, B_2 for distinguishing successors and predecessors on the ABox path (these concept names implement a unary counter that counts modulo three)

- $T_t, \bar{T}_t, t \in \mathfrak{T}$, representing tile types present/not present at individuals on the ABox path
- $T'_t, t \in \mathfrak{T}$, representing tile types present at the descendant at distance exactly $3 \cdot 2^n$ from the given individual on the ABox path
- additionally:
 - L as a verification marker to be propagated along the ABox path
 - $\text{ok}_i, 1 \leq i \leq 2n$, to indicate that the incrementation of the counter values at an ABox individual is correct regarding the i -th bit (the 1st bit being that of least value)
 - $Y_1, \dots, Y_{2n}, \bar{Y}_1, \dots, \bar{Y}_{2n}$ for counting the length of the anonymous path
 - $Y'_1, \dots, Y'_n, \bar{Y}'_1, \dots, \bar{Y}'_n$ implement another counter on the anonymous path, used to continue extending the path by exactly $3 \cdot 2^n$ positions to reach the grid position immediately below
 - B'_0, B'_1, B'_2 for distinguishing successors and predecessors on the anonymous path
 - $M_t, t \in \mathfrak{T}$, for memorizing a tile type on the anonymous path.

\mathcal{T}_2 consists of the following CIs.

1. The initial grid position starts the propagation:

$$\bar{X}_1 \sqcap \dots \sqcap \bar{X}_{2n} \sqcap B_0 \sqcap \bigsqcup_{t \in \mathfrak{T}_0} T_t \sqsubseteq L$$

2. The verification proceeds upwards. We first verify that the counter is incremented properly when moving upwards along the S -path in the ABox:

$$\begin{aligned} B_0 \sqcap X_i \sqcap \exists S.(B_2 \sqcap X_i) \sqcap \bigsqcup_{1 \leq j < i} \exists S.(B_2 \sqcap X_j) &\sqsubseteq \text{ok}_i \\ B_0 \sqcap \bar{X}_i \sqcap \exists S.(B_2 \sqcap \bar{X}_i) \sqcap \bigsqcup_{1 \leq j < i} \exists S.(B_2 \sqcap X_j) &\sqsubseteq \text{ok}_i \\ B_0 \sqcap X_i \sqcap \exists S.(B_2 \sqcap \bar{X}_i) \sqcap \prod_{1 \leq j < i} \exists S.(B_2 \sqcap \bar{X}_j) &\sqsubseteq \text{ok}_i \\ B_0 \sqcap \bar{X}_i \sqcap \exists S.(B_2 \sqcap X_i) \sqcap \prod_{1 \leq j < i} \exists S.(B_2 \sqcap \bar{X}_j) &\sqsubseteq \text{ok}_i \\ B_{j+1} \sqcap X_i \sqcap \exists S.(B_j \sqcap X_i) &\sqsubseteq \text{ok}_i \\ B_{j+1} \sqcap \bar{X}_i \sqcap \exists S.(B_j \sqcap \bar{X}_i) &\sqsubseteq \text{ok}_i \end{aligned}$$

where i ranges over $1..2n$ and j over $\{0, 1\}$. These inclusions only work under the assumption that no individual has two S -neighbors that are labeled with the same B_i but are labeled differently regarding X_j and \bar{X}_j for some j . We shall prevent this situation later using \mathcal{T}_1 .

3. We next make a verification step inside a row of the grid:

$$\begin{aligned} B_0 \sqcap \text{ok}_1 \sqcap \dots \sqcap \text{ok}_n \sqcap T_{t_2} \sqcap \\ \bar{X}_i \sqcap \exists S.(B_2 \sqcap L \sqcap T_{t_1}) &\sqsubseteq L \\ B_1 \sqcap \text{ok}_1 \sqcap \dots \sqcap \text{ok}_n \sqcap T_{t_2} \sqcap \\ \bar{X}_i \sqcap \exists S.(B_0 \sqcap L \sqcap T_{t_1}) &\sqsubseteq L \\ B_2 \sqcap \text{ok}_1 \sqcap \dots \sqcap \text{ok}_n \sqcap T_{t_2} \sqcap \prod_{t \in \mathfrak{T} \setminus \{t_2\}} \bar{T}_t \sqcap T'_{t_3} \sqcap \\ \bar{X}_i \sqcap \exists S.(B_1 \sqcap L \sqcap T_{t_2}) &\sqsubseteq L \end{aligned}$$

where t_1, t_2, t_3 range over \mathfrak{T} such that $(t_1, t_2) \in H$ and $(t_3, t_2) \in V$ and i ranges over $1..n$. The use of \bar{X}_i on the left-hand sides ensures that we move inside a row. In the first line, we make a move between horizontally neighboring grid positions, verifying the horizontal matching condition. In the other lines, we move along the three points representing the same grid position, ensuring that they are all labeled by the same tile.⁵

4. We also have to consider the case where we jump from one grid row to the next, ignoring the tiling condition:

$$B_0 \sqcap \text{ok}_1 \sqcap \dots \sqcap \text{ok}_n \sqcap T_t \sqcap \\ \bar{X}_i \sqcap \exists S.(B_2 \sqcap L) \sqsubseteq L$$

where t ranges over \mathfrak{T} and i ranges over $n+1..2n$. The use of \bar{X}_i on the left-hand side ensures that we are not yet in the topmost row.

5. When the final individual of the ABox path is reached (maximum counter value and B_2 -label), we make an extra step in the ABox to a B_0 -labeled individual and then generate the first object of an existential path:

$$B_0 \sqcap \exists S.(B_2 \sqcap X_1 \sqcap \dots \sqcap X_{2n} \sqcap L) \sqsubseteq \exists S.C$$

where

$$C = Y_1 \sqcap \dots \sqcap Y_{2n} \sqcap B'_2$$

The purpose of the extra step will be explained later on.

6. Here and in the following, we use the abbreviation

$$\exists S^{(3)}.(X_1, X_2, X_3) := \exists S.(X_1 \sqcap \exists S.(X_2 \sqcap \exists S.X_3))$$

for concept names X_1, X_2, X_3 .

We continue building the path, decrementing the Y -counter:

$$Y_i \sqcap B'_2 \sqsubseteq \exists S^{(3)}.(B'_1, B'_0, B'_2) \\ B'_j \sqcap \exists S.(B'_{j+1} \sqcap Y_i) \sqsubseteq Y_i \\ B'_j \sqcap \exists S.(B'_{j+1} \sqcap \bar{Y}_i) \sqsubseteq \bar{Y}_i \\ B'_2 \sqcap \exists S.(B'_0 \sqcap \prod_{1 \leq j \leq i} \bar{Y}_i) \sqsubseteq Y_i \\ B'_2 \sqcap \exists S.(B'_0 \sqcap Y_i \sqcap \prod_{1 \leq j < i} \bar{Y}_i) \sqsubseteq \bar{Y}_i \\ B'_2 \sqcap \exists S.(B'_0 \sqcap Y_i \sqcap \bigsqcup_{1 \leq j < i} Y_i) \sqsubseteq Y_i \\ B'_2 \sqcap \exists S.(B'_0 \sqcap \bar{Y}_i \sqcap \bigsqcup_{1 \leq j < i} Y_i) \sqsubseteq \bar{Y}_i$$

where i ranges over $1..2n$ and j over $\{0, 1\}$. It is essential to use different B'_i and counter concepts Y_i, \bar{Y}_i than in the ABox; otherwise the anonymous path could not homomorphically embed into the ABox path in a folded way. It would actually suffice to build a path of length $3 \cdot (2^{2n-1})$ because no violation of (*) can start in the bottommost row. However, overcounting does not compromise correctness.

⁵Note that we expect to see T'_t labels also in row 0; this is why we assume that for every $t \in \mathfrak{T}$, there is a $(t', t) \in V$; we could avoid the assumption at the cost of dealing with row 0 as a special case.

7. At the end of the anonymous path, we implement a violation of (*) as described above: we guess a tile t involved in the violation, make sure that T'_t holds at the current point, start a new counter, travel exactly $3 \cdot 2^n$ steps (without any folding), and verify that \bar{T}_t holds where we arrive:

$$\begin{aligned}
& \bar{Y}_1 \sqcap \dots \sqcap \bar{Y}_{2n} \sqcap B'_2 \sqsubseteq Y'_1 \sqcap \dots \sqcap Y'_n \sqcap \\
& \quad \bigsqcup_{t \in \mathfrak{T}} (T'_t \sqcap M_t) \\
& \quad Y'_i \sqcap B_2 \sqsubseteq \exists S^{(3)}.(B_0, B_1, B_2) \\
& B_{j+1} \sqcap \exists S.(B_j \sqcap Y'_i) \sqsubseteq Y'_i \\
& B_{j+1} \sqcap \exists S.(B_j \sqcap \bar{Y}'_i) \sqsubseteq \bar{Y}'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap \prod_{1 \leq j \leq i} \bar{Y}'_j) \sqsubseteq Y'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap Y'_i \sqcap \prod_{1 \leq j < i} \bar{Y}'_j) \sqsubseteq \bar{Y}'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap Y'_i \sqcap \bigsqcup_{1 \leq j < i} Y'_j) \sqsubseteq Y'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap \bar{Y}'_i \sqcap \bigsqcup_{1 \leq j < i} Y'_j) \sqsubseteq \bar{Y}'_i \\
& B_{j+1} \sqcap \exists S.(B_j \sqcap M_t) \sqsubseteq M_t \\
& B_0 \sqcap \exists S.(B_2 \sqcap Y'_i \sqcap M_t) \sqsubseteq M_t \\
& \bar{Y}'_1 \sqcap \dots \sqcap \bar{Y}'_n \sqcap M_t \sqsubseteq \bar{T}_t
\end{aligned}$$

where i ranges over $1..n$ and t over \mathfrak{T} . Here we use the same B_i as in the ABox to avoid folding.

This finishes the definition of \mathcal{T}_2 . We now define \mathcal{T}_1 , which uses the following additional concept names.

- D for indicating the occurrence of a defect
- Z_1, \dots, Z_n for an additional counter.

\mathcal{T}_1 consists of the following CIs.

1. Tiles are mutually exclusive: for all distinct $t, t' \in \mathfrak{T}$:

$$T_t \sqcap T_{t'} \sqsubseteq D$$

where D starts a path that implements a violation of (*), to be implemented below;

2. The problematic situation described at the end of Item 2 in the definition of \mathcal{T}_2 cannot occur:

$$\exists S.(B_k \sqcap X_i) \sqcap \exists S.(B_k \sqcap \bar{X}_i) \sqsubseteq D$$

where k ranges over $\{0, 1, 2\}$ and i over $1..2n$;

3. We next implement the path triggered by D :

$$\begin{aligned}
B_2 \sqcap (D \sqcup \exists S.D \sqcup \exists S.\exists S.D) &\sqsubseteq Z_1 \sqcap \dots \sqcap Z_n \sqcap T'_t \\
Z_i \sqcap B_2 &\sqsubseteq \exists S^{(3)}.(B_0, B_1, B_2) \\
B_{j+1} \sqcap \exists S.(B_j \sqcap Z_i) &\sqsubseteq Z_i \\
B_{j+1} \sqcap \exists S.(B_j \sqcap \bar{Z}_i) &\sqsubseteq \bar{Z}_i \\
B_0 \sqcap \exists S.(B_2 \sqcap \prod_{1 \leq j \leq i} \bar{Z}_i) &\sqsubseteq Z_i \\
B_0 \sqcap \exists S.(B_2 \sqcap Z_i \sqcap \prod_{1 \leq j < i} \bar{Z}_i) &\sqsubseteq \bar{Z}_i \\
B_0 \sqcap \exists S.(B_2 \sqcap Z_i \sqcap \bigsqcup_{1 \leq j < i} Z_i) &\sqsubseteq Z_i \\
B_0 \sqcap \exists S.(B_2 \sqcap \bar{Z}_i \sqcap \bigsqcup_{1 \leq j < i} Z_i) &\sqsubseteq \bar{Z}_i \\
\bar{Z}_1 \sqcap \dots \sqcap \bar{Z}_n &\sqsubseteq \bar{T}_t
\end{aligned}$$

where i ranges over $1..n$ and $t \in \mathfrak{T}$ is fixed.

Before we eliminate the disjunction used in \mathcal{T}_2 , let us mention the central property of \mathcal{T}_1 and \mathcal{T}_2 that can be used to show correctness of the reduction. Since \mathcal{T}_2 uses disjunction, a universal model for \mathcal{T}_2 and an ABox \mathcal{A} is not guaranteed to exist. Instead, there is a set of models for \mathcal{T}_2 and \mathcal{A} that is universal in the sense that for every model \mathcal{I} of \mathcal{T}_2 and \mathcal{A} , there is a model in the set that admits a homomorphism into \mathcal{I} . We refrain from giving a formal definition. Now, the central property of \mathcal{T}_1 and \mathcal{T}_2 is as follows: P has a solution iff there is a tree-shaped $\text{sig}(\mathcal{T}_1)$ -ABox \mathcal{A} consistent with $\mathcal{T}_1 \cup \mathcal{T}_2$ such that, for every \mathcal{I} in the universal set of models for $\mathcal{T}_1 \cup \mathcal{T}_2$ and \mathcal{A} , there is no $\text{sig}(\mathcal{T}_1)$ -homomorphism from \mathcal{I} to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

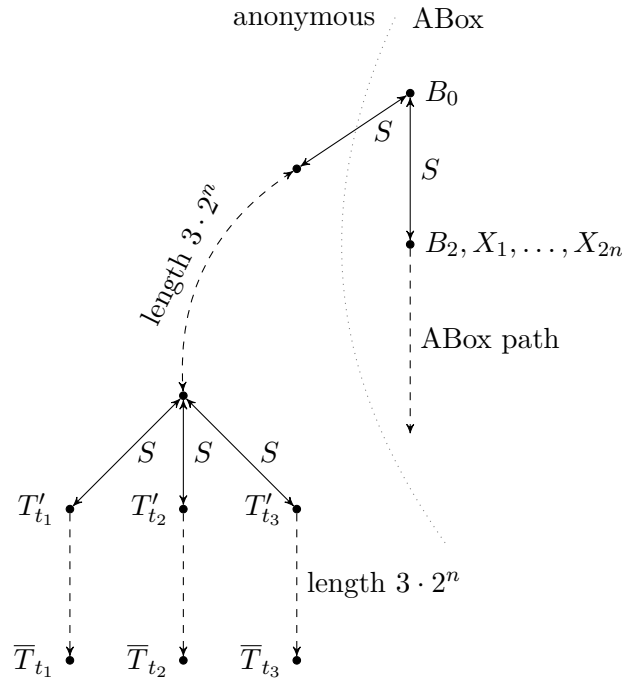
We now show how to get rid of disjunction. The central property of \mathcal{T}_1 and \mathcal{T}_2 will essentially be preserved, with a single universal model playing the role of the universal set of models. The disjunctions on the left-hand sides of CIs only serve as abbreviations and can easily be removed with only a polynomial blowup of TBox sizes. What remains is the disjunction in Item 7 of the definition of \mathcal{T}_2 . To get rid of it, we need to modify both \mathcal{T}_1 and \mathcal{T}_2 :

- In the first line of Point 7 of the definition of \mathcal{T}_2 , the disjunction is replaced with a conjunction, generating $|\mathfrak{T}|$ many defective chains at once:

$$\bar{Y}_1 \sqcap \dots \sqcap \bar{Y}_{2n} \sqsubseteq \prod_{t \in \mathfrak{T}} \exists S.(Y'_1 \sqcap \dots \sqcap Y'_n \sqcap B_2 \sqcap T'_t \sqcap M_t)$$

The resulting universal model is illustrated in Figure 4.1 where we assume $\mathfrak{T} = \{t_1, t_2, t_3\}$, showing the final individual on the ABox path, the extra step from Item 5 of the definition of \mathcal{T}_2 , the anonymous path, and the branching gadget attached to the end of it.

- We have now generated too many paths and thus the desired homomorphism may not exist even if (*) is violated in the ABox. We compensate by enforcing in \mathcal{T}_1 that when a B_2 -individual in the ABox path is labeled with T'_t , then it's B_0 -predecessor on that path roots $|\mathfrak{T}| - 1$ many additional paths, realizing every possible violation of (*) except the one induced by $t \in \mathfrak{T}$. This is illustrated in Figure 4.2 where we again assume $\mathfrak{T} = \{t_1, t_2, t_3\}$, showing a B_2 -individual labeled with T'_{t_1} and the extra successors of it's B_0 -predecessor generated by \mathcal{T}_1 . Note that this explains the extra step in Item 5 of the definition of \mathcal{T}_2 since also the final B_2 -element of the ABox path must have a B_0 -predecessor.

Figure 4.1: Part of the universal model of \mathcal{T}_2 .

We add the following to \mathcal{T}_1 , using fresh concept names M'_t , $t \in \mathfrak{T}$:

$$\begin{aligned}
& B_0 \sqcap \exists S.(B_2 \sqcap T'_i) \sqsubseteq \prod_{t' \in \mathfrak{T} \setminus \{t\}} \exists S.(\\
& \quad Y'_1 \sqcap \dots \sqcap Y'_n \sqcap B_2 \sqcap M'_{t'}) \\
& \quad Y'_i \sqcap B_2 \sqsubseteq \exists S^{(3)}.(B_0, B_1, B_2) \\
& B_{j+1} \sqcap \exists S.(B_j \sqcap Y'_i) \sqsubseteq Y'_i \\
& B_{j+1} \sqcap \exists S.(B_j \sqcap \bar{Y}'_i) \sqsubseteq \bar{Y}'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap \prod_{1 \leq j \leq i} \bar{Y}'_j) \sqsubseteq Y'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap Y'_i \sqcap \prod_{1 \leq j < i} \bar{Y}'_j) \sqsubseteq \bar{Y}'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap Y'_i \sqcap \bigsqcup_{1 \leq j < i} Y'_j) \sqsubseteq Y'_i \\
& B_0 \sqcap \exists S.(B_2 \sqcap \bar{Y}'_i \sqcap \bigsqcup_{1 \leq j < i} Y'_j) \sqsubseteq \bar{Y}'_i \\
& B_{j+1} \sqcap \exists S.(B_j \sqcap M'_t) \sqsubseteq Y'_i \sqcap M'_t \\
& B_0 \sqcap \exists S.(B_2 \sqcap Y'_i \sqcap M'_t) \sqsubseteq M'_t \\
& \bar{Y}'_1 \sqcap \dots \sqcap \bar{Y}'_n \sqcap M'_t \sqsubseteq \bar{T}_t
\end{aligned}$$

where i ranges over $1..n$ and t over \mathfrak{T} .

Lemma 4.5.6. $\mathcal{T}_1 \cup \mathcal{T}_2$ is a $(\text{sig}(\mathcal{T}_1), \text{sig}(\mathcal{T}_1))$ -stCQ-conservative extension of \mathcal{T}_1 iff there is no solution for P .

Proof. By Lemma 4.5.5, it suffices to show the following.

Claim. P has a solution iff there is a tree-shaped $\text{sig}(\mathcal{T}_1)$ -ABox \mathcal{A} such that there is no $\text{sig}(\mathcal{T}_1)$ -homomorphism from $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

We now sketch a proof of the claim.

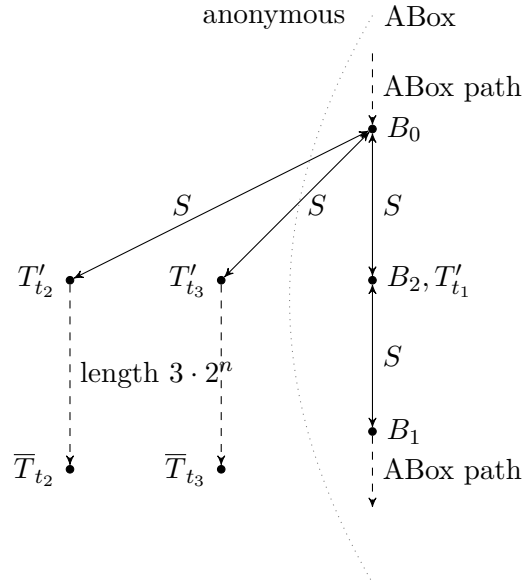


Figure 4.2: Extra successors to eliminate disjunction

“ \Rightarrow ”. Assume that P has a solution. Let \mathcal{A} be the ABox that contains a single S -path of length $3 \cdot 2^n$ which correctly encodes the solution to P via the concept names $X_i, \bar{X}_i, B_i, T_t, \bar{T}_t, T'_i$. Since \mathcal{A} correctly encodes the tiling, there is no $\text{sig}(\mathcal{T}_1)$ -homomorphism from $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$; in particular, the anonymous path described by \mathcal{T}_2 that ends in $|\mathfrak{I}|$ many violations of $(*)$, each represented via a path, cannot be mapped to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$.

“ \Leftarrow ”. Assume P has no solution and let \mathcal{A} be a tree-shaped $\text{sig}(\mathcal{T}_1)$ -ABox. If \mathcal{A} contains no path of length $3 \cdot 2^n$ that is labeled in the desired way with the concept names $X_i, \bar{X}_i, B_i, T_t, \bar{T}_t, T'_i$ (and which does not necessarily satisfy $(*)$), then the generation of the anonymous path in $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ is not triggered and the identity is a $\text{sig}(\mathcal{T}_1)$ -homomorphism from $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$. If there is such a path, then it violates $(*)$ since P has no solution. Consequently, the anonymous path in $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ homomorphically maps to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$, which is sufficient to show that there is a homomorphism from $\mathcal{I}_{\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}}$ to $\mathcal{I}_{\mathcal{T}_1, \mathcal{A}}$. \square

4.6 Concluding Remarks

We studied the decidability and computational complexity of query conservative extensions in Horn DLs with inverse roles. This was more challenging than without inverse roles because characterizations in terms of unbounded homomorphisms between universal models fail, blocking the standard approach to establishing decidability. We thus resorted to a combination of automata and mosaic techniques, proving that the problem is 2EXPTIME-complete in Horn-*ALCHIF* (and also in Horn-*ALC* and in *ELI*).

Additionally, we studied the case of deductive entailment between *ELHIF* $_{\perp}$ -TBoxes and showed that it is equivalent to a restricted version of query entailment in which the queries contain no multi-edges and their roots are the only answer variable. We obtained a model theoretic characterization, a decision procedure, and a 2EXPTIME upper bound. We also gave a CONEXPTIME lower bound via a reduction of an NEXPTIME-complete

tiling problem.

As future work, it would be interesting to close the gap in complexity between CON-EXPTIME and 2EXPTIME for deductive entailment in \mathcal{ELI} and \mathcal{ELHIF}_\perp . Furthermore, the results presented in this chapter show that query conservative extensions in Horn DLs with inverse roles are closely related to query conservative extensions in Datalog. Therefore, it would be interesting to extend the results to ontology languages from the family of Datalog_-^+ (also known as existential rules), in particular, to frontier-guarded existential rules.

Ontology-mediated Querying in UNFO with Regular Path Expressions

In ontology-mediated querying, queries against incomplete and heterogeneous data are supported by an ontology that provides domain knowledge and assigns a semantics to the data [BO15, BtCLW14, KZ14, PLC⁺08]. The ontologies are often formulated in a specialized language such as a description logic [BCM⁺07, BHLS17] or an existential rule language [BLMS11, BMRT11, CGK08, GOPŠ12] while the actual query is typically a conjunctive query (CQ) or a mild extension thereof such as a union of CQs (UCQ). It is useful, however, to consider more expressive decidable fragments of first-order logic (FO) as an ontology language, such as the guarded fragment (GFO), the unary negation fragment (UNFO), and the guarded negation fragment (GNFO). These fragments originate from the attempt to explain the good computational behaviour of modal and description logics and to extend their expressive power in a natural way. It is an important result that, for all these fragments, ontology-mediated querying with UCQs remains decidable and that the complexity stays within the expected, namely 2EXPTIME in combined complexity and CONP in data complexity.

From the perspective of database theory, it is an attractive property of both UNFO and GNFO (but not of GFO) that they can express CQs and UCQs. In ontology-mediated querying, this allows to ‘express’ the evaluation of ontology-mediated queries in terms of satisfiability in a natural way. It is easiest to state this for Boolean queries: if (\mathcal{O}, Σ, q) is an ontology-mediated query (OMQ) where \mathcal{O} is an ontology, Σ a set of predicate symbols (that is, relation names) that may occur in the data, and q a UCQ, and D is a Σ -database, then $D \models (\mathcal{O}, \Sigma, q)$ iff $\mathcal{O} \wedge D \wedge \neg q$ is unsatisfiable. When \mathcal{O} is formulated in UNFO or in GNFO, then so is $\mathcal{O} \wedge D \wedge \neg q$. What is more, the containment of OMQs can also be ‘expressed’ as a satisfiability problem in the natural case where both OMQs contain the same ontology and Σ is the set of all predicates symbols; from now on, we generally mean this case when speaking of OMQ containment. But also beyond ontology-mediated querying, we believe that the ability to express UCQs makes UNFO and GNFO attractive as an expressive logical backdrop for database theory. While GNFO is attractive as UNFO as ontology language, we focus only in UNFO and study the question introduced in Section 1.2.2:

- Q4 What is the complexity of OMQ evaluation and OMQ containment when UNFO is extended with regular expressions and transitive relations?

In this chapter, we study the natural extension UNFO^{reg} of UNFO with regular path expressions on binary relations. The resulting logic has the attractive property that it allows to express regular path queries [CMW87] and conjunctive two-way regular path queries (C2RPQs) [CDLV00] as well as unions thereof (UC2RPQs). Such queries play a central role in the area of graph databases [AG08, Bar13] and they have also received considerable attention in ontology-mediated querying [BCOŠ14, BOŠ15a, BOŠ15b, CEO07, CEO09, CEO14, ORŠ11]. An additional reason to consider UNFO^{reg} is provided by the observation that transitive roles are an important feature of many common description logics (a role is a binary relation), but that transitive roles cannot be expressed in UNFO. In UNFO^{reg} , even transitive closure of roles and regular expressions on roles are expressible, two features that are provided by several expressive description logics [Baa91, CDLN01]. As a concrete example, every ontology formulated in $\mathcal{ALCI}^{\text{reg}}$, the extension of the common description logic \mathcal{ALCI} with regular expressions on roles [Sch91], can be expressed in UNFO^{reg} and thus the evaluation of ontology-mediated queries (\mathcal{O}, Σ, q) where \mathcal{O} is formulated in $\mathcal{ALCI}^{\text{reg}}$ and q is a UC2RPQ can be ‘expressed’ as a satisfiability problem in UNFO^{reg} ; of course, the same is true when \mathcal{O} is formulated in UNFO^{reg} itself. We remark that transitive roles cannot be expressed in GFO and GNFO either, and that adding transitive relations to GFO without losing decidability requires to impose rather strong syntactical restrictions [ST04], especially so in an ontology-mediated querying context [GPT13]. Adding transitive relations to GNFO has, to the best of our knowledge, not yet been studied.

The main problem that we are interested in is evaluating OMQs in which the ontology is formulated in UNFO^{reg} and the actual query is a UC2RPQ. We show that this problem is decidable, 2EXPTIME-complete in combined complexity and CONP-complete in data complexity. We further consider the OMQ containment problem and show that it is 2EXPTIME-complete as well. We additionally show that the complexity of model checking in UNFO^{reg} is the same as in UNFO, namely complete for $\text{P}^{\text{NP}[O(\log^2 n)]}$.

As explained above, both OMQ evaluation and OMQ containment can be reduced to satisfiability in polynomial time. For studying the combined complexity of the former and the complexity of the latter, we thus concentrate on the satisfiability problem and prove a 2EXPTIME upper bound. Note that the addition of regular expressions does thus not increase the complexity of this problem as satisfiability in UNFO is also 2EXPTIME-complete [tCS13] and that the lower bound holds already when the arity of predicates is bounded by two, as a consequence of the results in [Lut08]. Our proof proceeds by first showing that every satisfiable UNFO^{reg} formula φ has a model whose treewidth is bounded by the size of φ , then establishing a characterization of the satisfaction of C2RPQs (that occur as a building block in φ) in such models in terms of certain witness trees, and finally showing that this infrastructure gives rise to a decision procedure based on two-way alternating tree automata. This ‘direct approach’ is in contrast to the reduction to satisfiability in the μ -calculus used for UNFO in [tCS13] which seems unwieldy in the presence of regular path expressions. Note in particular that an important reason for the relative simplicity of the reduction in [tCS13] is that there is always a model of bounded treewidth in which any two bags overlap in at most one element; this is no longer true in UNFO^{reg} . To establish the CONP upper bound on data complexity, we first observe that it suffices to consider a database satisfiability problem (given a database D , is there a model of the fixed UNFO^{reg} sentence φ that extends D ?) and then establish a certain kind of decoration of D as a witness for D being a positive instance, in a way such that witnesses can be guessed and verified in polynomial time.

5.1 UNFO with Regular Path Expressions

We assume that a countably infinite supply of predicate symbols of each arity is available. In the *unary negation fragment of first-order logic extended with regular path expressions* (UNFO^{reg}), formulas φ are formed according to the following grammar:

$$\begin{aligned}\varphi &::= P(\mathbf{x}) \mid E(x, y) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \neg \varphi(x) \\ E &::= R \mid R^- \mid E \cup E \mid E \cdot E \mid E^* \mid \varphi(x)?\end{aligned}$$

where P ranges over predicate symbols, R over binary predicate symbols, and, in the $\neg \varphi(x)$ clause, φ has no free variables besides (possibly) x . Expressions E formed according to the second line are called (*regular*) *path expressions* and expressions $\varphi(x)?$ according to the last clause in that line are called *tests*. Tests are similar to the test operator in propositional dynamic logic (PDL) [FL79] and to node tests in XPath [GKP02] and in some versions of regular path queries [BOŠ13, BKR14]. When we write $\varphi(\mathbf{x})$, we generally mean that the free variables of φ are among \mathbf{x} , but not all variables from \mathbf{x} need actually be free in φ . For a UNFO^{reg} formula $\varphi(x)$, we use $\forall x \varphi$ to abbreviate $\neg \exists x \neg \varphi(x)$.

Example 5.1.1. The following are UNFO^{reg} formulas: $\forall x (\exists y R(x, y) \wedge \neg (R \cdot R^*)(x, x))$ and $\exists y (R^*(x, y) \wedge S^*(x, y))$.

A *structure* \mathfrak{A} takes the form $(A, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}, \dots)$ where A is a non-empty set called the *domain* and R_i is an n_i -ary relation over A if R_i is a predicate symbol of arity n_i . Whenever convenient, we use $\text{dom}(\mathfrak{A})$ to refer to A . Every path expression E is interpreted as a binary relation $E^{\mathfrak{A}}$ over A : $R^{\mathfrak{A}}$ is part of \mathfrak{A} , $(R^-)^{\mathfrak{A}}$ is the converse of $R^{\mathfrak{A}}$, $(E_1 \cup E_2)^{\mathfrak{A}} = E_1^{\mathfrak{A}} \cup E_2^{\mathfrak{A}}$, $(E_1 \circ E_2)^{\mathfrak{A}} = E_1^{\mathfrak{A}} \circ E_2^{\mathfrak{A}}$, $(E^*)^{\mathfrak{A}}$ is the reflexive-transitive closure of $E^{\mathfrak{A}}$, and $(\varphi(x)?)^{\mathfrak{A}} = \{(a, a) \mid \mathfrak{A} \models \varphi(a)\}$. UNFO^{reg} formulas are then interpreted under the standard first-order semantics with path expressions being treated in the same way as binary predicates. An UNFO^{reg} sentence $\varphi(\mathbf{x})$ is *satisfiable* if there is a structure \mathfrak{A} such that $\mathfrak{A} \models \varphi$. Such an \mathfrak{A} is called a *model* of $\varphi(\mathbf{x})$.

Example 5.1.2. Reconsider the UNFO^{reg} formulas from Example 5.1.1. It can be verified that the first sentence is satisfiable, but not in a finite model. Thus, in contrast to UNFO (and to propositional dynamic logic), UNFO^{reg} lacks the finite model property. The second sentence expresses a property that cannot be expressed in UNFO extended with fixed points, as studied in [tCS13], which can formally be shown using UN-bisimulations, also defined in [tCS13]. In fact, UNFO^{reg} and UNFO with fixed points are orthogonal in expressive power. Another related logic is ICPDL, that is, PDL extended with intersection and converse [GLL09]. This logic, too, is orthogonal in expressive power to UNFO^{reg} . For example, the existence of a 4-clique can be expressed as a UNFO sentence, but not in ICPDL since every satisfiable ICPDL formula is satisfiable in a structure of tree width two.

The expressive power of UNFO^{reg} is closely related to that of conjunctive 2-way regular path queries. A *database* D is a finite structure such that for every $a \in \text{dom}(D)$, there is an $\mathbf{a} \subseteq \text{dom}(D)$ and a predicate symbol P such that $a \in \mathbf{a} \in P^D$. Since a database is a syntactic object, we refer to the elements of $\text{dom}(D)$ as *constants* whereas we speak about *elements* in the context of (semantic) structures. A *conjunctive 2-way regular path query* (C2RPQ) is a formula of the form $q(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of *atoms* of the form $R(\mathbf{z})$ and $E(z_1, z_2)$, R a predicate symbol and E a

two-way regular path query, that is, an expression formed according to the second line of the syntax definition of UNFO^{reg} , but allowing only formulas $\varphi(x)$ that are C2RPQs in tests. The variables \mathbf{x} are the *answer variables* of $q(\mathbf{x})$ and $q(\mathbf{x})$ is *Boolean* if $\mathbf{x} = \emptyset$. A *union of C2RPQs* (UC2RPQ) is a disjunction of C2RPQs that all have the same answer variables. A *conjunctive query* (CQ) is a C2RPQ that does not use atoms of the form $E(z_1, z_2)$. The *answers* to a UC2RPQ $q(\mathbf{x})$ on a database D , denoted $\text{ans}(q, D)$, are defined in the standard way, see for example [RRV17]. Note that every UC2RPQ is a UNFO^{reg} formula.

Example 5.1.3. Consider the following database about family relationships, using binary predicates *Child* and *Spouse*, and written as a set of facts.

$$D = \{ \text{Child}(\text{Nívea}, \text{Clara}), \text{Child}(\text{Clara}, \text{Blanca}), \text{Child}(\text{Blanca}, \text{Alba}), \\ \text{Spouse}(\text{Nívea}, \text{Severo}), \text{Spouse}(\text{Esteban}, \text{Clara}) \}$$

The following C2RPQ asks for all pairs (x, y) such that x is an ancestor of y in a line of only married ancestors (using the shorthand $R^+ = R \cdot R^*$).

$$q(x, y) = (m(z)? \cdot \text{Child})^+(x, y) \quad \text{where} \quad m(z) = \exists z' (\text{Spouse} \cup \text{Spouse}^-)(z, z')$$

We have $\text{ans}(q, D) = \{(\text{Nívea}, \text{Clara}), (\text{Nívea}, \text{Blanca}), (\text{Clara}, \text{Blanca})\}$.

Let $q(\mathbf{x}) = \exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$ be a C2RPQ. We use $\text{var}(q)$ to denote the variables that occur in q outside of tests, that is, $\mathbf{x} \cup \mathbf{y}$. We do not distinguish between $q(\mathbf{x})$ and the set of all atoms in φ , writing e.g. $R(x, y, z) \in q(\mathbf{x})$ to mean that $R(x, y, z)$ is an atom in φ . For simplicity, we treat an atom $E(x, x)$ in a C2RPQ $q(\mathbf{x})$ where E is the test $\varphi(y)?$ as an atom of the form $\varphi(x)$; that is, w.l.o.g. we use tests not only in path expressions but also directly as atoms of a C2RPQ. A C2RPQ $q(\mathbf{x})$ can be viewed as a finite hypergraph in the expected way, that is, every atom $R(\mathbf{z})$ and $E(z_1, z_2)$ is viewed as a hyperedge. We say that $q(\mathbf{x})$ is *connected* if the Gaifman graph of this hypergraph is connected. It is interesting to observe that foundational problems concerning UC2RPQs can be phrased as (un)satisfiability problems in UNFO^{reg} .

Example 5.1.4.

1. The problem whether a Boolean UC2RPQ $q()$ evaluates to true on a database D (i.e., whether the empty tuple is in $\text{ans}(q, D)$) corresponds to the unsatisfiability of $\varphi_D() \wedge \neg q()$ where $\varphi_D()$ is D viewed as a Boolean CQ in the obvious way.
2. The problem whether a Boolean UC2RPQ $q_1()$ is contained in a Boolean UC2RPQ $q_2()$ (defined in the usual way) corresponds to the unsatisfiability of $q_1() \wedge \neg q_2()$.

Both reductions extend to the case of non-Boolean queries by simulating answer variables using fresh unary predicates, see the proof of Lemma 5.1.1 below.

5.1.1 Ontology-mediated Querying

We are primarily interested in ontology-mediated queries (\mathcal{O}, Σ, q) where \mathcal{O} is an UNFO^{reg} sentence and q is a UC2RPQ. We use $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ to denote the set of OMQs of this form and similarly for other ontology languages and query languages. Let $Q = (\mathcal{O}, \Sigma, q)$ be from $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ and D a database that uses only symbols from Σ . The *certain answers* to Q on D , denoted $\text{cert}(Q, D)$, are defined as in Section 1.2.2.

Example 5.1.5. Consider the OMQ $Q = (\mathcal{O}, \Sigma, q')$ based on an extension of the C2RPQ q from Example 5.1.3, where \mathcal{O} defines a single mother as an unmarried woman who has a child, using additional unary predicates `Female` and `SingleMother`, and q' has an additional conjunct requiring that y is a single mother, that is:

$$\begin{aligned}\mathcal{O} &= \forall x (\text{SingleMother}(x) \leftrightarrow \text{Female}(x) \wedge \text{Single}(x) \wedge \exists y \text{Child}(x, y)) \\ q'(x, y) &= q(x, y) \wedge \text{SingleMother}(y) \\ \Sigma &= \{\text{Child}, \text{Spouse}, \text{Female}, \text{Single}\}\end{aligned}$$

Note that \mathcal{O} is equivalent to a UNFO^{reg} (even plain UNFO) formula obtained by eliminating \leftrightarrow in the usual way. Let $D' = D \cup \{\text{Female}(\text{Blanca}), \text{Single}(\text{Blanca})\}$, where D is the database from Example 5.1.3. Then $\text{cert}(Q, D') = \{(\text{Nívea}, \text{Blanca}), (\text{Clara}, \text{Blanca})\}$, but $\text{cert}(Q, D) = \emptyset$.

OMQ evaluation in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ is a relevant problem since ontologies formulated in many logics used as ontology languages can be translated into an equivalent UNFO^{reg} sentence in polynomial time. In particular, this is the case for the basic description logics \mathcal{ALC} and \mathcal{ALCI} [BtCLW14] and for their extensions with transitive closure of roles [Baa91] and with regular expressions over roles [CDLN01]. For any of these logics \mathcal{L} , this of course also yields a polynomial time reduction of OMQ evaluation in $(\mathcal{L}, \text{UC2RPQ})$ to OMQ evaluation in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$. Even UNFO itself has occasionally been considered as an ontology language [BtCLW14].

For the rather common extension of the description logic \mathcal{ALC} with transitive roles [BHLS17], an equivalence preserving translation of ontologies into UNFO^{reg} sentences is not possible since UNFO^{reg} cannot enforce that a binary predicate is transitive. However, a transitive role can be simulated using the transitive closure of a binary predicate R (and never using R without transitive closure). In this way, one still obtains the desired polynomial time reduction of OMQ evaluation. The same reduction can be applied even to UNFO^{reg} extended with transitive relations. We use $\text{UNFO}_{\text{trans}}^{\text{reg}}$ to denote the extension of UNFO^{reg} where sentences take the form $\varphi_{\text{trans}} \wedge \varphi$ with φ_{trans} a conjunction of atoms of the form $\text{trans}(R)$, R a binary predicate symbol, and φ a UNFO^{reg} sentence. An atom $\text{trans}(R)$ is satisfied in a structure \mathfrak{A} if $R^{\mathfrak{A}}$ is transitive.

Evaluation of Boolean OMQs in $(\text{UNFO}_{\text{trans}}^{\text{reg}}, \text{UC2RPQ})$ reduces in polynomial time to satisfiability in $\text{UNFO}_{\text{trans}}^{\text{reg}}$ since $D \models (\mathcal{O}, \Sigma, q)$ iff $\varphi_D() \wedge \mathcal{O} \wedge \neg q()$ is unsatisfiable. The reduction can be extended to non-Boolean queries by simulating answer variables using fresh unary predicates. Because of this observation, we concentrate on deciding satisfiability rather than OMQ evaluation.

Lemma 5.1.1. *OMQ evaluation in $(\text{UNFO}_{\text{trans}}^{\text{reg}}, \text{UC2RPQ})$ reduces in polynomial time to satisfiability in UNFO^{reg} , and so does satisfiability in $\text{UNFO}_{\text{trans}}^{\text{reg}}$.*

Proof. We proceed in three steps: first, we reduce evaluation of *Boolean* OMQs in $(\text{UNFO}_{\text{trans}}^{\text{reg}}, \text{UC2RPQ})$ to satisfiability in $\text{UNFO}_{\text{trans}}^{\text{reg}}$; second, we reduce satisfiability in $\text{UNFO}_{\text{trans}}^{\text{reg}}$ to satisfiability in UNFO^{reg} ; third, we reduce evaluation of (general) OMQs to the Boolean case. For the sake of a convenient notation, we denote structures (and thus databases) as sets of facts.

(1) For the reduction from Boolean OMQ evaluation to satisfiability, let D be a database and $Q = (\mathcal{O}, \Sigma, q)$ an OMQ with \mathcal{O} a $\text{UNFO}_{\text{trans}}^{\text{reg}}$ formula and q a Boolean UC2RPQ. We show that

$$D \models Q \quad \text{iff} \quad \varphi_D \wedge \mathcal{O} \wedge \neg q \text{ is unsatisfiable,}$$

where φ_D is D viewed as a Boolean CQ in the obvious way.

The ‘if’ direction is immediate. To prove ‘only if’, assume $\mathfrak{A} \models \varphi_D() \wedge \mathcal{O} \wedge \neg q$ for some structure \mathfrak{A} . In particular, $\mathfrak{A} \models \varphi_D()$ is witnessed by a homomorphism h from D to \mathfrak{A} . If h is injective, then \mathfrak{A} is an extension of D and thus witnesses $D \not\models Q$ as desired. If h is not injective, we have to extend \mathfrak{A} iteratively, in each step taking an element a that has two distinct preimages b_1, b_2 under h , creating a fresh copy a' , duplicating all tuples in which a participates, and updating h such that it maps b_1 to a and b_2 to a' . After exhaustive application of this step, we obtain a structure \mathfrak{A}' that homomorphically embeds into \mathfrak{A} and is UN-bisimilar [tCS13] to \mathfrak{A} , plus an injective homomorphism h' from D to \mathfrak{A}' . Together with the assumption, this implies that $\mathfrak{A}' \models \mathcal{O} \wedge \neg q$ and \mathfrak{A}' extends D ; hence $D \not\models Q$ as desired.

(2) Let φ be UNFO_{trans}^{reg} sentence with transitivity atoms $\text{trans}(R_1), \dots, \text{trans}(R_n)$. Transform φ into a UNFO^{reg} sentence φ' by dropping the transitivity atoms and replacing each atom $R_i(x, y)$ with the (regular expression) atom $R_i^+(x, y)$. It is straightforward to show that φ is satisfiable if and only if φ' is.

(3) For the reduction from (general) OMQ evaluation to the Boolean case, let $Q = (\mathcal{O}, \Sigma, q(\mathbf{x}))$ be an OMQ in (UNFO_{trans}^{reg}, UC2RPQ) with $q(\mathbf{x}) = q_1(\mathbf{x}) \vee \dots \vee q_n(\mathbf{x})$ such that $q_i(\mathbf{x}) = \exists \mathbf{y}_i \varphi_i(\mathbf{x}, \mathbf{y}_i)$ and $\mathbf{x} = x_1, \dots, x_n$, D a database, and $\mathbf{a} \subseteq \text{dom}(D)$ with $\mathbf{a} = a_1, \dots, a_n$. We construct a new OMQ Q' and database D' by taking fresh unary predicates P_1, \dots, P_n and setting:

$$\begin{aligned} D' &= D \cup \{P_1(a_1), \dots, P_n(a_n)\} \\ Q' &= (\mathcal{O}, \Sigma, q'()), \quad \text{where} \\ q'() &= q'_1() \vee \dots \vee q'_n(), \quad \text{with} \\ q'_i() &= \exists \mathbf{y}_i \varphi_i(\mathbf{x}, \mathbf{y}_i) \wedge P_1(x_1) \wedge \dots \wedge P_n(x_n) \end{aligned}$$

It suffices to prove the following claim, which is nearly straightforward.

Claim. $\mathbf{a} \in \text{cert}(Q, D) \Leftrightarrow () \in \text{cert}(Q', D')$

Proof of Claim. We proceed via contraposition in both directions.

‘ \Rightarrow ’ Assume $() \notin \text{cert}(Q', D')$. Then there is a structure \mathfrak{A} that extends D' and is a model of \mathcal{O} with $() \notin \text{ans}(q', \mathfrak{A})$. Since \mathfrak{A} extends D' it also holds that $\mathbf{a} \notin \text{ans}(q, \mathfrak{A})$ (because otherwise the witnessing homomorphism would also witness $() \in \text{ans}(q', \mathfrak{A})$, a contradiction). Hence $\mathbf{a} \notin \text{cert}(Q, D)$.

‘ \Leftarrow ’ Suppose $\mathbf{a} \notin \text{cert}(Q, D)$. Then there is a structure \mathfrak{A} that extends D and is a model of \mathcal{O} with $\mathbf{a} \notin \text{ans}(q, \mathfrak{A})$. Consider the structure $\mathfrak{A}' = \mathfrak{A} \cup \{P_1(a_1), \dots, P_n(a_n)\}$, which extends \mathfrak{A} and is a model of \mathcal{O} (since the P_i were fresh). In addition, it holds that $() \notin \text{ans}(q', \mathfrak{A}')$ (because otherwise the witnessing homomorphism would also witness $\mathbf{a} \notin \text{ans}(q, \mathfrak{A})$, a contradiction). Hence $() \notin \text{cert}(Q', D')$. \square

Together with Theorem 5.3.2 it thus follows that UNFO can be extended with transitive relations without losing decidability or affecting the complexity of satisfiability and of OMQ evaluation. This is in contrast to the guarded fragment, where in both cases decidability can only be obtained by adopting additional syntactic restrictions. While for satisfiability it suffices to assume that transitive relations are only used in guard positions, even stronger restrictions are necessary for OMQ evaluation [Grä99, ST04, GPT13].

There are also other interesting reasoning problems that can be reduced to satisfiability in UNFO^{reg} . Here we consider OMQ containment, leaving out transitive roles for simplicity. Let $Q_1 = (\mathcal{O}, \Sigma_{\text{full}}, q_1)$ and $Q_2 = (\mathcal{O}, \Sigma_{\text{full}}, q_2)$ be OMQs from $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ with the same number of answer variables and where Σ_{full} is the *full* data signature, that is, the set of all predicate symbols. We say that Q_1 is contained in Q_2 and write $Q_1 \subseteq Q_2$ if for every database D , $\text{cert}(Q_1, D) \subseteq \text{cert}(Q_2, D)$. We observe that OMQ containment can also be reduced to satisfiability in polynomial time.

Lemma 5.1.2. *OMQ containment in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ reduces in polynomial time to satisfiability in UNFO^{reg} .*

Proof. It suffices to reduce containment between *Boolean* OMQs to satisfiability; the case of general OMQs can be reduced to the Boolean case by applying the construction described in the proof of Lemma 5.1.1, Step 3, to both input OMQs.

Let $Q_1 = (\mathcal{O}, \Sigma_{\text{full}}, q_1)$ and $Q_2 = (\mathcal{O}, \Sigma_{\text{full}}, q_2)$ be OMQs with \mathcal{O} a UNFO^{reg} sentence and q_1, q_2 Boolean UC2RPQs. Then the following holds:

$$Q_1 \subseteq Q_2 \quad \text{iff} \quad \mathcal{O} \wedge q_1 \wedge \neg q_2 \text{ is unsatisfiable.}$$

For the ‘if’ direction, assume $Q_1 \not\subseteq Q_2$. Then there is a database D with $\text{cert}(Q_1, D) \not\subseteq \text{cert}(Q_2, D)$, i.e., there is a structure \mathfrak{A} extending D such that $\mathfrak{A} \models \mathcal{O}$, $\mathfrak{A} \models q_1$, and $\mathfrak{A} \not\models q_2$. Hence $\mathcal{O} \wedge q_1 \wedge \neg q_2$ is satisfiable.

For the ‘only if’ direction, assume that $\mathcal{O} \wedge q_1 \wedge \neg q_2$ is satisfiable. Then $\mathfrak{A} \models \mathcal{O}$, $\mathfrak{A} \models q_1$, and $\mathfrak{A} \not\models q_2$ for some structure \mathfrak{A} . Then \mathfrak{A} contains some (finite) database D witnessing $Q_1 \not\subseteq Q_2$. \square

There are also versions of OMQ containment that admit different ontologies in the two involved OMQs and more restricted data signatures in place of Σ_{full} [BBP18, BHLW16, BLW12, BL16]. These are computationally harder and a polynomial time reduction to satisfiability cannot be expected. In fact, it follows from results in [BL16] that this more general form of OMQ containment is 2NEXPTIME-hard already when the ontologies are formulated in the description logic \mathcal{ALCZ} , a fragment of UNFO, and when the actual queries are CQs. Decidability remains an open problem. We remark that when the actual queries in OMQs are CQs, then OMQ containment under the full data signature can be reduced to query evaluation in a straightforward way, essentially by viewing the query from the left-hand OMQ as a database. In the presence of regular path queries, however, this does not seem to be easily possible.

5.2 Model-theoretic Characterization

We give a characterization of satisfiability in UNFO^{reg} that is tailored towards implementation by tree automata. In particular, we show that every satisfiable UNFO^{reg} formula φ has a model whose treewidth is bounded by the width of φ , introduce a representation of such models in terms of labeled trees, and characterize the satisfaction of C2RPQs in models represented in this way in terms of tree-shaped witnesses. To simplify the technical development, in this section and the subsequent one we disallow predicates of arity zero. Note that an atom $P()$ can be simulated by the formula $\exists x P(x)$, so this assumption is w.l.o.g. We work with UNFO^{reg} sentences that are in certain normal form.

5.2.1 Normal Form

We next introduce a normal form for UNFO^{reg} sentences, similar but not identical to the normal form used for UNFO in [tCS13]. For a set \mathcal{L} of UNFO^{reg} formulas with one free variable, a *C2RPQ extended with \mathcal{L} -formulas* is a C2RPQ in which all tests $\varphi(x)?$ in atoms $E(z_1, z_2)$ have been replaced with tests $\psi(x)?$, $\psi(x)$ a formula from \mathcal{L} . The set of *normal UNFO^{reg} formulas* is the smallest set of formulas such that

1. every connected C2RPQ with exactly one free variable, extended with normal UNFO^{reg} formulas, is a normal UNFO^{reg} formula;
2. if $\varphi(x)$ and $\psi(x)$ are normal UNFO^{reg} formulas, then $\neg\varphi(x)$, $\varphi(x) \vee \psi(x)$, and $\exists x \varphi(x)$ are normal UNFO^{reg} formulas.

Observe that Item 1 serves as an induction start since every connected C2RPQ without tests (and with one free variable) is a normal UNFO^{reg} formula. Note that normal formulas are closed under conjunction in the sense that the conjunction of normal formulas $\varphi_1(x)$ and $\varphi_2(x)$ is a C2RPQ extended with normal UNFO^{reg} formulas and thus a normal formula. Thus, unary disjunction could be eliminated, but for our purposes it is more convenient to keep it. We note in passing that using this normal form, it is easy to observe that UNFO^{reg} has the same expressive power as C2RPQs with exactly one free variable that admit both tests and negated tests.

The *width* of a normal UNFO^{reg} formula is the maximal number of variables in a C2RPQ that occurs in it (not counting the variables that occur in the C2RPQ only inside tests). The *atom width* is defined analogously, but referring to the number of atoms instead of the number of variables. In the context of normal UNFO^{reg} formulas, for brevity we speak of C2RPQs when meaning C2RPQs extended with normal UNFO^{reg} formulas. The *size* of a UNFO^{reg} formula is the number of symbols needed to write it, with variable symbols and predicate symbols being counted as a single symbol.

Lemma 5.2.1. *Every UNFO^{reg} sentence φ can be transformed into an equivalent normal UNFO^{reg} sentence φ' in single exponential time. Moreover, the width and the atom width of φ' are at most polynomial in the size of φ and the path expressions that occur in φ' are exactly those in φ .*

Proof. By Lemma 4.1 of [tCS13], we can convert any UNFO sentence φ in single exponential time into an equivalent UNFO sentence φ' generated by the following grammar:

$$\varphi(x) ::= \exists \mathbf{y} \psi(x, \mathbf{y}) \mid \neg\varphi(x) \mid \varphi(x) \vee \varphi(x)$$

where $\exists \mathbf{y} \psi(x, \mathbf{y})$ is a CQ that might contain equality atoms. The transformation steps are rather straightforward and also work for UNFO^{reg} with the only difference that $\exists \mathbf{y} \psi(x, \mathbf{y})$ is then a C2RPQ that might contain equality atoms. The transformation may cause an at most single exponential blowup in formula size and it satisfies the requirements regarding parameters formulated in Lemma 5.2.1.

We can easily eliminate equality atoms in C2RPQs by identifying variables; when a free variable is identified with a quantified variable, we use the name of the free variable.

It remains to make C2RPQs connected. Let $\exists \mathbf{y} \psi(x, \mathbf{y})$ be a C2RPQ subformula such that $\psi(x, \mathbf{y})$ has the connected components $\psi(x, \mathbf{y}_0), \psi(\mathbf{y}_1), \dots, \psi(\mathbf{y}_k)$, $k \geq 1$. We replace it with the conjunction of $\varphi_0 = \exists \mathbf{y}_0 \psi(x, \mathbf{y}_0)$ and $\varphi_1 = \exists \mathbf{y}_1 \psi(\mathbf{y}_1), \dots, \varphi_k = \exists \mathbf{y}_k \psi(\mathbf{y}_k)$, that is, with the C2RPQ $\varphi_0?(x), \varphi_1?(x), \dots, \varphi_k?(k)$. Note that this C2RPQ

and all C2RPQs inside the tests are connected. In fact, it can be verified that the resulting UNFO^{reg} sentence is normal according to our definition. \square

In the following sections, we replace atoms $E(z_1, z_2)$ in the C2RPQs that occur in a normal UNFO^{reg} formula with atoms of the form $\mathcal{A}(z_1, z_2)$ where \mathcal{A} is a *nondeterministic automaton on finite words (NFA)* over a suitable alphabet; we call such atoms *NFA atoms*. Formally, an NFA is a tuple $(Q, \Sigma, \Delta, q_0, F)$ where Q is a finite set of states, Σ a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ a transition relation, $q_0 \in Q$ an initial state and $F \subseteq Q$ a set of final states. When deciding the satisfiability of a UNFO^{reg} sentence φ_0 , we will generally take Σ to be $\{R, R^- \mid R \text{ a binary predicate in } \varphi_0\} \cup \{\varphi(x)? \mid \varphi(x)? \text{ a test in } \varphi_0\}$. Clearly, all path expressions in φ_0 are regular expressions over this alphabet. Since every regular expression can be converted into an equivalent NFA in polynomial time, we can thus w.l.o.g. assume the NFA-based presentation. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ be an NFA. Then we use $\mathcal{A}[F/F']$ to denote the NFA obtained from \mathcal{A} by replacing F with $F' \subseteq Q$ and $\mathcal{A}[q_0/q]$ for the NFA obtained from \mathcal{A} by replacing q_0 with $q \in Q$. For a structure \mathfrak{A} , an NFA \mathcal{A} , and $a, b \in A$, we write $\mathfrak{A} \models \mathcal{A}(a, b)$ if there are $a_1, \dots, a_n \in A$ and a word $w \in L(\mathcal{A})$ of length $n - 1$ such that $a = a_1$, $a_n = b$, $(a_i, a_{i+1}) \in R^{\mathfrak{A}}$ if the i th symbol in w is R , $(a_{i+1}, a_i) \in R^{\mathfrak{A}}$ if the i th symbol in w is R^- , and $a_i = a_{i+1}$ and $\mathfrak{A} \models \varphi(a_i)$ if the i th symbol in w is $\varphi(x)?$. This gives a semantics to NFA atoms. The *size* of a normal UNFO^{reg} formula with NFA atoms is defined in the same way as the size of a UNFO formula, where every NFA $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ contributes the cardinality of Q plus the cardinality of Δ plus the cardinality of F .

5.2.2 Tree-like Structures

We aim to show the tree-like model property for UNFO^{reg}. In order to achieve that we represent tree-like models via type-decorated trees.

We start with some preliminaries. A (*directed*) *tree* is a prefix-closed subset $T \subseteq (\mathbb{N} \setminus \{0\})^*$. A node $w \in T$ is a *successor* of $v \in T$ and v is a *predecessor* of w if $w = v \cdot i$ for some $i \in \mathbb{N}$. Moreover, w is a *neighbor* of v if it is a successor or predecessor of v . A *tree-like structure* is a pair (T, \mathbf{bag}) where T is a tree and \mathbf{bag} a function that assigns to every $w \in T$ a finite structure $\mathbf{bag}(w)$ such that

the set of nodes $\{w \in T \mid a \in \mathbf{dom}(w)\}$ is connected in T , for each $a \in \bigcup_{w \in T} \mathbf{dom}(w)$

where, here and in the remainder of the chapter, $\mathbf{dom}(w)$ is a shorthand for $\mathbf{dom}(\mathbf{bag}(w))$. The *width* of (T, \mathbf{bag}) is the maximum domain size of structures that occur in the range of \mathbf{bag} . Its *outdegree* is the outdegree of T . A tree-like structure (T, \mathbf{bag}) defines the associated structure $\mathfrak{A}_{(T, \mathbf{bag})}$ which is the (non-disjoint) union of all structures $\mathbf{bag}(w)$, $w \in T$. We use $\mathbf{dom}(T, \mathbf{bag})$ as a shorthand for $\mathbf{dom}(\mathfrak{A}_{(T, \mathbf{bag})})$. As witnessed by its representation (T, \mathbf{bag}) , the treewidth of the structure $\mathfrak{A}_{(T, \mathbf{bag})}$ is bounded by the maximum cardinality of $\mathbf{dom}(\mathbf{bag}(w))$, $w \in T$.

We will show that every satisfiable UNFO^{reg} sentence φ_0 is satisfiable in a tree-like structure whose width is bounded by the width of φ_0 . In UNFO, it suffices to consider structures of this form in which bags overlap in at most one element; this is not the case in UNFO^{reg}.

Let φ_0 be a normal UNFO^{reg} sentence. We use $\mathbf{sub}(\varphi_0)$ to denote the subformulas of φ_0 with at most one free variable, and where the free variable is renamed to x . Then $\mathbf{cl}(\varphi_0)$ denotes the smallest set of normal UNFO^{reg} formulas that contains $\mathbf{sub}(\varphi_0)$ and is closed under single negation. A *1-type* for φ_0 is a subset $t \subseteq \mathbf{cl}(\varphi_0)$ that satisfies the following conditions:

1. $\varphi \in t$ iff $\neg\varphi \notin t$ for all $\neg\varphi \in \text{cl}(\varphi_0)$;
2. $\varphi \vee \psi \in t$ iff $\varphi \in t$ or $\psi \in t$ for all $\varphi \vee \psi \in \text{cl}(\varphi_0)$.

We use $\text{TP}(\varphi_0)$ to denote the set of all 1-types for φ_0 .

A *type decorated tree-like structure* for φ_0 is a triple (T, bag, τ) with (T, bag) a tree-like structure such that only predicates from φ_0 occur in the range of bag and $\tau : \text{dom}(T, \text{bag}) \rightarrow \text{TP}(\varphi_0)$. Let (T, bag, τ) be such a structure, \mathcal{A} an NFA, and $a, b \in \text{dom}(T, \text{bag})$. We write $\mathfrak{A}_{(T, \text{bag}), \tau} \models \mathcal{A}(a, b)$ if $\mathfrak{A}_{(T, \text{bag})} \models \mathcal{A}(a, b)$ with the semantics of tests reinterpreted: instead of demanding that $\mathfrak{A} \models \varphi(a')$ for a test $\varphi_0(x)$? to hold at an element a' , we now require that $\varphi \in \tau(a')$. Let $\varphi(x) = \exists \mathbf{y} \psi(x, \mathbf{y})$ be a C2RPQ and $a \in \text{dom}(T, \text{bag})$. A *homomorphism from $\varphi(x)$ to (T, bag, τ)* is a function $h : \{x\} \cup \mathbf{y} \rightarrow \text{dom}(T, \text{bag})$ such that the following conditions are satisfied:

- $h(\mathbf{x}) \in R^{\mathfrak{A}_{(T, \text{bag})}}$ for each $R(\mathbf{x}) \in \varphi(x)$;
- $\mathfrak{A}_{(T, \text{bag}), \tau} \models \mathcal{A}(h(y), h(z))$ for each $\mathcal{A}(y, z) \in \varphi(x)$.

A type decorated tree-like structure (T, bag, τ) for φ_0 is *proper* if:

1. for all $\exists x \varphi(x) \in \text{cl}(\varphi_0)$, $\exists x \varphi(x) \in \tau(a)$ iff there is a $b \in \text{dom}(T, \text{bag})$ with $\varphi(x) \in \tau(b)$;
2. for all C2RPQs $\varphi(x) \in \text{cl}(\varphi_0)$, $\varphi(x) \in \tau(a)$ iff there is a homomorphism h from $\varphi(x)$ to (T, bag, τ) such that $h(x) = a$.

We are now ready to give a model-theoretic characterization of satisfiability for UNFO^{reg} .

5.2.3 Characterization of Satisfiability

We aim to characterise the satisfaction of C2RPQs in type-decorated trees via tree-shaped witnesses. The following lemma establishes proper type decorated tree-like structures for φ_0 as witnesses for the satisfiability of φ_0 . The proof of the ‘only if’ direction is via an unraveling procedure that constructs a type decorated tree-like structure in a top-down manner, introducing fresh bags to satisfy C2RPQs and to implement a step-by-step chase of paths that witness satisfaction of NFA atoms in C2RPQs.

Lemma 5.2.2. *A normal UNFO^{reg} sentence φ_0 of size n and width m is satisfiable iff there is a proper type decorated tree-like structure (T, bag, τ) for φ_0 of width at most m and outdegree at most $n^2 + n$ such that $\varphi_0 \in \tau(a)$ for some a .*

Proof. (\Leftarrow) Assume a proper type decorated tree-like structure (T, bag, τ) for φ_0 . It is easily verified by induction on the structure of formulas that, for all $\psi(x) \in \text{cl}(\varphi_0)$ and all $a \in \text{dom}(T, \text{bag})$, we have $\mathfrak{A}_{(T, \text{bag}), \tau} \models \psi(a)$ if, and only if, $\psi(x) \in \tau(a)$. Since $\varphi_0 \in \tau(a)$ for some a , we get $\mathfrak{A}_{(T, \text{bag}), \tau} \models \varphi_0$.

(\Rightarrow) Let $\mathfrak{A} \models \varphi_0$ for a UNFO^{reg} sentence φ_0 of size n . In order to construct a tree-like structure (T, bag) , we use an unraveling technique during which we maintain a mapping $h : \text{dom}(T, \text{bag}) \rightarrow A$ and an additional labeling $E(w)$ containing expressions of the form $\mathcal{A}(b, b')$, for each $w \in T$. Throughout the construction, we preserve as an invariant that h is a homomorphism and that for each $\mathcal{A}(b, b') \in E(w)$, we have $b, b' \in \text{dom}(w)$ and $\mathfrak{A} \models \mathcal{A}(h(b), h(b'))$.

Start with choosing elements $a_1, \dots, a_k \in A$ such that, for every formula of the form $\exists x \psi(x) \in \text{cl}(\varphi_0)$ with $\mathfrak{A} \models \exists x \psi(x)$, there is some $i \in \{1, \dots, k\}$ and $\mathfrak{A} \models \psi(a_i)$. Initialize (T, bag) by setting $T = \{\varepsilon, 1, \dots, k\}$, $\text{bag}(\varepsilon)$ to the empty structure, and $\text{bag}(i) = \mathfrak{A}|_{\{a_i\}}$, for every $i \in \{1, \dots, k\}$, where $\mathfrak{A}|_X$ denotes the restriction of \mathfrak{A} to domain X . Moreover, set $h(a_i) = a_i$ and $E(\varepsilon) = E(i) = \emptyset$, for all $i \in \{1, \dots, k\}$. Clearly, the invariants are satisfied.

Then, apply the following steps exhaustively and in a fair way:

- Choose a node $w \in T$, an element $a \in \text{dom}(w)$, and a C2RPQ $\psi(x) \in \text{cl}(\varphi_0)$ with $\mathfrak{A} \models \psi(h(a))$. There is a mapping $\beta : \text{var}(\psi) \rightarrow A$ such that $\beta(x) = h(a)$, and for each atom $R(\mathbf{z}) \in \psi$, we have $\beta(\mathbf{z}) \in R^{\mathfrak{A}}$, and for each $\mathcal{A}(z, z') \in \psi$, \mathcal{A} an NFA, we have $\mathfrak{A} \models \mathcal{A}(\beta(z), \beta(z'))$. Let $B = \{\beta(z) \mid z \in \text{var}(\psi)\}$, and create a successor v of w where the associated structure $\text{bag}(v)$ is obtained from $\mathfrak{A}|_B$ by replacing each $b \in B \setminus \{h(a)\}$ with a fresh b' , and $h(a)$ with a . Extend h by setting $h(b') = b$, for all introduced b' . Finally, set $E(w) = \{\mathcal{A}(\beta(z), \beta(z')) \mid \mathcal{A}(z, z') \in \psi\}$ where \bar{b} is b' for all $b \in B \setminus \{h(a)\}$ and $\bar{h}(a)$ is a .
- Choose a node $w \in T$ and a label $\mathcal{A}(b, b') \in E(w)$. By the invariant for $E(w)$, we know that $\mathfrak{A} \models \mathcal{A}(h(b), h(b'))$. Take a shortest sequence $a_1, \dots, a_n \in A$ and word $\hat{w} \in L(\mathcal{A})$ of length $n - 1$ such that $a_1 = h(b)$, $a_n = h(b')$, and $(a_i, a_{i+1}) \in R^{\mathfrak{A}}$ if the i th symbol in \hat{w} is R , $(a_{i+1}, a_i) \in R^{\mathfrak{A}}$ if the i th symbol in \hat{w} is R^- , and $a_i = a_{i+1}$ and $\mathfrak{A} \models \varphi(a_i)$ if the i th symbol in \hat{w} is $\varphi(x)$?. If $n \leq 2$, then, by construction, $\text{bag}(w) \models \mathcal{A}(b, b')$, so we can stop. If $n > 2$, let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ and q_0, \dots, q_n be a sequence of states such that $(q_i, a_{i+1}, q_{i+1}) \in \Delta$ for all i with $0 \leq i < n$ and $q_n \in F$. Define $B = \{a_1, a_2, a_n, a_{n-1}\}$ and $B' = B \setminus \{a_1, a_n\}$. Then create a successor v of w , and set $\text{bag}(v)$ to the structure obtained from $\mathfrak{A}|_B$ by replacing every $d \in B'$ with a fresh d' , a_1 with b , and a_n with b' . Finally, extend h by setting $h(d') = d$ for all $d \in B'$, and set $E(v)$ to the singleton set containing $\mathcal{A}[q_0/q_1, F/\{q_{n-1}\}](\bar{a}_2, \bar{a}_{n-1})$, where \bar{d} is d' , for all $d \in B'$, and $\bar{a}_1 = b$ and $\bar{a}_n = b'$.

By definition of the rules, the invariants regarding h and E are preserved.

Let (T^*, bag^*) and h^* be the tree-like structure and homomorphism, respectively, obtained in the limit of the unraveling. We define a type decoration τ by taking $\tau(a) = \{\psi(x) \in \text{cl}(\varphi_0) \mid \mathfrak{A} \models \psi(h^*(a))\}$, for all $a \in \text{dom}(T^*, \text{bag}^*)$. It is not difficult to verify that $\mathfrak{A}|_{(T^*, \text{bag}^*)} \models \psi(a)$ iff $\psi(x) \in \tau(a)$, for all $a \in \text{dom}(T^*, \text{bag}^*)$ and all $\psi(x) \in \text{cl}(\varphi_0)$. In particular, we have:

- if $\psi(x) \in \tau(a)$ is a C2RPQ, then $\mathfrak{A} \models \psi(h^*(a))$. Since steps 1 and 2 were applied exhaustively, we know that $\mathfrak{A}|_{(T^*, \text{bag}^*)} \models \psi(a)$.
- Let $\neg\psi(x) \in \tau(a)$ for a C2RPQ $\psi(x)$ and assume $\mathfrak{A}|_{(T^*, \text{bag}^*)} \models \psi(a)$. Since h^* is a homomorphism from $\mathfrak{A}|_{(T^*, \text{bag}^*)}$ to \mathfrak{A} , we also have $\mathfrak{A} \models \psi(h^*(a))$, a contradiction to the definition of τ .

Consequently, $(T^*, \text{bag}^*, \tau)$ is proper. Finally, observe that the size of the bag created is bounded by m in the first step and by 4 in the second step. For the outdegree, observe that a bag created in the second step has outdegree 1, while a bag created in the first step has outdegree at most $n^2 + n$. \square

As the next step, we take a closer look at Point 2 of properness, that is, we characterize carefully the existence of a homomorphism h from $\varphi(x)$ to (T, \mathbf{bag}, τ) such that $h(x) = a$ in a way that is tailored towards implementation by tree automata. This gives rise to the notion of a witness tree below. We start with introducing the notions of subdivisions and splittings which shall help us to take care of the fact that the homomorphic image of a query $q(\mathbf{x})$ may be spread over several bags of a tree-like structure, and in fact this might even be the case for a single NFA atom.

An *instantiated C2RPQ* is a C2RPQ in which all free variables have been replaced with constants. We write $\varphi(\mathbf{a})$ to indicate that the constants in the instantiated C2RPQ are exactly \mathbf{a} . When working with instantiated C2RPQs, we drop existential quantifiers, assuming that all variables are implicitly existentially quantified. For brevity, we often omit the word ‘instantiated’ and only speak of C2RPQs. We speak of *terms* to mean both variables and constants, and we denote terms with t .

Let $\varphi(\mathbf{a})$ be a connected C2RPQ, Δ be a domain, and $s \geq 1$. A (Δ, s) -*subdivision* of an atom $\mathcal{A}(t, t') \in \varphi(\mathbf{a})$ is a set of atoms

$$\mathcal{A}[F/\{q_1\}](t, b_1), \mathcal{A}[q_0/q_1, F/\{q_2\}](b_1, b_2), \dots, \mathcal{A}[q_0/q_{k-1}, F/\{q_k\}](b_{k-1}, b_k), \mathcal{A}[q_0/q_k](b_k, t')$$

where q_1, \dots, q_k are states of \mathcal{A} , $k \leq s$, and b_1, \dots, b_k are constants from Δ . A C2RPQ $\psi(\mathbf{a}')$ is a (Δ, s) -*subdivision* of $\varphi(\mathbf{a})$ if it is obtained from $\varphi(\mathbf{a})$ by replacing zero or more NFA atoms with (Δ, s) -subdivisions. Let $\psi(\mathbf{a}')$ be a (Δ, s) -subdivision of $\varphi(\mathbf{a})$. A *splitting* of $\psi(\mathbf{a}')$ is a sequence $\psi_0(\mathbf{a}_0), \dots, \psi_\ell(\mathbf{a}_\ell)$, $\ell \geq 0$, of C2RPQs that is a partition of $\psi(\mathbf{a}')$ (viewed as a set of atoms) where we also allow the special case that $\psi_0(\mathbf{a}_0)$ is empty (and thus $\psi_1(\mathbf{a}_1), \dots, \psi_\ell(\mathbf{a}_\ell)$ is the actual partition). We require that the following conditions are satisfied:

1. $\psi_1(\mathbf{a}_1), \dots, \psi_\ell(\mathbf{a}_\ell)$ are connected;
2. $\text{var}(\psi_i(\mathbf{a}_i)) \cap \text{var}(\psi_j(\mathbf{a}_j)) \subseteq \text{var}(\psi_0(\mathbf{a}_0))$ for $1 \leq i < j \leq \ell$;
3. each of $\psi_1(\mathbf{a}_1), \dots, \psi_\ell(\mathbf{a}_\ell)$ contains at most one atom from each subdivision of an atom in $\varphi(\mathbf{a})$.

Intuitively, the $\vartheta_0(\mathbf{a})$ component of a splitting is the part of $\psi(\mathbf{a}')$ that maps into a bag that we are currently focussing on while the other components are pushed to neighboring bags.

Example 5.2.1. Consider $q(a) = \{\mathcal{A}(a, y), T(a, z), Q(a, y, z)\}$ with $\mathcal{A} = \rightarrow \textcircled{0} \xrightarrow{R} \textcircled{1} \curvearrowright$. Let $\Delta = \{a, b, c\}$ and $\mathcal{A}_{ij} = \mathcal{A}[0/i, F/j]$. An example for a $(\Delta, 2)$ -subdivision of $\mathcal{A}(a, y)$ is $\{\mathcal{A}_{01}(a, b), \mathcal{A}_{11}(b, b), \mathcal{A}_{11}(b, y)\}$, which yields the following $(\Delta, 2)$ -subdivision of $\varphi(a)$: $\psi(a, b) = \{\mathcal{A}_{01}(a, b), \mathcal{A}_{11}(b, b), \mathcal{A}_{11}(b, y), T(a, z), Q(a, y, z)\}$. $\psi(a, b)$ admits a splitting into ψ_0, ψ_1 as follows: $\psi_0(a, b) = \{\mathcal{A}_{01}(a, b), \mathcal{A}_{11}(b, b), T(a, z)\}$ and $\psi_1(a, b) = \{\mathcal{A}_{11}(b, y), Q(y, z, a)\}$.

The *query closure* $\text{qcl}(\varphi_0, \Delta, s)$ is defined as the smallest set such that the following conditions are satisfied:

- if $\varphi(x) \in \text{cl}(\varphi_0)$ is a C2RPQ and $a \in \Delta$, then $\varphi(a) \in \text{qcl}(\varphi_0, \Delta, s)$;
- if $\varphi(\mathbf{a}) \in \text{qcl}(\varphi_0, \Delta, s)$, $\psi(\mathbf{a}')$ is a (Δ, s) -subdivision of $\varphi(\mathbf{a})$, $\psi_0(\mathbf{a}_0), \dots, \psi_\ell(\mathbf{a}_\ell)$ is a splitting of $\psi(\mathbf{a}')$, $1 \leq i \leq \ell$, and $\psi'_i(\mathbf{a}'_i)$ is obtained from $\psi_i(\mathbf{a}_i)$ by consistently replacing zero or more variables with constants from Δ , then $\psi'_i(\mathbf{a}'_i) \in \text{qcl}(\varphi_0, \Delta, s)$.

Lemma 5.2.3. *The cardinality of $\text{qcl}(\varphi_0, \Delta, s)$ is bounded by $p \cdot (a^2 d^m)^{m'}$, where p is the number of C2RPQs in φ_0 , a the maximal number of states in an NFA in φ_0 , d the cardinality of Δ , m the width of φ_0 , and m' the atom width of φ_0 .*

Proof. Each query in $\text{qcl}(\varphi_0, \Delta, s)$ can be obtained from a C2RPQ $\varphi(x) \in \text{cl}(\varphi_0)$ by first dropping atoms, then replacing NFA atoms $\mathcal{A}(t, t')$ with an atom $\mathcal{A}'(\hat{t}, \hat{t}')$ where \mathcal{A}' is obtained from \mathcal{A} by replacing the initial state and replacing the set of final states with a single state, \hat{t} is t or a fresh constant, and \hat{t}' is t' or a fresh constant, and finally replacing existentially quantified variables with constants from Δ ; to see this, it is important to consider Condition 3 of splittings and to note that the ψ_0 -component of splittings is not included in $\text{qcl}(\varphi_0, \Delta, s)$. Thus the number of atoms in each query in $\text{qcl}(\varphi_0, \Delta, s)$ is at most m' . Furthermore, each atom can take on $a^2 d^m$ variations by replacing NFA states as described and/or replacing variables with constants. Thus each of the p C2RPQs $\varphi(x) \in \text{cl}(\varphi_0)$ contributes at most $(a^2 d^m)^{m'}$ C2RPQs to $\text{qcl}(\varphi_0, \Delta, s)$. \square

We are almost ready to define witness trees. The following notion of a homomorphism is more local than the ones used so far as it only concerns a single bag rather than the entire tree-like structure. Let (T, bag, τ) be a type decorated tree-like structure, $w \in T$, \mathcal{A} an NFA, and $a, b \in \text{dom}(w)$. We write $\text{bag}(w), \tau \models \mathcal{A}(a, b)$ if $\text{bag}(w) \models \mathcal{A}(a, b)$ with the semantics of tests reinterpreted: instead of demanding $\text{bag}(w) \models \varphi(a')$ for a test $\varphi(x)$? to hold at an element a' , we now require that $\varphi(x) \in \tau(a')$. Let $\varphi(\mathbf{a})$ be a C2RPQ. A *homomorphism from $\varphi(\mathbf{a})$ to $\text{bag}(w)$ given τ* is a function $h : \mathbf{a} \cup \text{var}(\varphi) \rightarrow \text{dom}(w)$ such that the following conditions are satisfied:

- $h(a) = a$ for each $a \in \mathbf{a}$;
- $h(\mathbf{t}) \in R^{\text{bag}(w)}$ for each $R(\mathbf{t}) \in \varphi(\mathbf{a})$;
- $\text{bag}(w), \tau \models \mathcal{A}(h(t), h(t'))$ for each $\mathcal{A}(t, t') \in \varphi(\mathbf{a})$.

Let n be the size of φ_0 , $a \in \text{dom}(T, \text{bag})$, and $\varphi(x) \in \text{cl}(\varphi_0)$ a C2RPQ. A *witness tree for $\varphi(a)$ in (T, bag, τ)* is a finite labeled tree (W, σ) with $\sigma : W \rightarrow T \times \text{qcl}(\varphi_0, \text{dom}(T, \text{bag}), n^2)$ such that the root is labeled with $\sigma(\varepsilon) = (w, \varphi(a))$ for some $w \in T$ with $a \in \text{dom}(w)$ and the following conditions are satisfied for all $u \in W$:

- (*) if $\sigma(u) = (w, \psi(\mathbf{a}))$, then there is a $(\text{dom}(w), n^2)$ -subdivision $\vartheta'(\mathbf{a})$ of $\psi(\mathbf{a})$, a splitting $\vartheta_0(\mathbf{a}_0), \dots, \vartheta_\ell(\mathbf{a}_\ell)$ of $\vartheta'(\mathbf{a})$, a homomorphism h from $\vartheta_0(\mathbf{a}_0)$ to $\text{bag}(w)$ given τ , and successors u_1, \dots, u_ℓ of u such that $\sigma(u_i) = (w_i, \vartheta'_i(\mathbf{a}'_i))$ for $1 \leq i \leq \ell$, where each w_i is a neighbor of w in T with $\mathbf{a}'_i \subseteq \text{dom}(w_i)$ and $\vartheta'_i(\mathbf{a}'_i)$ is obtained from $\vartheta_i(\mathbf{a}_i)$ by replacing each variable x in the domain of h with the constant $h(x)$.

Informally, a witness tree decomposes a homomorphism h from $\varphi(x)$ to $\mathfrak{A}_{T, \text{bag}}$ into local ‘chunks’, each of which concerns only a single bag. In particular, the splitting $\vartheta_0(\mathbf{a}_0), \dots, \vartheta_k(\mathbf{a}_k)$ in (*) breaks the current C2RPQ down into components that are satisfied in different parts of the tree-like structure. We need to first subdivide since satisfaction of NFA atoms is witnessed by an entire path, and this path can pass through the current node several times. Fortunately, the number of points introduced in a subdivision can be bounded: we can w.l.o.g. choose a shortest path and such a path can pass through w at most once for each element in $\text{dom}(w)$ and each state of the automaton \mathcal{A} , thus we need at most n^2 points in subdivisions.

Lemma 5.2.4. *Let (T, bag, τ) be a type decorated tree-like structure, $\varphi(x) \in \text{cl}(\varphi_0)$ a C2RPQ, and $a \in \text{dom}(T, \text{bag})$. Then there is a homomorphism h from $\varphi(x)$ to (T, bag, τ) with $h(x) = a$ iff there is a witness tree for $\varphi(a)$ in (T, bag, τ) .*

Proof. (\Rightarrow) Let h be a homomorphism from $\varphi(x)$ to (T, bag, τ) with $h(x) = a$. We inductively construct a witness tree (W, σ) for $\varphi(a)$ in (T, bag, τ) . During the construction, we maintain the following invariants for all nodes $u \in W$ with $\sigma(u) = (w, \psi(\mathbf{a}))$:

- (i) for all $R(\mathbf{t}) \in \psi(\mathbf{a})$, we have $h(\mathbf{t}) \in R^{\mathfrak{A}(T, \text{bag})}$;
- (ii) for all $\mathcal{A}(t, t') \in \psi(\mathbf{a})$, we have $\mathfrak{A}(T, \text{bag}), \tau \models \mathcal{A}(h(t), h(t'))$.

We start the construction with setting $\sigma(\varepsilon) = (w, \varphi(a))$ for some $w \in T$ with $a \in \text{dom}(w)$, obviously satisfying (i) and (ii).

Then, apply the following step exhaustively. Let $u \in W$ be an unprocessed node in the witness tree constructed so far, and assume that $\sigma(u) = (w, \psi(\mathbf{a}))$. First, assign to each constant $a \in \text{dom}(T, \text{bag}) \setminus \text{dom}(w)$ the (uniquely defined) value $\kappa(a) \in \{-1, 1, \dots, m\}$, m the outdegree of T , such that $w \cdot \kappa(a)$ lies on the shortest path from w to the unique world where a appears for the first time in (T, bag) .

We use the mapping κ to assign atoms occurring in (a subdivision of) $\psi(\mathbf{a})$ to neighboring nodes of w , intuitively, to reflect where h maps different parts of $\psi(\mathbf{a})$. Formally, we use queries $\psi_{-1}(\mathbf{b}_{-1}), \dots, \psi_\ell(\mathbf{b}_\ell)$, initialized with \emptyset , where $\psi_i(\mathbf{b}_i)$ collects the parts of $\psi(\mathbf{a})$ which are sent to $w \cdot i$, for all i . We process all atoms in $\psi(\mathbf{a})$ as follows:

1. For each atom $R(\mathbf{t}) \in \psi(\mathbf{a})$, by invariant (i), we can fix a $v \in T$ with $h(\mathbf{t}) \in R^{\text{bag}(v)}$ which has minimal distance to w . We distinguish three cases:
 - (a) if $h(\mathbf{t}) \in R^{\text{bag}(w)}$, then add $R(\mathbf{t})$ to ψ_0 ;
 - (b) if $h(\mathbf{t}) \subseteq \text{dom}(w)$ and $h(\mathbf{t}) \notin R^{\text{bag}(w)}$, then add $R(\mathbf{t})$ to ψ_i where i is the (unique) number such that $w \cdot i$ is on the shortest path to v ;
 - (c) if $h(\mathbf{t}) \not\subseteq \text{dom}(w)$, add $R(\mathbf{t})$ to ψ_i where $i = \kappa(h(x))$ for some $x \in \mathbf{t}$ with $h(x) \notin \text{dom}(w)$. It is important to note that i is uniquely defined in this way. Indeed, assume two variables $x, y \in \mathbf{t}$ with $h(x), h(y) \notin \text{dom}(w)$ and $\kappa(h(x)) \neq \kappa(h(y))$. Then, one of $w \cdot \kappa(h(x))$ and $w \cdot \kappa(h(y))$ does not lie on the shortest path from w to v , say the latter. However, by the connectedness property of tree decompositions, we know that then $h(y)$ appears in $\text{dom}(w)$, a contradiction.
2. For an atom $\mathcal{A}(t, t') \in \psi(\mathbf{a})$, by invariant (ii), we can fix sequences a_1, \dots, a_n and s_0, \dots, s_n , and a word $\nu_1 \cdots \nu_{n-1} \in L(\mathcal{A})$ of minimal length such that $a_1 = h(t)$, $a_n = h(t')$, $s_0 = q_0$, $s_n \in F$, $(s_i, \nu_i, s_{i+1}) \in \Delta$, for all $i \in \{1, \dots, n-1\}$, $(a_i, a_{i+1}) \in R^{\mathfrak{A}(T, \text{bag}, \tau)}$ if $\nu_i = R$, $(a_{i+1}, a_i) \in R^{\mathfrak{A}(T, \text{bag}, \tau)}$ if $\nu_i = R^-$, and $\theta(x) \in \tau(a_i)$ and $a_{i+1} = a_i$ if $\nu_i = \theta(x)$?. Let I be the set of all $i \in \{1, \dots, n\}$ such that $a_i \in \text{dom}(w)$.

If $I = \emptyset$, then t, t' are variables with $\kappa(h(t)) = \kappa(h(t'))$. Add $\mathcal{A}(t, t')$ to $\psi_{\kappa(h(t))}$.

Otherwise, that is, $I \neq \emptyset$, let $i_1 < \dots < i_k$ be a linear order of the elements in I . If $a_1 \notin \text{dom}(w)$ and thus $1 \notin I$, then add $\mathcal{A}[F/\{s_{i_1+1}\}](t, a_{i_1})$ to $\psi_{\kappa(a_{i_1})}$, and, if $a_n \notin \text{dom}(w)$ and thus $n \notin I$, then add $\mathcal{A}[q_0/s_{i_k}](a_{i_k}, t')$ to $\psi_{\kappa(a_n)}$. Moreover, for each $j \in \{1, \dots, k\}$ such that ν_{i_j} is not a test $\theta(x)$? do:

- if $\nu_{i_j} = R$ and $(a_{i_j}, a_{i_{j+1}}) \in R^{\text{bag}(w)}$, then add $\mathcal{A}[q_0/s_{i_j}, F/\{s_{i_{j+1}}\}](a_{i_j}, a_{i_{j+1}})$ to ψ_0 ;
- if $\nu_{i_j} = R$ and $(a_{i_j}, a_{i_{j+1}}) \notin R^{\text{bag}(w)}$, then add $\mathcal{A}[q_0/s_{i_j}, F/\{s_{i_{j+1}}\}](a_{i_j}, a_{i_{j+1}})$ to ψ_i , where $i = \kappa(a_{i_{j+1}})$;
- if $\nu_{i_j} = R^-$ and $(a_{i_{j+1}}, a_{i_j}) \in R^{\text{bag}(w)}$, then add $\mathcal{A}[q_0/s_{i_j}, F/\{s_{i_{j+1}}\}](a_{i_j}, a_{i_{j+1}})$ to ψ_0 ;
- if $\nu_{i_j} = R^-$ and $(a_{i_{j+1}}, a_{i_j}) \notin R^{\text{bag}(w)}$, then add $\mathcal{A}[q_0/s_{i_j}, F/\{s_{i_{j+1}}\}](a_{i_j}, a_{i_{j+1}})$ to ψ_i , where $i = \kappa(a_{i_{j+1}})$.

It is crucial that in Step 2 the cardinality of I is bounded by m^2 , m the size of φ_0 . More precisely, in sequences a_1, \dots, a_n and s_0, \dots, s_n of minimal length, there are no $i < j$ such that $a_i = a_j$ and $s_i = s_j$, as otherwise we can obtain shorter sequences by dropping a_{i+1}, \dots, a_j and s_{i+1}, \dots, s_j . As both $\text{dom}(w)$ and the number of states in \mathcal{A} is bounded by m , the claimed bound follows. It should thus be clear that $\vartheta(\mathbf{a}) = \bigcup_i \psi_i(\mathbf{b}_i)$ is a subdivision of $\psi(\mathbf{a})$.

Note that the $\psi_i(\mathbf{b}_i)$ need not be connected. Define a splitting $\vartheta_0(\mathbf{a}_0), \dots, \vartheta_\ell(\mathbf{a}_\ell)$ of $\vartheta(\mathbf{a})$ by setting $\vartheta_0(\mathbf{a}_0) = \psi_0(\mathbf{b}_0)$, and including, for each $i \in \{-1, 1, \dots, m\}$, each connected component $\psi'(\mathbf{a}')$ of $\psi_i(\mathbf{b}_i)$ in the sequence. By construction, h is a homomorphism from $\vartheta_0(\mathbf{a}_0)$ to $\text{bag}(w)$ given τ . Finally, extend the witness tree by adding, for each $\vartheta_i(\mathbf{a}_i)$, $1 \leq i \leq \ell$, a successor u_i of u with $\sigma(u_i) = (w \cdot j, \vartheta'_i(\mathbf{a}'_i))$ where j is such that $\vartheta_i(\mathbf{a}_i) \subseteq \psi_j(\mathbf{b}_j)$ and $\vartheta'_i(\mathbf{a}'_i)$ is obtained from $\vartheta_i(\mathbf{a}_i)$ by replacing each variable x in the domain of h with $h(x)$. By construction, u satisfies $(*)$. Moreover, it is routine to verify that invariants (i) and (ii) are preserved.

It remains to argue that the described process results in a finite tree. Clearly, the constructed tree is finitely branching. For finite depth, consider first atoms of the form $R(\mathbf{t}) \in \varphi(a)$. In Step 1, these atoms are always ‘sent’ to a closest v such that $h(\mathbf{t}) \in R^{\text{bag}(v)}$ (Items 1b and 1c). This v is reached after finitely many steps. Consider now atoms of the form $\mathcal{A}(t, t')$. Assume some atom $\mathcal{A}(t_0, t'_0)$ is obtained in Step 2 applied to $\mathcal{A}(t, t')$ and that $\mathcal{A}(t_0, t'_0) \in \vartheta_i(\mathbf{a}_i)$. Let $\mathcal{A}(t'_0, t'_1)$ be the corresponding atom in $\vartheta'_i(\mathbf{a}'_i)$. Then, by the minimality condition, the witnessing sequences selected in Step 2 when applied to $\mathcal{A}(t'_0, t'_1)$ while processing $\vartheta'_i(\mathbf{a}'_i)$ is strictly shorter than the witnessing sequence for $\mathcal{A}(t, t')$. Thus, finite depth follows.

(\Leftarrow) Let (W, σ) be a witness tree for $\varphi(a)$ in (T, bag, τ) . We inductively construct a homomorphism h from $\varphi(x)$ to (T, bag, τ) such that $h(x) = a$.

Start with $h(x) = a$. Then apply the following rule exhaustively. Let $u \in W$ be a node in the witness tree with $\sigma(u) = (w, \psi(\mathbf{a}))$ such that all predecessor nodes have been processed. Let g be the homomorphism witnessing Condition $(*)$ for u , and define $h(z) = g(z)$ for all z in the domain of g but not in the domain of h .

It is a consequence of Condition 2 of splittings and $(*)$, that h is well-defined. We show that the result h of this process is a homomorphism from $\varphi(x)$ to (T, bag, τ) .

- Consider first atoms of the form $R(\mathbf{t})$. We prove by induction that, if $R(\mathbf{t}) \in \psi(\mathbf{a})$ for some $u \in W$ with $\sigma(u) = (w, \psi(\mathbf{a}))$, then $h(\mathbf{t}) \in R^{\mathfrak{A}(T, \text{bag})}$. The induction base is the case when $R(\mathbf{t})$ is put into $\vartheta_0(\mathbf{a})$ by $(*)$. By definition of h , we know $h(\mathbf{t}) \in R^{\text{bag}(w)}$, thus also $h(\mathbf{t}) \in R^{\mathfrak{A}(T, \text{bag})}$. In the induction step, let $R(\mathbf{t}) \in \psi(\mathbf{a})$, but in $(*)$ the atom $R(\mathbf{t})$ is put into $\vartheta_i(\mathbf{a}_i)$ for some $i > 0$. By the definition of, $R(\mathbf{t}') \in \vartheta'_i(\mathbf{a}'_i)$ where \mathbf{t}' is obtained from \mathbf{t} by instantiating some variables $z \in \mathbf{t}$ with $h(z)$. By induction, we know that $h(\mathbf{t}') \in R^{\mathfrak{A}(T, \text{bag})}$, thus also $h(\mathbf{t}) \in R^{\mathfrak{A}(T, \text{bag})}$.

- Consider now atoms of the form $\mathcal{A}(t, t')$. We prove by induction that, if $\mathcal{A}(t, t') \in \psi(\mathbf{a})$ for some $u \in W$ with $\sigma(u) = (w, \psi(\mathbf{a}))$, then $\mathfrak{A}_{(T, \text{bag})}, \tau \models \mathcal{A}(h(t), h(t'))$. The induction base is the case when $\mathcal{A}(t, t')$ is put into $\vartheta_0(\mathbf{a}_0)$ by (*). By (*) and the definition of h , we know that h is a homomorphism from $\mathcal{A}(t, t')$ to $\text{bag}(w)$, hence $\text{bag}(w), \tau \models \mathcal{A}(h(t), h(t'))$ and thus $\mathfrak{A}_{(T, \text{bag})}, \tau \models \mathcal{A}(h(t), h(t'))$. In the induction step, the atom $\mathcal{A}(t, t')$ is subdivided into atoms, say $\alpha_1, \dots, \alpha_k$. However, by definition of subdivisions and by (*), it is straightforward to prove that $\mathfrak{A}_{(T, \text{bag})}, \tau \models \mathcal{A}(h(t), h(t'))$ given that $\mathfrak{A}_{(T, \text{bag})}, \tau \models \alpha_i$ for all i , by induction hypothesis. □

5.3 Decidability and Complexity

We now reduce satisfiability of UNFO^{reg} sentences to the nonemptiness problem of two-way alternating tree automata. We start discussing the encoding of tree-like structures as an input to automata and then show the automata constructions.

Encoding of tree-like structures. Let φ_0 be a normal UNFO^{reg} sentence whose satisfiability we want to decide. By Lemma 5.2.2, this corresponds to deciding the existence of a proper type decorated tree-like structure for φ_0 (of certain dimensions) and thus our aim is to build a 2ATA \mathcal{A} such that $L(\mathcal{A}) \neq \emptyset$ if and only if there is such a structure. 2ATAs cannot run directly on tree-like structures because the labeling of the underlying trees is not finite: we have already shown that UNFO^{reg} does not have the finite model property and thus it might be necessary that infinitely many elements occur in the bags. We therefore use an appropriate encoding that ‘reuses’ element names so that we can make do with finitely many element names overall, similar to what has been done, for example, in [GW99, ABBV16].

Let R_1, \dots, R_ℓ be the predicate symbols that occur in φ_0 and let m be the width of φ_0 . Fix a finite set Δ with $2m$ elements and define Σ to be the set of all pairs (bag, τ) such that $\text{bag} = (A, R_1^{\text{bag}}, \dots, R_\ell^{\text{bag}})$ is a structure that satisfies $A \subseteq \Delta$ and $|A| \leq m$, and $\tau : A \rightarrow \text{TP}(\varphi_0)$ is a map that assigns a 1-type to every element in bag .

Let (T, L) be a Σ -labeled tree. For convenience, we use bag_w to refer to the first component of $L(w)$ and τ_w to refer to the second component, that is, $L(w) = (\text{bag}_w, \tau_w)$. Moreover, dom_w is shorthand for $\text{dom}(\text{bag}_w)$. For an element $d \in \Delta$, we say that $v, w \in T$ are *d-equivalent* if $d \in \text{dom}_u$ for all u on the unique shortest path from v to w . Informally, this means that d represents the same element in bag_v and in bag_w . In case that $d \in \text{dom}_w$, we use $[w]_d$ to denote the set of all v that are d -equivalent to w . We say that (T, L) is *type consistent* if, for all $d \in \Delta$ and all d -equivalent $v, w \in T$, $\tau_v(d) = \tau_w(d)$. Each type consistent (T, L) represents a type decorated tree-like structure (T, bag', τ') of width at most m as follows. The domain of $\mathfrak{A}_{(T, \text{bag}')}$ is the set of all equivalence classes $[w]_d$ with $w \in T$ and $d \in \text{dom}_w$. The function τ' maps each domain element $[w]_d$ to $\tau_w(d)$, which is well-defined since (T, L) is type consistent. Finally, for every $w \in T$, the structure $\text{bag}'(w) = (A(w), R_1^{\text{bag}'(w)}, \dots, R_\ell^{\text{bag}'(w)})$ is defined by:

$$\begin{aligned} A(w) &= \{[w]_d \mid d \in \text{dom}_w\}, \\ R_i^{\text{bag}'(w)} &= \{([w]_{d_1}, \dots, [w]_{d_j}) \mid (d_1, \dots, d_j) \in R_i^{\text{bag}_w}\} \quad \text{for } 1 \leq i \leq \ell. \end{aligned}$$

Conversely, for every type decorated tree-like structure (T, bag, τ) of width m , there is a Σ -labeled tree (T, L) that represents a type decorated tree-like structure (T, bag', τ')

such that there is an isomorphism π between $\mathfrak{A}_{(T, \mathbf{bag})}$ and $\mathfrak{A}_{(T, \mathbf{bag}'')}$ that satisfies $\tau(d) = \tau'(\pi(d))$, for all $d \in \text{dom}(T, \mathbf{bag})$. In fact, since Δ is of size $2m$, it is possible to select a mapping $\pi : \text{dom}(T, \mathbf{bag}) \rightarrow \Delta$ such that for each $w \in T \setminus \{\varepsilon\}$ and each $d \in \text{dom}(w) \setminus \text{dom}(w \cdot -1)$, we have $\pi(d) \notin \{\pi(e) \mid e \in \text{dom}(w \cdot -1)\}$. Define the Σ -labeled tree (T, L) by setting, for all $w \in T$, \mathbf{bag}_w to the image of $\mathbf{bag}(w)$ under π and τ_w to the map defined by $\tau_w(h(d)) = \tau(d)$, for all $d \in \text{dom}_w$. Clearly, π satisfies the desired properties.

The notion of a witness tree carries over straightforwardly from type decorated tree-like structures to type consistent Σ -labeled trees. In fact, one only needs to replace τ with τ_w in Condition (*). Then, there is a witness tree for $\varphi(a)$ in a type consistent (T, L) iff there is a witness tree for $\varphi(a)$ in the type decorated tree-like structure $(T, \mathbf{bag}'', \tau')$ represented by (T, L) . The notion of properness also carries over straightforwardly. For easier reference, we spell it out explicitly below, and also replace the homomorphisms from the original formulation by witness trees as suggested by Lemma 5.2.4. A type consistent Σ -labeled tree (T, L) is *proper* if for all $w \in T$ and $a \in \text{dom}_w$,

- 1'. for all $\exists x \varphi(x) \in \text{cl}(\varphi_0)$, $\exists x \varphi(x) \in \tau_w(a)$ iff there is a $v \in T$, $b \in \text{dom}_v$ with $\varphi(x) \in \tau_v(b)$;
- 2'. for all C2RPQs $\varphi(x) \in \text{cl}(\varphi_0)$, $\varphi(x) \in \tau_w(a)$ iff there is a witness tree for $\varphi(a)$ in (T, L) .

It is straightforward to verify that (T, L) is proper iff the type decorated tree-like structure $(T, \mathbf{bag}'', \tau')$ represented by (T, L) is proper. Thus, our aim is to build a 2ATA \mathcal{A} that accepts exactly the proper type consistent Σ -labeled trees (T, L) such that $\varphi_0 \in \tau_w(a)$ for some $w \in T$ and $a \in \text{dom}_w$.

Automata construction. Let n be the size of φ_0 , $k = n^2 + n$ the bound on the outdegree from Lemma 5.2.2, and assume from now on that the automata run over k -ary Σ -labeled trees. It is straightforward to construct a 2ATA \mathcal{A}_0 that accepts (T, L) iff it is type consistent and satisfies Condition 1' of properness and the condition that $\varphi_0 \in \tau_w(a)$ for some $w \in T$ and $a \in \text{dom}_w$. The number of states of the automaton is linear in the size of φ_0 ; details are omitted. We next show how to construct a 2ATA $\mathcal{A}_1 = (Q, \Sigma, q_0, \delta, F)$ that accepts a type consistent (T, L) iff Condition 2' is satisfied. The automaton uses the set of states

$$Q = \{q_0\} \cup \{\varphi(\mathbf{a}), \bar{\varphi}(\mathbf{a}) \mid \varphi(\mathbf{a}) \in \text{qcl}(\varphi_0, \Delta, n^2)\}.$$

where states of the form $\varphi(\mathbf{a})$ are used to verify the ‘only if’ part of Condition 2' while states of the form $\bar{\varphi}(\mathbf{a})$ are used to verify the contrapositive of the ‘if’ part, that is, whenever a C2RPQ $\varphi(x) \in \text{cl}(\varphi_0)$ is not in $\tau_w(a)$, then there is no witness tree for $\varphi(a)$ in (T, L) .

Starting from the initial state, \mathcal{A}_1 loops over all nodes and domain elements using the following transitions, for all $(\mathbf{bag}, \tau) \in \Sigma$:

$$\delta(q_0, (\mathbf{bag}, \tau)) = \bigwedge_{1 \leq i \leq k} (i, q_0) \wedge \bigwedge_{a \in \text{dom}(\mathbf{bag})} \left(\bigwedge_{\substack{\varphi(x) \in \tau(a), \\ \varphi(x) \text{ a C2RPQ}}} \varphi(a) \wedge \bigwedge_{\substack{\neg \varphi(x) \in \tau(a), \\ \varphi(x) \text{ a C2RPQ}}} \bar{\varphi}(a) \right)$$

We next give transitions for states of the form $\varphi(\mathbf{a}) \in \text{qcl}(\varphi_0, \Delta, n^2)$. Informally, if the automaton visits a node w in state $\varphi(\mathbf{a})$, then this is an obligation to show that there

is a witness tree whose root is labeled with $(w, \varphi(\mathbf{a}))$. In particular, the automaton has to demonstrate that there are suitable successors for the root of the witness tree, implementing Condition (*). For a more concise definition of the transitions, we first establish a suitable notation. Let $\varphi(\mathbf{a}), \vartheta_1(\mathbf{a}_1), \dots, \vartheta_\ell(\mathbf{a}_\ell) \in \text{qcl}(\varphi_0, \Delta, n^2)$ and $(\text{bag}, \tau) \in \Sigma$. We write $\varphi(\mathbf{a}) \rightarrow_{(\text{bag}, \tau)} \vartheta_1(\mathbf{a}_1), \dots, \vartheta_\ell(\mathbf{a}_\ell)$ if there is a (Δ, n^2) -subdivision $\vartheta(\mathbf{a}')$ of $\varphi(\mathbf{a})$, a splitting $\vartheta'_0(\mathbf{a}'_0), \dots, \vartheta'_\ell(\mathbf{a}'_\ell)$ of $\vartheta(\mathbf{a}')$, a homomorphism h from $\vartheta'_0(\mathbf{a}'_0)$ to bag given τ , and $\vartheta_i(\mathbf{a}_i)$ is obtained from $\vartheta'_i(\mathbf{a}'_i)$ by replacing each variable x in the domain of h with the constant $h(x)$; please note that this is an essential part of Condition (*). Then, we include for each $\varphi(\mathbf{a}) \in \text{qcl}(\varphi_0, \Delta, n^2)$ and each $(\text{bag}, \tau) \in \Sigma$ the transition

$$\delta(\varphi(\mathbf{a}), (\text{bag}, \tau)) = \bigvee_{\varphi(\mathbf{a}) \rightarrow_{(\text{bag}, \tau)} \vartheta_1(\mathbf{a}_1), \dots, \vartheta_\ell(\mathbf{a}_\ell)} \bigwedge_{1 \leq i \leq \ell} \bigvee_{j \in [k] \setminus \{0\}} (j, \vartheta_i(\mathbf{a}_i))$$

if $\mathbf{a} \subseteq \text{dom}(\text{bag})$ and set $\delta(\varphi(\mathbf{a}), (\text{bag}, \tau)) = \text{false}$ otherwise. States of the form $\bar{\varphi}(\mathbf{a})$ are treated dually, that is, using the transitions

$$\delta(\bar{\varphi}(\mathbf{a}), (\text{bag}, \tau)) = \bigwedge_{\varphi(\mathbf{a}) \rightarrow_{(\text{bag}, \tau)} \vartheta_1(\mathbf{a}_1), \dots, \vartheta_\ell(\mathbf{a}_\ell)} \bigvee_{1 \leq i \leq \ell} \bigwedge_{j \in [k] \setminus \{0\}} (j, \bar{\vartheta}_i(\mathbf{a}_i))$$

if $\mathbf{a} \subseteq \text{dom}(\text{bag})$ and setting $\delta(\bar{\varphi}(\mathbf{a}), (\text{bag}, \tau)) = \text{true}$ otherwise.

To ensure that the witness trees constructed by the states of the form $\varphi(\mathbf{a})$ are finite, we use the parity condition $F = G_1, G_2$ with $G_1 = \text{qcl}(\varphi_0, \Delta, n^2)$ and $G_2 = Q$. From an accepting run of \mathcal{A}_1 on an input tree (T, L) , one can extract the witness trees that are required to show that the ‘only if’ direction of Condition 2' is satisfied. Moreover, the run demonstrates that the witness trees forbidden by the ‘if’ direction do not exist. We thus obtain the following.

Lemma 5.3.1. *The UNFO^{reg} sentence φ_0 is satisfiable iff $L(\mathcal{A}_0) \cap L(\mathcal{A}_1)$ is not empty.*

Putting together Lemmas 5.2.1, 5.2.3, and 5.3.1, it follows that satisfiability in UNFO^{reg} is in 2EXPTIME. The corresponding lower bound is inherited from UNFO [tCS13].

Theorem 5.3.2. *In UNFO^{reg}, satisfiability is 2EXPTIME-complete.*

Proof. The lower bound follows from that for UNFO [tCS13]. For the upper bound, let φ be a UNFO^{reg} sentence of size n . By Lemma 5.2.1, we can transform φ into an equivalent normal UNFO^{reg} sentence φ_0 whose width m and atom width m' are polynomial in n and whose size is single exponential in n . Note that the number of 1-types for φ_0 is double exponential in n and that, by the bounds stated in Lemma 5.2.1 and by Lemma 5.2.3, the cardinality of $\text{qcl}(\varphi_0, \Delta, n^2)$ is single exponential in n .

We then build \mathcal{A}_0 and \mathcal{A}_1 for φ_0 as described above. The number of states of \mathcal{A}_0 is exponential in n and, by the bound on $\text{qcl}(\varphi_0, \Delta, n^2)$ stated above, the same is true for the number of states of \mathcal{A}_1 . The alphabet Σ is of cardinality double exponential in n . The transition functions of \mathcal{A}_0 and \mathcal{A}_1 can be computed in time double exponential in n . Constructing the intersection 2ATA does not increase the number of states. In summary, the final 2ATA \mathcal{A} can be constructed in time double exponential in n and has single exponentially many states in n . The number of sets in the parity condition is a constant. Consequently, nonemptiness of \mathcal{A} can be decided in time double exponential in φ . \square

5.4 OMQ Evaluation and Containment

We study the computational complexity of OMQ evaluation and OMQ containment in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$. Recall that the complexity of OMQ evaluation can be measured in different ways. In *combined complexity*, both the OMQ and the database on which it is evaluated are considered to be an input. In *data complexity*, the OMQ is fixed and the database is the only input. We first state our main result regarding the combined complexity of OMQ evaluation and the complexity of OMQ containment.

Theorem 5.4.1. *In $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$,*

1. *OMQ evaluation is 2EXPTIME-complete in combined complexity and*
2. *OMQ containment is 2EXPTIME-complete.*

The upper bounds in Theorem 5.4.1 are a consequence of Lemmas 5.1.1 and 5.1.2 and Theorem 5.3.2. The lower bounds hold already when predicates are at most binary. For Point 1 this follows from the fact that OMQ evaluation is 2EXPTIME-hard even for OMQs from the class $(\mathcal{ALCT}, \text{CQ})$ where the ontology is formulated in the description logic \mathcal{ALCT} , a fragment of UNFO with only unary and binary predicates, and the actual query is a CQ [Lut08]. The same is true for Point 2 since in $(\mathcal{ALCT}, \text{CQ})$, OMQ evaluation can be reduced in polynomial time to OMQ containment in a straightforward way.

5.4.1 Data Complexity

We next study the data complexity of $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$. A coNP lower bound is again inherited from (rather small) fragments of $(\text{UNFO}^{\text{reg}}, \text{C2RPQ})$ [KL07, CDL⁺13]. We give a coNP upper bound, thus establishing the following.

Theorem 5.4.2. *OMQ evaluation in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ is coNP -complete in data complexity.*

Instead of directly considering OMQ evaluation, we work with a problem that we call database satisfiability. A database D is *satisfiable* with an UNFO^{reg} sentence φ if there is a model of φ that extends D . Let φ be an UNFO^{reg} sentence and Σ a set of predicate symbols. The *database satisfiability problem associated with φ and Σ* is to decide, given a Σ -database D , whether D is satisfiable with φ . Note that OMQ evaluation can be reduced in polynomial time to Boolean OMQ evaluation as in the proof of Lemma 5.1.1. Moreover, for a Boolean OMQ $Q = (\mathcal{O}, \Sigma, q)$ and a Σ -database D , $D \models Q$ iff D is satisfiable with $\mathcal{O} \wedge \neg q$. Consequently, a coNP upper bound for OMQ evaluation in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ can be proved by establishing an NP upper bound for database satisfiability in UNFO^{reg} .

Let φ_0 be an UNFO^{reg} formula and Σ a set of predicate symbols. We may assume w.l.o.g. that φ_0 is normal and that every symbol from Σ occurs in φ_0 . Subdivisions and splittings, defined as in Section 5.2, shall again play an important role. However, instead of subdividing an atom $\mathcal{A}(t, t')$ into at most n^2 many atoms, we use at most *two* intermediary points. Informally, this splits a witnessing path for $\mathcal{A}(t, t')$ into three parts: the first part is from t to the first element from D that appears on the path, the third subdivision atom represents the part from the last element from D that appears on the path to t' , and the second atom represents the remaining middle part of the path.

We use $\text{ecl}(\varphi_0)$ to denote the union of $\text{cl}(\varphi_0)$ and $\text{qcl}(\varphi_0)$, closed under single negation, where $\text{qcl}(\varphi_0)$ is $\text{qcl}(\varphi_0, \{x\}, 2)$ extended with the set of all $\mathcal{A}[q_0/s, F/\{s'\}](x, x)$ such that \mathcal{A} is an NFA that occurs in φ_0 and s, s' are states in \mathcal{A} . An *extended 1-type* for φ_0 is a subset $t \subseteq \text{ecl}(\varphi_0)$ such that t satisfies the conditions for being a 1-type from Section 5.2. We denote with $\text{eTP}(\varphi_0)$ the set of all extended 1-types for φ_0 .

Let D be a Σ -database. A *type decoration* for D is a mapping $\tau : \text{dom}(D) \rightarrow \text{eTP}(\varphi_0)$. We write $D, \tau \models \mathcal{A}(a, b)$ if $D \models \mathcal{A}(a, b)$ with the semantics of tests reinterpreted: instead of demanding $D \models \varphi(a')$ for a test $\varphi(x)?$ to hold at an element a' , we now require that $\varphi(x) \in \tau(a')$. Let $\varphi(\mathbf{a})$ be an (instantiated) C2RPQ. A *homomorphism from $\varphi(\mathbf{a})$ to D given τ* is a function $h : \mathbf{a} \cup \text{var}(\varphi) \rightarrow \text{dom}(D)$ such that the following conditions are satisfied: $h(\mathbf{a}) = \mathbf{a}$, $h(\mathbf{t}) \in R^D$ for each $R(\mathbf{t}) \in \varphi(\mathbf{a})$, and for each $\mathcal{A}(t, t') \in \varphi(\mathbf{a})$, there are $a_1, \dots, a_n \in \text{dom}(D)$ and states s_0, \dots, s_n from \mathcal{A} , and a word $\nu_1 \cdots \nu_{n-1}$ from the alphabet of \mathcal{A} such that

- (a) $a_1 = h(t)$, $a_n = h(t')$, $s_0 = q_0$, and $s_n \in F$,
- (b) $(a_i, a_{i+1}) \in R^D$ if $\nu_i = R$, $(a_{i+1}, a_i) \in R^D$ if $\nu_i = R^-$, and $\theta(x) \in \tau(a_i)$ and $a_{i+1} = a_i$ if $\nu_i = \theta(x)?$, for $1 \leq i < n$, and
- (c) $(s, \nu_i, s_{i+1}) \in \Delta$ for some s with $\mathcal{A}[q_0/s_i, F/\{s\}](x, x) \in \tau(a_{i+1})$, for $0 \leq i < n$.

Note that Condition (c) admits the spontaneous change from state s_i to state s at a_{i+1} , without reading any of the ν_j symbols, when the atom $\mathcal{A}[q_0/s_i, F/\{s\}](x, x)$ is contained in $\tau(a_{i+1})$, asserting that we can indeed get from s_i to s starting at a_{i+1} and cycling back there while reading some unknown subword.

A type decoration τ is called *proper* if for all $a \in \text{dom}(D)$, the following hold:

1. $\bigwedge_{\psi(x) \in \tau(a)} \psi(a)$ is satisfiable;
2. $\exists x \varphi(x) \in \tau(a)$ iff $\exists x \varphi(x) \in \tau(b)$, for all $a, b \in \text{dom}(D)$ and all $\exists x \varphi(x) \in \text{cl}(\varphi_0)$;
3. if $\neg \psi(x) \in \tau(a)$ for some $\psi(x) \in \text{qcl}(\varphi_0)$, then for each $(\text{dom}(D), 2)$ -subdivision $\vartheta(\mathbf{a})$ of $\psi(a)$ and each splitting $\vartheta_0(\mathbf{a}_0), \vartheta_1(a_1), \dots, \vartheta_\ell(a_\ell)$ of $\vartheta(\mathbf{a})$ such that there is a homomorphism h from $\vartheta_0(\mathbf{a}_0)$ to D given τ , there is an $i \in \{1, \dots, k\}$ such that $\neg \vartheta_i(x) \in \tau(a_i)$.

Our NP procedure for database satisfiability is, given a Σ -database D , to guess a type decoration τ for D and to then verify in deterministic polynomial time that D is proper. Note that the size of a type decoration is $\mathcal{O}(c \cdot |D|)$ for some constant c . The satisfiability checks in Point 1 of properness concern sentences whose size is independent of D , thus they need only constant time. Point 2 can be checked in time quadratic in the size of D . For Point 3, note that there are only polynomially many $(\text{dom}(D), 2)$ -subdivisions and splittings (in the size of D). To check the existence of the required homomorphism h , we can go through all candidates, directly verifying the homomorphism condition for relational atoms and proceedings as follows for NFA atoms: first extend D by exhaustively adding ‘implied facts’ of the form $\mathcal{A}(a, b)$, also taking into account assertions of the form $\mathcal{A}[q_0/s_i, F/\{s\}](x, x)$ that occur in τ -labels, as in Condition (c) above, and then treat NFA atoms like relational atoms. The following lemma finishes the proof of Theorem 5.4.2.

Lemma 5.4.3. *D is satisfiable with φ_0 iff D has a proper type decoration τ such that $\varphi_0 \in \tau(a_0)$ for some $a_0 \in \text{dom}(D)$.*

Proof. The ‘only if’ direction is rather straightforward. Let \mathfrak{A} be a model of D and φ_0 . Then we can define, for every $a \in \text{dom}(D)$,

$$\tau(a) = \{\varphi(x) \in \text{ecl}(\varphi_0) \mid \mathfrak{A} \models \varphi(a)\}.$$

We aim to show that τ is a proper type decoration of D and that $\varphi_0 \in \tau(a_0)$, for some $a_0 \in D$. The latter is clear as, by assumption, $\mathfrak{A} \models \varphi_0$. We verify Point 1 to 3 of properness. Points 1 and 2 are clear as $\tau(a)$ is read off from a model of $\tau(a)$. Assume to the contrary of what we aim to show that Point 3 is violated. Then there is an $a \in \text{dom}(D)$, a $\neg\psi(x) \in \tau(a)$ for some $\psi(x) \in \text{qcl}(\varphi_0)$, a subdivision $\vartheta(\mathbf{a})$ of $\psi(a)$, a splitting $\vartheta_0(\mathbf{a}_0), \vartheta_1(a_1), \dots, \vartheta_k(k_k)$ of $\vartheta(\mathbf{a})$ and a homomorphism h from $\vartheta_0(\mathbf{a}_0)$ to D given τ , such that $\neg\vartheta_i(x) \notin \tau(a_i)$, for all $1 \leq i \leq k$. By definition of τ , we get that $\mathfrak{A} \models \vartheta_i(a_i)$, for all i . It can be verified that this implies $\mathfrak{A} \models \psi(a)$, in contradiction to $\neg\psi(x) \in \tau(a)$. Thus, τ is proper.

Now for the ‘if’ direction. Assume that D has a proper type decoration τ . Then we find, for each $a \in \text{dom}(D)$, a model \mathfrak{A}_a of the formula in Point 1 of the definition of properness. We assume w.l.o.g. that the domains of \mathfrak{A}_a and \mathfrak{A}_b are disjoint when $a \neq b$ and that each \mathfrak{A}_a shares with D only the constant a . Let \mathfrak{A} be obtained by taking the union of D and all models \mathfrak{A}_a . Clearly, \mathfrak{A} is a model of D . It thus remains to show that it is also a model of φ_0 . We start with an auxiliary claim.

Claim. For all $a \in \text{dom}(D)$, $b \in \text{dom}(\mathfrak{A}_a)$, and for all $\varphi(x) \in \text{ecl}(\varphi_0)$ with a free variable x , we have $\mathfrak{A}_a \models \varphi(b)$ iff $\mathfrak{A} \models \varphi(b)$.

Proof of the Claim. The proof is by induction on the structure of $\varphi(x)$. Since φ_0 is normal, there are three cases: negation $\neg\varphi(x)$, unary disjunction $\varphi(x) \vee \psi(x)$, and C2RPQs $\varphi(x)$. Negation and unary disjunction are immediate; we consider only C2RPQs.

Let $\varphi(x) \in \text{ecl}(\varphi_0)$ be a C2RPQ. For (\Rightarrow) , assume that $\mathfrak{A}_a \models \varphi(b)$, that is, there is a mapping $h : \text{var}(\varphi) \rightarrow \text{dom}(\mathfrak{A}_a)$ such that $h(x) = b$ and

- $h(\mathbf{x}) \in R^{\mathfrak{A}_a}$ for all $R(\mathbf{x}) \in \varphi(x)$, and
- $\mathfrak{A}_a \models \mathcal{A}(h(z), h(z'))$ for all $\mathcal{A}(z, z') \in \varphi(x)$.

By definition of \mathfrak{A} and the induction hypothesis applied to tests in $\mathcal{A}(z, z')$, we also have

- $h(\mathbf{x}) \in R^{\mathfrak{A}}$ for all $R(\mathbf{x}) \in \varphi(x)$, and
- $\mathfrak{A} \models \mathcal{A}(h(z), h(z'))$ for all $\mathcal{A}(z, z') \in \varphi(x)$.

Thus, we obtain $\mathfrak{A} \models \varphi(b)$.

For (\Leftarrow) , assume that $\mathfrak{A} \models \varphi(b)$, that is, there is a mapping $h : \text{var}(\varphi) \rightarrow \text{dom}(\mathfrak{A})$ such that $h(x) = b$ and

- $h(\mathbf{x}) \in R^{\mathfrak{A}}$ for all $R(\mathbf{x}) \in \varphi(x)$, and
- $\mathfrak{A} \models \mathcal{A}(h(z), h(z'))$ for all $\mathcal{A}(z, z') \in \varphi(x)$.

Fix $a \in \text{dom}(D)$ such that $b \in \text{dom}(\mathfrak{A}_a)$, and let $U = \{a\} \cup (\text{dom}(\mathfrak{A}) \setminus \text{dom}(\mathfrak{A}_a))$. We first define a query $\psi_0(x)$, which intuitively contains those parts of φ that are mapped to U by h (that is, outside \mathfrak{A}_a); the free variable x ‘represents’ the domain element a . Formally, we process $\varphi(x)$ as follows:

1. for all $R(\mathbf{x}) \in \varphi(x)$ with $h(\mathbf{x}) \not\subseteq \text{dom}(\mathfrak{A}_a)$, we have, by construction of \mathfrak{A} , that $h(\mathbf{x}) \subseteq U$. In this case, we include $R(\mathbf{x}')$ in $\psi_0(x)$, where \mathbf{x}' is obtained from \mathbf{x} by renaming every $y \in \mathbf{x}$ satisfying $h(y) = a$ with x .
2. for all $\mathcal{A}(z, z') \in \varphi(x)$, there are sequences $a_1, \dots, a_n, s_0, \dots, s_n$, and a word $\nu_1 \cdots \nu_{n-1} \in L(\mathcal{A})$ such that $a_1 = h(z), a_n = h(z'), s_0 = q_0, s_n \in F, (s_i, \nu_i, s_{i+1}) \in \Delta$, for all $i \in \{1, \dots, n-1\}$, $(a_i, a_{i+1}) \in R^{\mathfrak{A}}$ if $\nu_i = R$, $(a_{i+1}, a_i) \in R^{\mathfrak{A}}$ if $\nu_i = R^-$, and $\theta(x) \in \tau(a_i)$ and $a_{i+1} = a_i$ if $\nu_i = \theta(x)$?

For all i, j with $1 \leq i \leq j \leq n$ such that a_i, \dots, a_j is a subsequence of a_1, \dots, a_n maximal with $a_k \in U$ for all $i \leq k \leq j$, we distinguish four cases:

- if $i = 1$ and $j = n$, then add $\mathcal{A}[F/\{s_n\}](z, z')$ to $\psi_0(x)$,
- if $i = 1$ and $j \neq n$, then add $\mathcal{A}[F/\{s_{j-1}\}](z, x)$ to $\psi_0(x)$,
- if $i \neq 1$ and $j = n$, then add $\mathcal{A}[q_0/s_{i-1}, F/\{s_n\}](x, z')$ to $\psi_0(x)$,
- if $i \neq 1$ and $j \neq n$, then add $\mathcal{A}[q_0/s_{i-1}, F/\{s_{j-1}\}](x, x)$ to ψ_0 .

Note that $a_j = a$ in the first, $a_i = a$, in the second, and $a_i = a_j = a$ in the last case.

Let $\psi_1(x), \dots, \psi_k(x)$ be all connected components of $\psi_0(x)$ with x considered as ‘constant’. It is not hard to see that $\psi_i(x) \in \text{qcl}(\varphi_0)$, for every $i \in \{1, \dots, k\}$.

Assume first that $\mathfrak{A}_a \models \psi_i(a)$, for all $i \in \{1, \dots, k\}$. In this case, we can modify h and the witnessing sequences to ‘live’ completely in \mathfrak{A}_a , and thus obtain $\mathfrak{A}_a \models \varphi(b)$. Assume now that $\mathfrak{A}_a \not\models \psi_i(a)$, for some $i \in \{1, \dots, k\}$. By Point 1, we have $\neg\psi_i(x) \in \tau(a)$. We now derive a contradiction to Point 3 using the mapping h from above. For doing so, we assume that $\text{dom}(D) = \{b_1, \dots, b_\ell\}$ and define queries $\vartheta_0(\mathbf{a}_0), \vartheta'_1(b_1), \dots, \vartheta'_\ell(b_\ell)$, where intuitively $\vartheta_0(\mathbf{a}_0)$ contains those atoms from $\psi_i(x)$ which are mapped to $\text{dom}(D)$ by h while $\vartheta'_i(b_i)$ contains all atoms which are mapped to \mathfrak{A}_{b_i} by h . Formally, we proceed as follows:

- for all $R(\mathbf{x}) \in \psi_i(x)$, we either have $h(\mathbf{x}) \in R^D$ or $h(\mathbf{x}) \in R^{\mathfrak{A}_{b_j}}$, for some j . In the former case, add $R(\mathbf{x})$ to $\vartheta_0(\mathbf{a}_0)$; in the latter case, add $R(\mathbf{x})$ to $\vartheta'_j(b_j)$, where \mathbf{x}' is obtained from \mathbf{x} by replacing every $y \in \mathbf{x}$ satisfying $h(y) = b_j$ with b_j ;
- Let $\mathcal{A}^*(z, z')$ be an atom that was added in Step 2 above, and let a_i, \dots, a_j and s_{i-1}, \dots, s_j be the sub-sequences corresponding to this atom which were assumed there. Depending on where the sequence a_i, \dots, a_j lies with respect to $\text{dom}(D)$, we distinguish five cases.
 - If $\{a_i, \dots, a_j\}$ is disjoint from $\text{dom}(D)$, then there is a k such that $\{a_i, \dots, a_j\} \subseteq \text{dom}(\mathfrak{A}_{b_k})$. Add $\mathcal{A}^*(z, z')$ to $\vartheta'_k(b_k)$ in this case.
 - If $\{a_i, \dots, a_j\}$ is not disjoint from $\text{dom}(D)$ and $a_i, a_j \notin \text{dom}(D)$, then let l, u be the unique numbers with $i \leq l \leq u \leq j$ such that $a_l, a_u \in \text{dom}(D)$ but $a_k \notin \text{dom}(D)$ for all $i \leq k < l$ and all $u < k \leq j$. Moreover, fix k, k' such that $a_l \in \text{dom}(\mathfrak{A}_{b_k})$ and $a_j \in \text{dom}(\mathfrak{A}_{b_{k'}})$. Note that $a_l = b_k$ and $a_u = b_{k'}$ in this case. Then add $\mathcal{A}^*[F/\{s_{l-1}\}](z, a_l)$ to $\vartheta'_k(b_k)$, $\mathcal{A}^*[q_0/s_{l-1}, F/\{s_{u-1}\}](a_l, a_u)$ to $\vartheta_0(\mathbf{a}_0)$, and $\mathcal{A}^*[q_0/s_{u-1}](a_u, z')$ to $\vartheta'_{k'}(b_{k'})$.
 - If $a_i \in \text{dom}(D)$ and $a_j \notin \text{dom}(D)$, then let u be the unique number with $i \leq u \leq j$ such that $a_u \in \text{dom}(D)$, but $a_k \notin \text{dom}(D)$ for all $u < k \leq j$, and fix k such that $a_j \in \text{dom}(\mathfrak{A}_{b_k})$. Note that $a_u = b_k$ in this case. Then add $\mathcal{A}^*[F/\{s_{u-1}\}](a_i, a_u)$ to $\vartheta_0(\mathbf{a}_0)$ and $\mathcal{A}^*[q_0/s_{u-1}](a_u, z')$ to $\vartheta'_k(b_k)$.

- If $a_j \in \text{dom}(D)$ and $a_i \notin \text{dom}(D)$, then let l be the unique number with $i \leq l \leq j$ such that $a_l \in \text{dom}(D)$, but $a_k \notin \text{dom}(D)$ for all $i \leq k < l$, and fix k such that $a_i \in \text{dom}(\mathfrak{A}_{b_k})$. Note that $a_l = b_k$ in this case. Then add $\mathcal{A}^*[F/\{s_{l-1}\}](z, a_l)$ to $\vartheta'_k(a_l)$ and $\mathcal{A}^*[q_0/s_{l-1}](a_l, a_i)$ to $\vartheta_0(\mathbf{a}_0)$.
- If $a_i, a_j \in \text{dom}(D)$, then add $\mathcal{A}^*(a_i, a_j)$ to $\vartheta_0(\mathbf{a}_0)$

Now obtain a sequence $\vartheta_1(a_1), \dots, \vartheta_k(a_k)$ by replacing each $\vartheta'_i(b_i)$ with its connected components. It should be clear that $\vartheta_0(\mathbf{a}_0), \vartheta_1(a_1), \dots, \vartheta_k(a_k)$ is a splitting of a subdivision of $\psi_i(a)$. Moreover, h is a homomorphism from $\vartheta_0(\mathbf{a}_0)$ to D given τ . However, by construction of $\vartheta_i(a_i)$, it should be clear that $\mathfrak{A}_{a_i} \models \vartheta_i(a_i)$, for all i . Thus, $\vartheta_i(x) \in \tau(a_i)$ for all i , a contradiction to Condition 3. This finishes the proof of the Claim.

Based on the previous claim, we can establish by structural induction that, for all sentences $\varphi \in \text{ecl}(\varphi_0)$, we have:

$$\varphi \in \tau(a) \text{ for all } a \in \text{dom}(D) \quad \text{iff} \quad \mathfrak{A} \models \varphi.$$

Note that Condition 2 is used to prove the induction base. As $\varphi_0 \in \tau(a_0)$ for some $a_0 \in \text{dom}(D)$, this yields the desired $\mathfrak{A} \models \varphi_0$. \square

5.5 Model Checking

We show that model checking in UNFO^{reg} is complete for $\text{P}^{\text{NP}[O(\log^2 n)]}$, the class of problems that can be solved in polynomial time given access to an NP oracle, but with only $O(\log^2 n)$ many oracle calls admitted. It thus has the same complexity as model checking in UNFO. Formally, the model checking problem for UNFO^{reg} is as follows: given a finite structure \mathfrak{A} and a UNFO^{reg} sentence φ , does $\mathfrak{A} \models \varphi$ hold? Without tests in path expressions, UNFO^{reg} model checking can easily be reduced to model checking in UNFO: simply extend the input structure by exhaustively adding ‘implied facts’ of the form $\mathcal{A}(a, b)$ and then replace every \mathcal{A} with a fresh binary relation symbol in both φ and \mathfrak{A} , obtaining an instance of UNFO model checking. With tests, this does not work. We would need multiple calls to UNFO model checking, essentially one call for every subformula inside a test in the input formula, but this brings us outside of $\text{P}^{\text{NP}[O(\log^2 n)]}$. We thus resort to expanding the $\text{P}^{\text{NP}[O(\log^2 n)]}$ upper bound proof from [tCS13], which is by reduction to a $\text{P}^{\text{NP}[O(\log^2 n)]}$ -complete circuit value problem.

Theorem 5.5.1. *The UNFO^{reg} model checking problem is $\text{P}^{\text{NP}[O(\log^2 n)]}$ -complete.*

Proof. The lower bound follows from that for UNFO [tCS13]. For the upper bound, we give a polynomial-time reduction of the model checking problem for UNFO^{reg} to a restricted version of the problem ‘Tree Block Satisfaction’, which was shown to be $\text{P}^{\text{NP}[O(\log^2 n)]}$ -complete in [Sch03]. This version, called TB(SAT) in [tCS13], is defined as follows.

A *TB-tree* of width $k \geq 1$ is a tree consisting of *blocks*, where each block is a kind of Boolean circuit that has k outputs and, for each of its n children, has k inputs,¹ see Figure 5.1. The i -th output of a block is determined by the values of its inputs in a way defined by an existentially quantified Boolean formula (\exists QBF) χ_i of the form

$$\chi_i = \exists \mathbf{b}_1 c_1 \dots \mathbf{b}_m c_m \mathbf{d} (c_1 = \text{input}_{i_1}(\mathbf{b}_1) \wedge \dots \wedge c_m = \text{input}_{i_m}(\mathbf{b}_m) \wedge \psi), \quad \text{where}$$

¹In the general case [Sch03], a block may have additional inputs, which do not connect to children and are thus inputs of the TB-tree. Our version does not allow this; i.e., we restrict ourselves to TB-trees without inputs.

- $i_1, \dots, i_m \leq n$;
- each \mathbf{b}_j is a tuple of $\log k$ variables, encoding a number $\#\mathbf{b}_j \leq k$;
- $\text{input}_{i_j}(\mathbf{b}_j)$ represents the value of the $\#\mathbf{b}_j$ -th output bit of the i_j -th child block (e.g., if $\#\mathbf{b}_j = 5$, then $\text{input}_2(\mathbf{b}_j) = y_5^{(2)}$ in Figure 5.1);
- ψ is a Boolean formula using any of the existentially quantified variables.

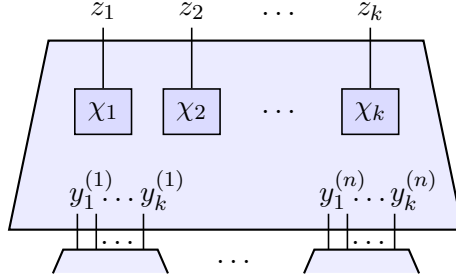


Figure 5.1: A block in a TB-tree of width k with n children.

TB(SAT) is the following problem: given a TB-tree of width k , does the first output bit of the root block have value 1? For the reduction, we show how to construct, for a given UNFO^{reg} sentence φ and structure \mathfrak{A} , a TB-tree $T_{\mathfrak{A},\varphi}$ such that

$$T_{\mathfrak{A},\varphi} \text{ is a yes-instance of TB(SAT) iff } \mathfrak{A} \models \varphi. \quad (*)$$

Let $|\text{dom}(\mathfrak{A})| = k$ and assume a linear order on the elements of \mathfrak{A} from 1 to k . For a given $a \in \text{dom}(\mathfrak{A})$, we use $\#a$ to denote the position of a in this order. We construct $T_{\mathfrak{A},\varphi}$ of width k via induction on the structure of φ . The construction satisfies the following invariant: For every subformula $\psi(x)$ of φ with *at most one* free variable, and every $a \in \text{dom}(\mathfrak{A})$ with $\#a = i$,

$$\text{The } i\text{-th output gate of } T_{\mathfrak{A},\psi(x)} \text{ is true iff } \mathfrak{A} \models \psi(a). \quad (**)$$

It is easy to see that $(**)$ implies $(*)$: just set $\psi(x) = \varphi$ and $i = 1$. Furthermore, $(**)$ is readily checked in every step of the induction.

The shape of each $T_{\mathfrak{A},\psi(x)}$ will roughly reflect that of the syntactic tree of $\psi(x)$. When we construct the \exists QBFs χ_i of each block, we will use \mathbf{b} to denote a vector of $\log k$ variables and $\#\mathbf{b} = i$ as a shorthand for the Boolean formula expressing that \mathbf{b} represents the binary encoding of i .

Let $\psi(x)$ be a subformula of φ .

Case 1: $\psi(x) = \neg\vartheta(x)$. Construct $T_{\mathfrak{A},\psi(x)}$ from $T_{\mathfrak{A},\vartheta(x)}$ by adding a new root block whose i -th output is defined by the formula that negates the i -th input of the single child:

$$\chi_i := \exists \mathbf{b} c (c = \text{input}_1(\mathbf{b}) \wedge \#\mathbf{b} = i \wedge c = 0)$$

Case 2: $\psi(x)$ is built from atomic formulas and UNFO^{reg} formulas in one free variable using conjunction, disjunction, and existential quantification. Let y_1, \dots, y_n be the variables in $\psi(x)$ that are quantified on the ‘top level’, i.e., outside the scope of any test in $\psi(x)$. Let $\vartheta_1(z_1), \dots, \vartheta_m(z_m)$ the maximal subformulas in one free variable, where $z_i \in \{x, y_1, \dots, y_n\}$ for all $i \leq m$. We can assume w.l.o.g. that the z_i are distinct

and coincide with y_1, \dots, y_m : if they are not, then one can always introduce additional quantified variables and equality atoms. We construct $T_{\mathfrak{A}, \psi(x)}$ from the $T_{\mathfrak{A}, \vartheta_i(x)}$ by adding a new root block whose children are the roots of the $T_{\mathfrak{A}, \vartheta_i(x)}$, and whose i -th output is defined by the formula

$$\chi_i := \exists \mathbf{b}_1 c_1 \dots \mathbf{b}_m c_m \mathbf{b}_{m+1} \dots \mathbf{b}_n \left(\bigwedge_{j \leq m} c_j = \text{input}_j(\mathbf{b}_j) \right) \wedge \chi_{\mathfrak{A}},$$

where the \mathbf{b}_j, c_j are used to refer to the values of the subformulas $\vartheta_j(y_j)$, the $\mathbf{b}_{m+1}, \dots, \mathbf{b}_n$ correspond to the additional y_j and $\chi_{\mathfrak{A}}$ is obtained from $\psi(x)$ as follows:

- Every subformula $\vartheta_j(y_j)$ is replaced by c_j .
- Every equational atom $x = y_j$ is replaced by $\#\mathbf{b}_j = i$ and $y_j = y_\ell$ by $\#\mathbf{b}_j = \#\mathbf{b}_\ell$.
- Every relational atom $R(y_{j_1}, \dots, y_{j_\ell})$ is replaced by a Boolean formula enumerating all tuples in $R^{\mathfrak{A}}$:

$$\bigvee_{(a_1, \dots, a_\ell) \in R^{\mathfrak{A}}} \left(\#\mathbf{b}_{j_1} = \#a_1 \wedge \dots \wedge \#\mathbf{b}_{j_\ell} = \#a_\ell \right)$$

- Every regular atom $\mathcal{A}(y_j, y_h)$ is replaced with the Boolean formula

$$\bigvee_{a, b \in \text{dom}(\mathfrak{A})} \left(\#\mathbf{b}_j = \#a \wedge \#\mathbf{b}_h = \#b \wedge \alpha_{\mathcal{A}, a, b} \right),$$

where $\alpha_{\mathcal{A}, a, b}$ is an \exists QBF that evaluates to true iff there is a path from element a to b in \mathfrak{A} that is accepted by \mathcal{A} . After bound renaming, the quantifiers from $\alpha_{\mathcal{A}, a, b}$ can be moved forward such that χ_i becomes a well-formed \exists QBF. To encode \mathcal{A} 's behavior in $\alpha_{\mathcal{A}, a, b}$, we assume that $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$, where

- Q with $|Q| = t$ is the set of states;
- $\Sigma = \{R, R^- \mid R \text{ a binary predicate in } \varphi\} \cup \{\vartheta(x)? \mid \vartheta(x)? \text{ a test in } \varphi\}$ is the input alphabet;
- q_0 is the initial state;
- $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation;
- $F \subseteq Q$ is the set of accepting states.

For every $p, q \in Q$, denote with $\mathcal{A}[p, q]$ the NFA obtained from \mathcal{A} by setting p to be the initial state and q to be the only accepting state.

The encoding uses Boolean variables $x_{p, q, a, b}^\ell$ with $\ell \leq t \cdot k$, $p, q \in Q$, and $a, b \in \text{dom}(\mathfrak{A})$. The truth value of the variable $x_{p, q, a, b}^\ell$ indicates whether there is a path of length ℓ from a to b in \mathfrak{A} that is accepted by $\mathcal{A}[p, q]$. It is clear that, whenever there is some path from a to b accepted by $\mathcal{A}[p, q]$, then there is always a path of length $\leq t \cdot k$ because one can always omit loops between two positions in a path that agree in state and element visited. Therefore the above restriction $\ell \leq t \cdot k$ suffices for a correct modeling of \mathcal{A} 's behavior, and the number of variables needed is polynomial.

The formula $\alpha_{\mathcal{A},a,b}$ enforces the correct truth values of these variables via induction on ℓ and requires that some $x_{q_0,q_f,a,b}^\ell$ with $q_f \in F$ be true:

$$\begin{aligned} \alpha_{\mathcal{A},a,b} = & \exists \mathbf{b}'_{1,1} c'_{1,1} \dots \mathbf{b}'_{m,k} c'_{m,k} \quad \exists_{\substack{h=0,\dots,|Q| \\ p,q \in Q \\ a,b \in \text{dom}(\mathfrak{A})}} x_{p,q,a,b}^h \quad \exists_{\substack{h=0,\dots,|Q| \\ \ell=1,\dots,k \\ q \in Q \\ a,b \in \text{dom}(\mathfrak{A})}} \bar{x}_{p,q,a,b}^{h,\ell} \\ & \left(\bigwedge_{j \leq m, \ell \leq k} c'_{j,\ell} = \text{input}_j(\mathbf{b}'_{j,\ell}) \wedge \# \mathbf{b}'_{j,\ell} = \ell \right) \\ & \wedge \beta_{\mathcal{A}} \wedge \gamma_{\mathcal{A}} \wedge \delta_{\mathcal{A}} \wedge \bigvee_{\ell \leq t \cdot k} \bigvee_{q_f \in F} x_{q_0,q_f,a,b}^\ell, \end{aligned}$$

where the $\mathbf{b}'_{j,\ell}, c'_{j,\ell}$ make the values of all $\vartheta_j(y_j)$ in all elements of the structure accessible for evaluating tests in regular atoms, the $x_{\cdot,\cdot,\cdot}^h$ and $\bar{x}_{\cdot,\cdot,\cdot}^{h,\ell}$ will be used below to evaluate regular atoms, and the conjuncts $\beta_{\mathcal{A}}, \gamma_{\mathcal{A}}, \delta_{\mathcal{A}}$ have the purpose to set the $x_{\cdot,\cdot,\cdot}^\ell$ for $\ell = 0$, $\ell = 1$, and $\ell \geq 2$, respectively. They are defined as follows.

$$\begin{aligned} \beta_{\mathcal{A}} = & \bigwedge_{\substack{q \in Q \\ a \in \text{dom}(\mathfrak{A})}} x_{q,q,a,a}^0 \wedge \bigwedge_{\substack{p,q \in Q \\ a,b \in \text{dom}(\mathfrak{A}) \\ p \neq q \text{ or } a \neq b}} \neg x_{p,q,a,b}^0 \\ \gamma_{\mathcal{A}} = & \bigwedge_{\substack{(p,R,q) \in Q \\ (a,b) \in R^{\mathfrak{A}}}} x_{p,q,a,b}^1 \wedge \bigwedge_{\substack{(p,R^-,q) \in Q \\ (a,b) \in R^{\mathfrak{A}}}} x_{p,q,b,a}^1 \wedge \bigwedge_{\substack{(p,\vartheta_j^?,q) \in Q \\ a \in \text{dom}(\mathfrak{A})}} (x_{p,q,a,a}^1 \leftrightarrow c'_{\#a}) \wedge \gamma'_{\mathcal{A}}, \end{aligned}$$

where $\gamma'_{\mathcal{A}}$ is the conjunction of $\neg x_{\cdot,\cdot,\cdot}^1$ for all $x_{\cdot,\cdot,\cdot}^1$ that do not occur in the preceding conjuncts of $\gamma_{\mathcal{A}}$. Finally,

$$\delta_{\mathcal{A}} = \bigwedge_{\substack{p,q \in Q \\ a,b \in \text{dom}(\mathfrak{A}) \\ 1 \leq \ell < t \cdot k}} \left(x_{p,q,a,b}^{\ell+1} \leftrightarrow \bigvee_{\substack{r \in Q \\ c \in \text{dom}(\mathfrak{A})}} (x_{p,r,a,c}^1 \wedge x_{r,q,c,b}^{\ell+1}) \right)$$

Case 2 also covers the case where $\psi(x)$ has no subformula with one free variable. It is easy to check that the invariant (**) holds and that $T_{\mathfrak{A},\varphi}$ can be constructed in polynomial time. \square

5.6 Concluding Remarks

We have proved that OMQ evaluation in $(\text{UNFO}^{\text{reg}}, \text{UC2RPQ})$ is decidable, 2EXPTIME -complete in combined complexity, and CONP -complete in data complexity, and that OMQ containment and satisfiability are also 2EXPTIME -complete. There are several interesting topics for future work. First, in contrast to UNFO, UNFO^{reg} does not have the finite model property and thus it would be interesting to study OMQ evaluation over finite models as well as finite satisfiability. Second, there are various natural directions for further increasing the expressive power. For example, one could allow any UNFO^{reg} formula with two free variables as a base case in regular path expressions instead of only atomic formulas. Such a logic would be strictly more expressive than propositional dynamic logic (PDL) with converse and intersection [GLL09] and it would

push the expressive power of UNFO^{reg} into the direction of regular queries, which have recently been proposed as an extension of C2RPQs [RRV17]. Another natural extension would be to replace C2RPQs with linear Datalog to remove the asymmetry between binary relations and relations of higher arity in UNFO^{reg} . Additional relevant extensions could arise from the aim to capture additional description logics. From this perspective, it would for example be natural to extend UNFO^{reg} with constants, with fixed points, and with so-called role inclusions [BHLS17]. Since functional relations and similar forms of counting play an important role in description logics, we remark that it is implicit in [tCS13] that satisfiability (and thus OMQ evaluation) is undecidable in UNFO extended with two functional relations. Finally, it would be interesting to investigate the complexity of OMQ containment in $(\text{UNFO}^{\text{reg}}, \text{C2RPQ})$ without the restriction to a single ontology and to the full data signature. For (UNFO, CQ) , a 2-NEXPTIME upper bound can be proved by a slight adaptation of the technique in [BL16], also using (a slightly refined version of) the translation from (UNFO, CQ) to monadic disjunctive Datalog from [BtCLW14]. However, accommodating C2RPQs in this approach seems nontrivial.

Relation-Changing Modal Logics as Fragments of Hybrid Logics

Modal logics [BvB07, BdRV01] were originally conceived as logics of necessary and possible truths. They are now viewed, more broadly, as logics that explore a wide range of modalities, or modes of truth: epistemic (“it is known that”), doxastic (“it is believed that”), deontic (“it ought to be the case that”), or temporal (“it has been the case that”), among others. From a model-theoretic perspective, the field evolved into a discipline that deals with languages interpreted on various kinds of relational structures or graphs. Nowadays, modal logics are actively used in areas as diverse as software verification, artificial intelligence, semantics and pragmatics of natural language, law, philosophy, etc.

As we just mentioned, from an abstract point of view, modal logics can be seen as formal languages to navigate and explore properties of a given relational structure. If we are interested, on the other hand, in describing how a given relational structure evolves (through time or through the application of certain operations) then classical modal languages seem a priori to fall short of the mark. Of course, it is a priori possible to statically model the whole space of possible transformations as a graph, and use modal languages at that level, but this soon becomes unwieldy. It is also possible to represent model update conditions as parts of the model itself, and interact with them by means of the classical modal language. This is the approach taken by Gabbay’s in his study of reactive Kripke frames [Gab08, Gab13]. Alternatively, it is possible to use standard relational models, and use modal languages with *dynamic modalities* encoding the desired changes.

There exist several dynamic modal logics that fit in this last approach. A clear example are the dynamic operators introduced in dynamic epistemic logics (see, e.g., [vDvdHK07]). These operators are used to model changes in the epistemic state of an agent by removing edges from the graph that represents the information states the agent considers possible. A less obvious example is given by hybrid logics [AtC07, BS95] equipped with the down arrow operator \downarrow which is used to ‘rebind’ names for states to the current point of evaluation. Finally, a classical example is Sabotage Logic introduced by van Benthem in [vB05]. The sabotage operator deletes individual edges in a graph and was introduced to solve the *sabotage game*. This game is played on a graph by two players, *Runner* and *Blocker*. Runner can move on the graph from node to accessible node, starting from a designated point, and with the goal of reaching a

given final point. Blocker, on the other hand, can delete one edge from the graph every time it is his turn. Runner wins if he manages to move from the origin to the final point, while Blocker wins otherwise. Van Benthem turns the sabotage game into a modal logic, where the (global) sabotage operator $\langle \text{gsb} \rangle$ models the moves of Blocker, and is interpreted on a graph \mathcal{M} at a point w as:

$$\mathcal{M}, w \models \langle \text{gsb} \rangle \varphi \text{ iff there is a pair } (u, v) \text{ of } \mathcal{M} \text{ such that } \mathcal{M}_{(u,v)}^-, w \models \varphi$$

where $\mathcal{M}_{(u,v)}^-$ is identical to \mathcal{M} except that the edge (u, v) has been removed. The moves of Runner, on the other hand, can be modeled using the standard \diamond operator of classical modal logics.

More recently, Sabotage Logic was proposed as a formalism for reasoning about formal learning theory [GKVQ09]. Learning can be seen as a game with two players, *Teacher* and *Learner*, where Learner changes his information state through a step-by-step process. The process is successful if he eventually reaches an information state describing the real state of affairs. The information that Teacher provides can be interpreted as feedback about Learner's conjectures about the current state of affairs, allowing him to discard inconsistent hypotheses. It should be clear that from this game-theoretical perspective, the interaction between Teacher and Learner can be modeled using Sabotage Logic.

The dynamic approach seems appealing and very flexible: it is easy to come up with situations that nicely fit and extend the examples we just mentioned. Discovering alternative routes for Runner in van Benthem's sabotage game, or possible shortcuts that Learner can take in learning theory can be modeled by adding new edges to the graph. Swapping an edge can be used to represent other scenarios such as changing the direction of a route, or allowing Learner to return to a previous information state. All these primitives can also be turned into a modal logic in the same way as Sabotage Logic, in order to get a formal language for reasoning about the games.

Motivated by scenarios like the ones we just described, we investigate three dynamic primitives that can change the accessibility relation of a model: *sabotage* (deletes edges from the model), *bridge* (adds edges to a model), and *swap* (turns around edges), both in a global version (performing changes anywhere in the model) and local (changing adjacent edges from the evaluation point). The particular operators we will investigate should be seen as just examples of the possibilities offered by the framework, with no intention of being complete or comprehensive. Intuitively, they were chosen because they represented simple, different ways in which a relation could be updated.

The six primitive operators we will study in this chapter were first introduced in [AFH12] where their expressive power and the complexity of their model checking problem are investigated. Tableaux methods for relation-changing modal logics were introduced in [AFH13]. Local swap logic is studied in [AFH14], in particular its decidability problem and its relation with first-order logic. In [AFH15] a general framework for representing model updates is defined, and connections with dynamic epistemic logic were introduced in [AvDFS14, AvDFS15]. Finally, we know that the satisfiability problem for the six relation-changing logics considered is undecidable [AFHM17, Mar15]. We thus study the questions introduced in Section 1.2.3:

Q5 Is it possible to provide translations of relation-changing logics to hybrid logics in order to obtain decidable fragments?

Q6 Are relation-changing logics as expressive as hybrid logics? How are they related?

We show that relation-changing logics can be seen as fragments of hybrid logics. We consider hybrid logic because it is the best known *modal* logic that can simulate the semantics of relation-changing operators. We introduce translations to $\mathcal{HL}(\mathbf{E}, \downarrow)$, the basic modal logic extended with nominals, the down arrow binder \downarrow , and the universal modality \mathbf{E} (in some cases the translations fall into the less expressive hybrid logic $\mathcal{HL}(\mathbf{@}, \downarrow)$, i.e., with the satisfiability operator $\mathbf{@}$ instead of \mathbf{E}). We discuss how we can benefit from known decidable fragments of $\mathcal{HL}(\mathbf{E}, \downarrow)$ to find decidable fragments of relation-changing modal logics. We also show that relation-changing logics are strictly less expressive than the hybrid logics they are translated into.

6.1 Relation-Changing Modal Logics

In this section, we formally introduce extensions of the basic modal logic with relation-changing operators. We call these extensions Relation-Changing Modal Logics (RCMLs for short). For more details and motivations, we direct the reader to [Fer14].

Definition 6.1.1 (Syntax). Let PROP be a countable, infinite set of propositional symbols. The set FORM of formulas over PROP is defined as:

$$\text{FORM} ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \diamond\varphi \mid \blacklozenge\varphi,$$

where $p \in \text{PROP}$, $\blacklozenge \in \{\langle \text{sb} \rangle, \langle \text{br} \rangle, \langle \text{sw} \rangle, \langle \text{gsb} \rangle, \langle \text{gbr} \rangle, \langle \text{gsw} \rangle\}$, and $\varphi, \psi \in \text{FORM}$. Other operators are defined as usual.

We denote $\mathcal{ML}(\blacklozenge)$ the extension of \mathcal{BML} allowing the \blacklozenge operator, for $\blacklozenge \in \{\langle \text{sb} \rangle, \langle \text{br} \rangle, \langle \text{sw} \rangle, \langle \text{gsb} \rangle, \langle \text{gbr} \rangle, \langle \text{gsw} \rangle\}$. In particular, $\mathcal{ML}(\langle \text{sb} \rangle, \langle \text{gsb} \rangle)$ will be called Sabotage Logic, $\mathcal{ML}(\langle \text{br} \rangle, \langle \text{gbr} \rangle)$ Bridge Logic, and $\mathcal{ML}(\langle \text{sw} \rangle, \langle \text{gsw} \rangle)$ Swap Logic.

Semantically, formulas are evaluated in standard Kripke models, and the meaning of the operators of the basic modal logic remains unchanged (see Section 2.4). When we evaluate formulas containing relation-changing operators, we will need to keep track of the edges that have been modified. To that end, let us define precisely the model updates we will use.

Definition 6.1.2 (Model Updates). We define the following notations:

$$\begin{aligned} \text{(sabotaging)} \quad & \mathcal{M}_S^- = \langle W, R_S^-, V \rangle, \text{ with } R_S^- = R \setminus S, S \subseteq R. \\ \text{(bridging)} \quad & \mathcal{M}_S^+ = \langle W, R_S^+, V \rangle, \text{ with } R_S^+ = R \cup S, S \subseteq (W \times W) \setminus R. \\ \text{(swapping)} \quad & \mathcal{M}_S^* = \langle W, R_S^*, V \rangle, \text{ with } R_S^* = (R \setminus S^{-1}) \cup S, S \subseteq R^{-1}. \end{aligned}$$

Intuitively, \mathcal{M}_S^- is obtained from \mathcal{M} by deleting the edges in S , and similarly \mathcal{M}_S^+ adds the edges in S to the accessibility relation, and \mathcal{M}_S^* adds the edges in S as inverses of edges previously in the accessibility relation. These operators can be seen as particular cases of the *jump functions* introduced in [Gab99], or the model update functions from [AFH15].

In the rest of the chapter, we consider pointed models \mathcal{M}, w and use wv as a shorthand for $\{(w, v)\}$ or (w, v) ; context will always disambiguate the intended use.

Definition 6.1.3 (Semantics). Given a pointed model \mathcal{M}, w and a formula φ , we say that \mathcal{M}, w *satisfies* φ , and write $\mathcal{M}, w \models \varphi$, when

$\mathcal{M}, w \models p$	iff	$w \in V(p)$
$\mathcal{M}, w \models \neg\varphi$	iff	$\mathcal{M}, w \not\models \varphi$
$\mathcal{M}, w \models \varphi \wedge \psi$	iff	$\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \diamond\varphi$	iff	for some $v \in W$ s.t. $(w, v) \in R, \mathcal{M}, v \models \varphi$
$\mathcal{M}, w \models \langle \text{sb} \rangle \varphi$	iff	for some $v \in W$ s.t. $(w, v) \in R, \mathcal{M}_{wv}^-, v \models \varphi$
$\mathcal{M}, w \models \langle \text{br} \rangle \varphi$	iff	for some $v \in W$ s.t. $(w, v) \notin R, \mathcal{M}_{wv}^+, v \models \varphi$
$\mathcal{M}, w \models \langle \text{sw} \rangle \varphi$	iff	for some $v \in W$ s.t. $(w, v) \in R, \mathcal{M}_{vw}^*, v \models \varphi$
$\mathcal{M}, w \models \langle \text{gsb} \rangle \varphi$	iff	for some $v, u \in W$, s.t. $(v, u) \in R, \mathcal{M}_{vu}^-, w \models \varphi$
$\mathcal{M}, w \models \langle \text{gbr} \rangle \varphi$	iff	for some $v, u \in W$, s.t. $(v, u) \notin R, \mathcal{M}_{vu}^+, w \models \varphi$
$\mathcal{M}, w \models \langle \text{gsw} \rangle \varphi$	iff	for some $v, u \in W$, s.t. $(v, u) \in R, \mathcal{M}_{uv}^*, w \models \varphi$.

We say that φ is *satisfiable* if for some pointed model \mathcal{M}, w we have $\mathcal{M}, w \models \varphi$.

The meaning of the relation-changing operators $\langle \text{sb} \rangle$ (local sabotage), $\langle \text{br} \rangle$ (local bridge), $\langle \text{sw} \rangle$ (local swap), $\langle \text{gsb} \rangle$ (global sabotage), $\langle \text{gbr} \rangle$ (global bridge) and $\langle \text{gsw} \rangle$ (global swap) should be clear from the semantic conditions above. The local operators alter one arrow which is adjacent to the point of evaluation (deleting, adding and swapping it, respectively) while the global versions can change an arrow anywhere in the model.

6.2 Extensions of Modal Logic and Hybrid Logic

In this section, we present several extensions of the basic modal logic. The existential modality [GP92], written $E\varphi$, extends \mathcal{BML} in the following way:

$$\mathcal{M}, w \models E\varphi \quad \text{iff} \quad \text{for some } v \in W, \mathcal{M}, v \models \varphi.$$

In words, $E\varphi$ is true at a state w if φ is true somewhere in the model. The E operator, with its dual A , has been extensively investigated in classical modal logic [Spa93].

Now we consider several traditional ‘hybrid’ operators (see [AtC07] for details): nominals, the satisfaction operator, and the down-arrow binder. The basic hybrid logic \mathcal{HL} is obtained by adding *nominals* to \mathcal{BML} . A nominal is a propositional symbol that is true at exactly one state in a model. Fix the signature $\langle \text{PROP}, \text{NOM} \rangle$, with $\text{NOM} \subseteq \text{PROP}$. For $n \in \text{NOM}$, we require that its valuation is a singleton set, i.e., there is a single state w such that $V(n) = \{w\}$. In addition to nominals, hybrid logic typically involves the *satisfaction operator*. For each nominal n , the satisfaction operator is written $@_n$ and allows us to jump to the point named by n . The formula $@_n\varphi$ (read “at n , φ ”) moves the point of evaluation to the state named by n and evaluates φ there. Its semantics is given by the following clause:

$$\mathcal{M}, w \models @_n\varphi \quad \text{iff} \quad \mathcal{M}, v \models \varphi \text{ where } V(n) = \{v\}.$$

Observe that if the language has the E operator and nominals, then $@_n\varphi$ is definable because $@_n\varphi$ is equivalent to $E(n \wedge \varphi)$.

Finally, consider the *down-arrow binder* operator, written \downarrow . Let the valuation V_n^w be defined by $V_n^w(n) = \{w\}$ and $V_n^w(m) = V(m)$, when $n \neq m$. The semantic condition for \downarrow is the following:

$$\langle W, R, V \rangle, w \models \downarrow n.\varphi \quad \text{iff} \quad \langle W, R, V_n^w \rangle, w \models \varphi.$$

The language $\mathcal{HL}(@, \downarrow)$ is a reduction class of first-order logic, and is thus undecidable [BS95, tC05]. It remains undecidable even with a single accessibility relation, no satisfaction operator, and only nominal propositional symbols [ABM99]. $\mathcal{HL}(E, \downarrow)$ is equivalent to first-order logic, since \downarrow can define the operators \exists and \forall when combined with E and A .

The logic $\mathcal{HL}(E, \downarrow)$ is not able to modify the accessibility relation of a model. However, it can use the binder to name states and, hence, it can refer to *specific edges* in the model. This will be exploited by the translations introduced in the next section.

6.3 Translations to Hybrid Logics

Relation-changing modal logics and hybrid logics with the binder \downarrow are two families of logics that are dynamic in their own way. The dynamicity of RCMLs is quite obvious: they are able to modify the accessibility relation in a model in an explicit way. On the other hand, hybrid logics carefully move nominals around, avoiding to touch anything else in the model. If we consider both formalisms, it would seem that hybrid logics are the gentler and weaker of both. However, this is not true. Hybrid logics have the advantage of surgical precision over RCMLs. Being able to name states of the model and use these names turns out to be a crucial advantage. As we will see now, naming can be used to manipulate *edges* by naming pairs of states using the pattern $\downarrow x. \diamond \downarrow y. \varphi$. We use this naming technique to simulate edge deletion, addition, and swapping.

Our translations are parametrized over a set of pair of nominals $S \subseteq \text{NOM} \times \text{NOM}$. For a given RCML-formula φ , we write its translation as a hybrid formula $(\varphi)'_S$. When translating a formula, S will originally be empty and it will store pairs of nominals that we will use to simulate the edges affected by the relation-changing operators we encounter during the translation.

Intuitively, given that the hybrid operators cannot affect the accessibility relation, we have to simulate the updates by recording possible affected edges using nominals and \downarrow . Notice that as a result, in all the relation-changing logics we will consider, the RCML-formula $\diamond\psi$ cannot be simply translated into a hybrid formula $\diamond(\psi)'_S$, even though we have \diamond at our disposition in the hybrid language, because in the source language \diamond is interpreted over the updated accessibility relation. Instead, diamond-formulas need to be translated in a way that takes into account the edges that should be considered deleted, added, or swapped. This is why the translation of diamond-formulas involve the \diamond operator mixed with specific considerations about the set of altered edges S .

Consider Sabotage Logic with either the local or global operator. We use the set $S \subseteq \text{NOM} \times \text{NOM}$ to represent sabotaged edges, i.e., edges that have been deleted in a given updated model.

Definition 6.3.1 (Sabotage to Hybrid Logic). Let $S \subseteq \text{NOM} \times \text{NOM}$ and $n \in \text{NOM}$. We define the translation $(\)'_S$ from formulas of $\mathcal{ML}(\langle \text{sb} \rangle, \langle \text{gsb} \rangle)$ to formulas of $\mathcal{HL}(E, \downarrow)$ as:

$$\begin{aligned}
(p)'_S &= p \\
(\neg\varphi)'_S &= \neg(\varphi)'_S \\
(\varphi \wedge \psi)'_S &= (\varphi)'_S \wedge (\psi)'_S \\
(\diamond\varphi)'_S &= \downarrow n. \diamond(\neg \text{belongs}(n, S) \wedge (\varphi)'_S) \\
(\langle \text{sb} \rangle\varphi)'_S &= \downarrow n. \diamond(\neg \text{belongs}(n, S) \wedge \downarrow m. (\varphi)'_{S \cup nm}) \\
(\langle \text{gsb} \rangle\varphi)'_S &= \downarrow k. E \downarrow n. \diamond(\neg \text{belongs}(n, S) \wedge \downarrow m. @_k(\varphi)'_{S \cup nm})
\end{aligned}$$

where n , m and k are nominals that do not appear in S , and:

$$\text{belongs}(n, S) = \bigvee_{xy \in S} (y \wedge @_n x)$$

Some explanations are in order to understand the translation. First, given some model $\mathcal{M} = \langle W, R, V \rangle$ and some set $S \subseteq \text{NOM} \times \text{NOM}$, the formula $\downarrow n. \diamond (\neg \text{belongs}(n, S))$ is true at some state $w \in W$ if there exists some state v such that $(w, v) \in R$ and there is no pair of nominals $(x, y) \in S$ such that $(V(x), V(y)) = (w, v)$. Observe that the cases for $\langle \text{sb} \rangle$ and $\langle \text{gsb} \rangle$ modify the set of deleted pairs in the recursive call to the translation by adding an edge named nm . In the $\langle \text{sb} \rangle$ case, n names the evaluation state of the formula, while in the $\langle \text{gsb} \rangle$ case, n names some state anywhere in the model.

We will now prove that the translation preserves equivalence. We start by introducing some preliminary notions and definitions.

First, notice that all nominals used in the translation are bound exactly once. We can, then, define the following unequivocal notation: let $S \subseteq \text{NOM} \times \text{NOM}$, we define $\bar{S} = \{(\bar{x}, \bar{y}) \mid (x, y) \in S\}$, where \bar{n} is the state named by the nominal $n \in \text{NOM}$ under the current valuation of a model.

Second, when considering the truth of a translated formula $(\varphi)'_S$ in some model $\mathcal{M} = \langle W, R, V \rangle$, one question that may arise is what should be the initial valuation of the nominals that appear in $(\varphi)'_S$. Because all nominals in $(\varphi)'_S$ are bound by \downarrow , the truth value of $(\varphi)'_S$ does not depend on their initial valuation. Even if these symbols are not treated as nominals in the original model \mathcal{M} they will be interpreted correctly when evaluating $(\varphi)'_S$. This enables us to talk about equivalence preservation of the translation over the same model \mathcal{M} .

Theorem 6.3.1. *For $\mathcal{M} = \langle W, R, V \rangle$ a model, $w \in W$, and $\varphi \in \mathcal{ML}(\langle \text{sb} \rangle, \langle \text{gsb} \rangle)$ we have:*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}, w \models (\varphi)'_{\emptyset}.$$

Proof. We use structural induction on the relation-changing formula, the inductive hypothesis being:

$$\mathcal{M}_{\bar{S}}^-, w \models \varphi \text{ iff } \langle W, R, V' \rangle, w \models (\varphi)'_S$$

with $S \subseteq \text{NOM} \times \text{NOM}$, and V' is exactly as V except that for all $(x, y) \in S$, there are $v, u \in W$ such that $V'(x) = v$ and $V'(y) = u$. Boolean cases are straightforward, so we only prove the non-trivial inductive cases.

$\varphi = \diamond \psi$: For the left to right direction, suppose $\mathcal{M}_{\bar{S}}^-, w \models \diamond \psi$. Then there is some $v \in W$ such that $(w, v) \in R_{\bar{S}}^-$ and $\mathcal{M}_{\bar{S}}^-, v \models \psi$. Because $(w, v) \notin \bar{S}$, then there is no $(x, y) \in S$ such that $(\bar{x}, \bar{y}) = (w, v)$. By inductive hypothesis, we have $\mathcal{M}, v \models (\psi)'_S$, and because we can name w with a fresh nominal n , we obtain $\langle W, R, V_n^w \rangle, v \models \neg \text{belongs}(n, S) \wedge (\psi)'_S$. Therefore, we have $\mathcal{M}, w \models \downarrow n. \diamond (\neg \text{belongs}(n, S) \wedge (\psi)'_S)$, and as a consequence we get $\mathcal{M}, w \models (\psi)'_S$.

For the other direction, suppose $\mathcal{M}, w \models (\psi)'_S$, i.e., $\mathcal{M}, w \models \downarrow n. \diamond (\neg \text{belongs}(n, S) \wedge (\psi)'_S)$. Then we have $\langle W, R, V_n^w \rangle, w \models \diamond (\neg \text{belongs}(n, S) \wedge (\psi)'_S)$, and, by definition, there is some $v \in W$ such that $(w, v) \in R$, $\langle W, R, V_n^w \rangle, v \models \neg \text{belongs}(n, S)$ and $\langle W, R, V_n^w \rangle, v \models (\psi)'_S$. Because we have $\neg \text{belongs}(n, S)$, there is no $(x, y) \in S$ such that $(\bar{x}, \bar{y}) = (w, v)$, which implies $(w, v) \in R$ if and only if $(w, v) \in R_{\bar{S}}^-$. On the other hand, by inductive hypothesis we have $\mathcal{M}_{\bar{S}}^-, v \models \psi$, then we have $\mathcal{M}_{\bar{S}}^-, w \models \diamond \psi$.

$\varphi = \langle \text{sb} \rangle \psi$: For the left to right direction, suppose $\mathcal{M}_{\bar{S}}^-, w \models \langle \text{sb} \rangle \psi$. Then there is some $v \in W$ such that $(w, v) \in R_{\bar{S}}^-$ and $(\mathcal{M}_{\bar{S}}^-)_{wv}^-, v \models \psi$. This is equivalent to say $\mathcal{M}_{\bar{S} \cup wv}^-, v \models$

ψ . Because $(w, v) \notin \bar{S}$, then there is no $(x, y) \in S$ such that $(\bar{x}, \bar{y}) = (w, v)$ (\otimes). By inductive hypothesis we have $\langle W, R, ((V')_n^w)_m^v \rangle, v \models (\psi)'_{S \cup nm}$, where V' is exactly as V but it binds all the nominals which appear in S . By definition, we get $\langle W, R, (V')_n^w \rangle, v \models \downarrow m.(\psi)'_{S \cup nm}$, and by (\otimes) we have $\langle W, R, (V')_n^w \rangle, v \models \neg \text{belongs}(n, S) \wedge \downarrow m.(\psi)'_{S \cup nm}$. Then (by definition) $\langle W, R, V' \rangle, v \models \downarrow n. \diamond (\neg \text{belongs}(n, S) \wedge \downarrow m.(\psi)'_{S \cup nm})$, and, as a consequence, we have $\langle W, R, V' \rangle, v \models (\varphi)'_S$.

For the other direction, suppose $\langle W, R, V' \rangle, w \models (\psi)'_S$, i.e., $\langle W, R, V' \rangle, w \models \downarrow n. \diamond (\neg \text{belongs}(n, S) \wedge \downarrow m.(\psi)'_{S \cup nm})$, where V' is exactly as V but it binds all the nominals which appear in S . Then, we have $\langle W, R, (V')_n^w \rangle, w \models \diamond (\neg \text{belongs}(n, S) \wedge \downarrow m.(\psi)'_{S \cup nm})$, and, by definition, there is some $v \in W$ such that $(w, v) \in R$, $\langle W, R, V_n^w \rangle, v \models \neg \text{belongs}(n, S)$ and $\langle W, R, V_n^w \rangle, v \models \downarrow m.(\psi)'_{S \cup nm}$. Then, we have $\langle W, R, ((V')_n^w)_m^v \rangle, v \models (\psi)'_{S \cup nm}$. Because we have $\neg \text{belongs}(n, S)$, there is no $(x, y) \in S$ such that $(\bar{x}, \bar{y}) = (w, v)$, which implies $(w, v) \in R$ if and only if $(w, v) \in R_{\bar{S}}$. On the other hand, by inductive hypothesis we have $\mathcal{M}_{\bar{S} \cup wv}^-, v \models \psi$, and thus we have $\mathcal{M}_{\bar{S}}^-, w \models \langle \text{sb} \rangle \psi$.

$\varphi = \langle \text{gsb} \rangle \psi$: this case is very similar to the previous one. \square

For Bridge Logic, we use the set $B \subseteq \text{NOM} \times \text{NOM}$ to represent the new edges. New edges present in B mean that the translation of the modality \diamond should be able to take them. This explains why the translation of \diamond does not look like a \diamond with an extra condition, but like an E with two possibilities: we traverse an edge that is either in the original model or an edge from the B set.

Definition 6.3.2 (Bridge to Hybrid Logic). Let $B \subseteq \text{NOM} \times \text{NOM}$. We define $(\)'_B$ from formulas of $\mathcal{ML}(\langle \text{br} \rangle, \langle \text{gbr} \rangle)$ to formulas of $\mathcal{HL}(E, \downarrow)$ as:

$$\begin{aligned} (p)'_B &= p \\ (\neg \varphi)'_B &= \neg(\varphi)'_B \\ (\varphi \wedge \psi)'_B &= (\varphi)'_B \wedge (\psi)'_B \\ (\diamond \varphi)'_B &= \downarrow n. E \downarrow m. (\text{@}_n \diamond m \vee \text{belongs}(n, B)) \wedge (\varphi)'_B \\ (\langle \text{br} \rangle \varphi)'_B &= \downarrow n. E \downarrow m. (\neg \text{@}_n \diamond m \wedge \neg \text{belongs}(n, B) \wedge (\varphi)'_{B \cup nm}) \\ (\langle \text{gbr} \rangle \varphi)'_B &= \downarrow k. E \downarrow n. E \downarrow m. (\neg \text{@}_n \diamond m \wedge \neg \text{belongs}(n, B) \wedge \text{@}_k (\varphi)'_{B \cup nm}) \end{aligned}$$

where n, m and k are nominals that do not appear in B , and belongs is defined as in Definition 6.3.1.

Theorem 6.3.2. For $\mathcal{M} = \langle W, R, V \rangle$ a model, $w \in W$, and $\varphi \in \mathcal{ML}(\langle \text{br} \rangle, \langle \text{gbr} \rangle)$, we have:

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}, w \models (\varphi)'_{\emptyset}.$$

Proof. A similar reasoning can be done with the following inductive hypothesis:

$$\mathcal{M}_B^+, w \models \varphi \text{ iff } \langle W, R, V' \rangle, w \models (\varphi)'_B$$

with $B \subseteq \text{NOM} \times \text{NOM}$, and V' is exactly as V except that for all $(x, y) \in B$, there are $v, u \in W$ such that $V'(x) = v$ and $V'(y) = u$. \square

We finish with the case of Swap Logic. A different translation is presented in [AFH14] for the local case only. As we did for Sabotage Logic, we use $S \subseteq \text{NOM} \times \text{NOM}$ to represent the set of deleted edges, i.e., the edges that should not be possible to traverse in a given updated model. Indeed, swapping a non-reflexive edge of a model has the effect of deleting it, along with adding its inverse. This implies that S^{-1} is a set of

edges that we can currently traverse. All of this requires that S do not contain any reflexive edge, since a swapped reflexive edge is not deleted. Neither can it contain a pair of symmetric edges since that would be contradictory.

To ensure this, the translation gets more cautious when handling $\langle \text{sw} \rangle$ and $\langle \text{gsw} \rangle$. When swapping occurs, three possible cases are taken into account. The first one is when a reflexive edge is swapped. In that case, the translation continues with the set S left unchanged, but we require some reflexive edge to be present, be it at the current state for $\langle \text{sw} \rangle$ with $\downarrow n. \diamond n$, or anywhere in the model for $\langle \text{gsw} \rangle$ with $E \downarrow n. \diamond n$.

The second case is when we swap an irreflexive edge that has never been swapped before. Hence we ensure that this edge is present in the model, that it is irreflexive, and that neither this edge nor its inverse is in S . We then add the nominals that name it to S before moving on with the translation.

The last case is when we traverse an already swapped edge. That is, for some $xy \in S$, we traverse the edge referred to by the nominals yx . In this case, we do not need to require the presence of any new edge in the model. We assume to be standing at the state named by y and that the rest of the formula is satisfied at x , with the modification that we remove xy from S and add yx to it.

An attentive reader would object: why not just remove xy from the set S since swapping some edge twice just makes it return to its configuration in the original model? The answer is that there is a corner case when some edge *and* its symmetric are both present in the initial model. Then, the action of swapping it twice is not supposed to restore its symmetric. This is what we do by adding yx to the set S : we ensure the former symmetric edge is no longer present.

Definition 6.3.3 (Swap to Hybrid Logic). Let $S \subseteq \text{NOM} \times \text{NOM}$. We define $(\)'_S$ from formulas of $\mathcal{ML}(\langle \text{sw} \rangle, \langle \text{gsw} \rangle)$ to formulas of $\mathcal{HL}(E, \downarrow)$ as:

$$\begin{aligned}
(p)'_S &= p \\
(\neg\varphi)'_S &= \neg(\varphi)'_S \\
(\varphi \wedge \psi)'_S &= (\varphi)'_S \wedge (\psi)'_S \\
(\diamond\varphi)'_S &= (\downarrow n. \diamond (\neg \text{belongs}(n, S) \wedge (\varphi)'_S)) \vee \text{isSat}(S^{-1}, (\varphi)'_S) \\
(\langle \text{sw} \rangle \varphi)'_S &= (\downarrow n. \diamond n \wedge (\varphi)'_S) \\
&\quad \vee \downarrow n. \diamond (\neg n \wedge \neg \text{belongs}(n, S) \wedge \neg \text{belongs}(n, S^{-1}) \wedge \downarrow m. (\varphi)'_{S \cup nm}) \\
&\quad \vee \bigvee_{xy \in S} (y \wedge @_x (\varphi)'_{(S \setminus xy) \cup yx}) \\
(\langle \text{gsw} \rangle \varphi)'_S &= (E \downarrow n. \diamond n \wedge (\varphi)'_S) \\
&\quad \vee \downarrow k. E \downarrow n. \diamond (\neg n \wedge \neg \text{belongs}(n, S) \wedge \neg \text{belongs}(n, S^{-1}) \wedge \downarrow m. @_k (\varphi)'_{S \cup nm}) \\
&\quad \vee \bigvee_{xy \in S} (\varphi)'_{(S \setminus xy) \cup yx}
\end{aligned}$$

where n , m and k are nominals that do not appear in S , belongs is defined as in Definition 6.3.1, and

$$\text{isSat}(S, \varphi) = \bigvee_{xy \in S} (x \wedge @_y \varphi).$$

The formula $\text{isSat}(S, (\varphi)'_S)$ says that the translation of φ is satisfiable at the end of some of the edges belonging to S . Note that the translation maps formulas of $\mathcal{ML}(\langle \text{sw} \rangle)$ to the less expressive $\mathcal{HL}(@, \downarrow)$, i.e., the E operator is not required.

Theorem 6.3.3. For $\mathcal{M} = \langle W, R, V \rangle$ a model, $w \in W$ and $\varphi \in \mathcal{ML}(\langle \text{sw} \rangle, \langle \text{gsw} \rangle)$ we have:

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}, w \models (\varphi)'_{\emptyset}.$$

Proof. Again, a similar reasoning can be done with the following inductive hypothesis:

$$\mathcal{M}_{S^{-1}}^*, w \models \varphi \text{ iff } \langle W, R, V' \rangle, w \models (\varphi)'_S$$

with $S \subseteq \text{NOM} \times \text{NOM}$, and V' is exactly as V except that for all $(x, y) \in S$, there are $v, u \in W$ such that $V'(x) = v$ and $V'(y) = u$. \square

6.4 Decidable Fragments

Interesting decidable fragments of hybrid logics with binders have been found over time. Such decidable fragments are convenient for our relation-changing logics in the light of the (computable) translations presented in Section 6.3. First, let us consider restricting the satisfiability problem over certain classes of models. The following logics are known to be decidable over the indicated classes:

- $\mathcal{HL}(\mathbf{E}, \downarrow)$ over linear frames (i.e., irreflexive, transitive, and trichotomous frames) [FdRS03, Sch07] (this includes $(\mathbb{N}, <)$),
- $\mathcal{HL}(\mathbf{E}, \downarrow)$ over models with a single, transitive tree relation [Sch07],
- $\mathcal{HL}(\mathbf{E}, \downarrow)$ over models with a single, $S5$, or complete relation [Sch07],
- $\mathcal{HL}(@, \downarrow)$ over models with a single relation of bounded finite width [tCF05]; as a corollary, also over finite models.

Since the translations preserve equivalence, we get:

Corollary. *The satisfiability problem for all relation-changing modal logics over linear, transitive trees, $S5$, and complete frames is decidable.*

Corollary. *The satisfiability problem for local sabotage and local swap logics over models of bounded width is decidable.*

Curiously, these results mean that relation-changing modal logics are decidable over certain classes of models, even if the modifications implied by evaluating RCML-formulas yield models that *do not* belong to such class. For instance, these two facts are simultaneously true: sabotage logic is decidable on the class of $S5$ models, and deleting edges in an $S5$ model can yield a non- $S5$ model.

Now, let us turn to syntactical definitions of decidable fragments. We recall that local sabotage and local swap can be translated to $\mathcal{HL}(@, \downarrow)$. Consider formulas of $\mathcal{HL}(@, \downarrow)$ in negation normal form. $\mathcal{HL}(@, \downarrow) \setminus \square \downarrow \square$ is the fragment obtained by removing formulas that contain a nesting of \square , \downarrow and again \square . This fragment is decidable [tCF05].

Our translations do use the \downarrow binder in many places, but we can make them a little more economical in that sense, at the expense of losing succinctness.

Take the following case for $\mathcal{ML}(\langle \text{sw} \rangle)$:

$$(\diamond \varphi)'_S = \downarrow n. \diamond (\neg \text{belongs}(n, S) \wedge (\varphi)'_S).$$

Instead of using the down-arrow binder and later ensuring that we did not take a deleted edge by using $\neg \text{belongs}(n, S)$, we can do the following. For all pairs of nominals $(x, y) \in S$, the current state w satisfies one combination of the truth values of the

nominals x . Let X be the set of true nominals x at w . Then, $(\varphi)'_S$ should be true at some accessible state v that should not satisfy any of the corresponding y nominals for all $x \in X$.

Then, the translation becomes:

$$(\diamond\varphi)'_S = \bigvee_{X \subseteq \text{fst}(S)} \left(\bigwedge_{x \in X} x \wedge \bigwedge_{x \notin X} \neg x \wedge \diamond \left(\bigwedge_{y \in \text{snd}(S, X)} \neg y \wedge (\varphi)'_S \right) \right)$$

where $\text{fst}(S) = \{x \mid (x, y) \in S\}$ and $\text{snd}(S, X) = \{y \mid (x, y) \in S, x \in X\}$.

In the case of $\mathcal{ML}(\langle \text{sw} \rangle)$ we can do the same. We recall that the case introduced in Section 6.3 was:

$$(\diamond\varphi)'_S = (\downarrow n. \diamond(\neg \text{belongs}(n, S) \wedge (\varphi)'_S)) \vee \text{isSat}(S^{-1}, (\varphi)'_S).$$

Here the $\text{isSat}(S^{-1}, (\varphi)'_S)$ disjunct does not use the \downarrow binder, while the first disjunct is similar to the case of local sabotage, and can be replaced accordingly:

$$(\diamond\varphi)'_S = \bigvee_{X \subseteq \text{fst}(S)} \left(\bigwedge_{x \in X} x \wedge \bigwedge_{x \notin X} \neg x \wedge \diamond \left(\bigwedge_{y \in \text{snd}(S, X)} \neg y \wedge (\varphi)'_S \right) \right) \vee \text{isSat}(S^{-1}, (\varphi)'_S).$$

Let \blacklozenge be either $\langle \text{sb} \rangle$ or $\langle \text{sw} \rangle$ and \blacksquare be either $[\text{sb}]$ or $[\text{sw}]$. The following patterns in RCML-formulas result in the shown patterns in the hybrid formula obtained by the translations:

RCML pattern	Produced pattern
\square	\square
\blacklozenge	\downarrow
\blacksquare	$\downarrow \square \downarrow$

By considering these new versions of the translations, and by taking into account the syntactic decidable fragment of $\mathcal{HL}(@, \downarrow)$ mentioned above, we can establish the following result:

Corollary. *The following fragments are decidable on the class of all relational models:*

- $\mathcal{ML}(\langle \text{sb} \rangle) \setminus \{\blacksquare\blacksquare, \blacksquare\square, \square\blacksquare, \blacksquare\blacklozenge\blacksquare\}$
- $\mathcal{ML}(\langle \text{sw} \rangle) \setminus \{\blacksquare\blacksquare, \blacksquare\square, \square\blacksquare, \blacksquare\blacklozenge\blacksquare\}$

where \blacksquare is either \square or \blacksquare .

6.5 Comparing Expressive Power

We have introduced translations for the six relation-changing modal logics from Section 6.1 into hybrid logic. In some cases (for the local version of swap and sabotage), the obtained formulas fall into the fragment $\mathcal{HL}(@, \downarrow)$. On the other hand, for encoding the rest of the logics we need also to use the universal modality \mathbf{E} . An interesting question is whether we can obtain translations from hybrid to relation-changing logics, i.e., if some of the relation-changing logics considered in this chapter are as expressive as some hybrid logic.

In [AFH12, AFH14, Fer14, AFH15] the expressive power of relation-changing modal logics is discussed by comparing the logics among each other using appropriate notions of bisimulations, and it is shown that they are all *incomparable in terms of expressive*

power.¹ As a consequence, we conclude that it is not possible that two of them capture the same fragment of hybrid logic. In fact, we will prove that all the relation-changing logics considered here are strictly less expressive than the corresponding hybrid logic in which they are translated (see Theorem 6.5.2 below).

Let us first introduce bisimulations for RCMLs. Because we need to keep track of the changes on the accessibility relation that the dynamic operators can introduce, we define bisimulations as relations that link a point of evaluation together with the current accessibility relation. In [Fer14, AFH15] the following notions of bisimulations are introduced.

Definition 6.5.1 ($\mathcal{ML}(\diamond)$ -Bisimulations). Let $\mathcal{M} = \langle W, R, V \rangle$ and $\mathcal{M}' = \langle W', R', V' \rangle$ be two models. A non-empty relation $Z \subseteq (W \times \mathcal{P}(W^2)) \times (W' \times \mathcal{P}(W'^2))$ is a $\mathcal{ML}(\diamond)$ -bisimulation if it satisfies the conditions *atomic harmony*, *zig* and *zag* below, and the corresponding \diamond -*zig* and \diamond -*zag* conditions that the considered logic contains, for $\diamond \in \{\langle \text{sb} \rangle, \langle \text{gsb} \rangle, \langle \text{br} \rangle, \langle \text{gbr} \rangle, \langle \text{sw} \rangle, \langle \text{gsw} \rangle\}$. If $(w, S)Z(w', S')$ then

- (**atomic harmony**) for all $p \in \text{PROP}$, $w \in V(p)$ iff $w' \in V'(p)$;
- (**zig**) if $(w, v) \in S$ then for some v' , $(w', v') \in S'$ and $(v, S)Z(v', S')$;
- (**zag**) if $(w', v') \in S'$ then for some v , $(w, v) \in S$ and $(v, S)Z(v', S')$;
- (**\langle sb \rangle-zig**) if $(w, v) \in S$ then for some v' , $(w', v') \in S'$ and $(v, S_{wv}^-)Z(v', S_{w'v'}^-)$;
- (**\langle sb \rangle-zag**) if $(w', v') \in S'$ then for some v , $(w, v) \in S$ and $(v, S_{wv}^-)Z(v', S_{w'v'}^-)$;
- (**\langle gsb \rangle-zig**) if $(u, v) \in S$ then for some u' , v' , $(u', v') \in S'$ and $(w, S_{uv}^-)Z(w', S_{u'v'}^-)$;
- (**\langle gsb \rangle-zag**) if $(u', v') \in S'$ then for some u, v , $(u, v) \in S$ and $(w, S_{uv}^-)Z(w', S_{u'v'}^-)$;
- (**\langle br \rangle-zig**) if $(w, v) \notin S$ then for some v' , $(w', v') \notin S'$ and $(v, S_{wv}^+)Z(v', S_{w'v'}^+)$;
- (**\langle br \rangle-zag**) if $(w', v') \notin S'$ then for some v , $(w, v) \notin S$ and $(v, S_{wv}^+)Z(v', S_{w'v'}^+)$;
- (**\langle gbr \rangle-zig**) if $(u, v) \notin S$ then for some u' , v' , $(u', v') \notin S'$ and $(w, S_{uv}^+)Z(w', S_{u'v'}^+)$;
- (**\langle gbr \rangle-zag**) if $(u', v') \notin S'$ then for some u, v , $(u, v) \notin S$ and $(w, S_{uv}^+)Z(w', S_{u'v'}^+)$;
- (**\langle sw \rangle-zig**) if $(w, v) \in S$ then for some v' , $(w', v') \in S'$ and $(v, S_{vw}^*)Z(v', S_{v'w'}^*)$;
- (**\langle sw \rangle-zag**) if $(w', v') \in S'$ then for some v , $(w, v) \in S$ and $(v, S_{vw}^*)Z(v', S_{v'w'}^*)$;
- (**\langle gsw \rangle-zig**) if $(u, v) \in S$ then for some u' , v' , $(u', v') \in S'$ and $(w, S_{vu}^*)Z(w', S_{v'u'}^*)$;
- (**\langle gsw \rangle-zag**) if $(u', v') \in S'$ then for some u, v , $(u, v) \in S$ and $(w, S_{vu}^*)Z(w', S_{v'u'}^*)$.

Given two pointed models \mathcal{M}, w and \mathcal{M}', w' we say that they are $\mathcal{ML}(\diamond)$ -bisimilar and write $\mathcal{M}, w \Leftrightarrow_{\mathcal{ML}(\diamond)} \mathcal{M}', w'$ if there is a $\mathcal{ML}(\diamond)$ -bisimulation Z such that $(w, R)Z(w', R')$ where R and R' are respectively the relations of \mathcal{M} and \mathcal{M}' .

The next theorem establishes that two bisimilar models are not distinguishable for any formula of the corresponding language.

Theorem 6.5.1 (Invariance Under Bisimulations). *Let $\mathcal{M} = \langle W, R, V \rangle$ and $\mathcal{M}' = \langle W', R', V' \rangle$ be two models, $w \in W$, $w' \in W'$, and let $S \subseteq W^2$, $S' \subseteq W'^2$. If there is a $\mathcal{ML}(\diamond)$ -bisimulation Z between \mathcal{M}, w and \mathcal{M}', w' such that $(w, S)Z(w', S')$ then for any formula $\varphi \in \mathcal{ML}(\diamond)$, $\langle W, S, V \rangle, w \models \varphi$ iff $\langle W', S', V' \rangle, w' \models \varphi$.*

¹Except for the local and global swap operators, which is still open in one direction.

The proof is standard and can be found, e.g., in [Fer14]. With the appropriate notions of bisimulation at hand we can now start our study of expressive power. The next definition formalizes how we compare the expressive power of two logics.

Definition 6.5.2 ($\mathcal{L} \leq \mathcal{L}'$). We say that \mathcal{L}' is *at least as expressive as* \mathcal{L} (notation $\mathcal{L} \leq \mathcal{L}'$) if there is a function Tr between formulas of \mathcal{L} and \mathcal{L}' such that for every model \mathcal{M} and every formula φ of \mathcal{L} we have that

$$\mathcal{M} \models_{\mathcal{L}} \varphi \text{ iff } \mathcal{M} \models_{\mathcal{L}'} \text{Tr}(\varphi).$$

\mathcal{M} is seen as a model of \mathcal{L} on the left and as a model of \mathcal{L}' on the right, and we use in each case the appropriate semantic relation $\models_{\mathcal{L}}$ or $\models_{\mathcal{L}'}$ as required.

\mathcal{L}' is strictly more expressive than \mathcal{L} ($\mathcal{L} < \mathcal{L}'$) if $\mathcal{L} \leq \mathcal{L}'$ but $\mathcal{L}' \not\leq \mathcal{L}$. Finally, we say that \mathcal{L} and \mathcal{L}' are *incomparable* if $\mathcal{L} \not\leq \mathcal{L}'$ and $\mathcal{L}' \not\leq \mathcal{L}$.

We are now ready to state the main theorem of this section.

Theorem 6.5.2. *Let $\blacklozenge_1 \in \{\langle \text{sb} \rangle, \langle \text{sw} \rangle\}$, we have $\mathcal{ML}(\blacklozenge_1) < \mathcal{HL}(\@, \downarrow)$. For $\blacklozenge_2 \in \{\langle \text{gsb} \rangle, \langle \text{gsw} \rangle, \langle \text{br} \rangle, \langle \text{gbr} \rangle\}$, we have $\mathcal{ML}(\blacklozenge_2) < \mathcal{HL}(\text{E}, \downarrow)$.*

Proof. For any of the logics mentioned above, we have translations into the corresponding hybrid logic. Now we need to prove that these translations do not cover their entire target language (modulo equivalence). In order to do that, we provide bisimilar models for relation-changing modal logics which can be distinguished by some hybrid formula. In Figure 6.1, we show two pairs of models already introduced in [AFH15] that cover all possibilities of bisimilarity.



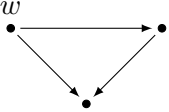

\mathcal{M}, w	\mathcal{M}', w'	Bisimilar for
		$\mathcal{ML}(\langle \text{sw} \rangle)$ $\mathcal{ML}(\langle \text{br} \rangle)$ $\mathcal{ML}(\langle \text{gsw} \rangle)$ $\mathcal{ML}(\langle \text{gbr} \rangle)$
		$\mathcal{ML}(\langle \text{sb} \rangle)$ $\mathcal{ML}(\langle \text{gsb} \rangle)$

Figure 6.1: Bisimilar models

The two models in the first row can be distinguished by the formula $\downarrow n. \Box n$, which establishes that the only successor of the evaluation point is itself. This formula is true at \mathcal{M}, w and false at \mathcal{M}', w' . Models in the second row can be distinguished by the formula $\downarrow n. \Diamond \downarrow m. \@_n \Diamond \Diamond m$, which says that from the evaluation point it is possible to arrive to the same state in one or two steps. This is true at \mathcal{M}, w but false at \mathcal{M}', w' . \square

Notice that both hybrid formulas we introduced above belong to the fragment $\mathcal{HL}(\@, \downarrow)$, i.e., it was not necessary to use the E operator. This means that even though we use E in some of the translations (and we strongly believe that it is essential for some encodings) there are fragments of $\mathcal{HL}(\@, \downarrow)$ that cannot be captured by relation-changing modal logics.

6.6 Concluding Remarks

We have introduced equivalence-preserving translations from six logics we named *relation-changing* to a very expressive hybrid logic. We considered three kinds of modifications: deleting, adding, and swapping edges, that can be performed both globally (anywhere in the model) and locally (modifying adjacent edges from the evaluation point). On the other hand, hybrid logic has operators to rename states in a model with some particular atomic symbols named nominals. We use the down-arrow operator \downarrow to name pairs of states that represent modified states. In this way, we keep track of the evolution of a model.

It is known that the hybrid logic $\mathcal{HL}(\mathbf{E}, \downarrow)$ has the same expressive power as \mathcal{FOL} , and standard translations from relation-changing logics to \mathcal{FOL} were introduced in [AFH15]. However, by giving explicit translations to hybrid logic we can benefit from its decidable fragments to find decidable fragments of relation-changing modal logics. Also, these translations are useful to analyze expressive power. We showed that the six logics we considered are strictly less expressive than $\mathcal{HL}(\mathbf{E}, \downarrow)$. In fact, despite we used the modality \mathbf{E} in some translations, all relation-changing logics we considered here cannot capture the full fragment $\mathcal{HL}(@, \downarrow)$ (which is less expressive than $\mathcal{HL}(\mathbf{E}, \downarrow)$). In summary, we learned that relation-changing modal logics are languages that enable to talk directly and succinctly about distinct kinds of model modifications, but with a little effort they can be simulated by hybrid logics.

We studied six relation-changing modal logics with the goal of covering a sufficiently varied sample of alternatives. Clearly, other operators could have been included in this exploration, and actually some alternative choices have been already investigated in the literature, e.g., the adjacent sabotage operator discussed in [Roh06], or the more generic approach investigated in [AFH15].

There are still many interesting questions to be answered. The *hybrid perspective* introduced in this chapter gives us a new way to think of the relation-changing framework. As future work, we can use *hybridization techniques* (a very standard technique in modal logic [BdRV01]) to find complete axiomatizations. Moreover, it would be interesting to compute interpolants constructively using tableaux techniques [BM03]. For example, tableau formulas as defined in [AFH13] contain the current model variant together with prefixes which indicate in which point of the model the formula has to be evaluated. This syntactic information contained in prefixes is useful information to construct interpolants, but prefixes are not directly expressible in relation-changing modal logics. An alternative would be to extend relation-changing modal logics with hybrid operators, and adapt the tableau rules to compute interpolants.

We have studied various topics in this thesis. It is now time to put everything in perspective and briefly review what we have learned so far.

We started by studying the computational complexity of (deductive) conservative extensions in expressive fragments of first-order logic, such as the two-variable fragment, the guarded fragment, and the guarded negation fragment. We pursued this line of research with the goal of understanding the limits of decidability of conservative extensions. We learned that conservative extensions are undecidable in the two-variable fragment and the guarded fragment, and that decidability can be retained if we allow in the language both guardedness and two variables. We left open the possibility to extend these fragments with counting, fixed points, transitive relations, etc. to further understand where is the border of decidability, which we believe is a very interesting line of research to pursue in the future.

After this foundational work, we moved on to study conservative extensions in ontology-based data access scenarios, where answering queries under ontologies is the most important task. We studied (query) conservative extensions in Horn description logics with inverse roles, established decidability and obtained complexity results. Although we left some problems open, there is much more to do. In particular, there is a recent trend to investigate rule-based languages (such as existential rules) as ontology languages, and it would be interesting to study (query) conservative extensions in these contexts.

Following this research trend in studying the complexity of query answering with background knowledge, we studied the computational complexity of ontology-mediated queries in the unary negation fragment extended with regular path expressions on binary relations. Since the unary negation fragment can express union of conjunctive queries as formulas, it was possible to reduce the ontology-mediated query answering problem to the satisfiability problem in order to obtain complexity results.

We then moved on from ontology languages to modal languages to investigate the expressive power of modal logics, with a special interest in modal logics that can modify the accessibility relation of a model during the evaluation of a formula. We studied their expressive power both thorough syntactic and semantic characterizations. We began by providing *syntactic characterizations*, and in particular, we presented translations into hybrid logic. We then turned to *semantic characterizations* by using appropriate

notions of bisimulations as natural notions of equivalence between models to compare expressive power. Although in the last chapter we dealt explicitly with expressive power, we also used semantic characterizations through the thesis to obtain complexity results. In particular, we provided model-theoretic characterizations based on appropriate notions of bisimulations or homomorphisms to characterize conservative extensions in fragments of first-order logic, and to characterize the satisfiability problem in extensions of the unary negation fragment, which were then used to provide decision procedures based on tree automata.

All in all, we have covered the computational complexity and expressive power of different logics that are relevant for both a theoretical and practical perspective. We obtained interesting results in this thesis, but of course this is just a start as there are several open problems to investigate and different directions of research one can follow.

Bibliography

- [ABBV16] Antoine Amarilli, Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Query answering with transitive and linear-ordered data. In *Proc. IJCAI*, pages 893–899, 2016. Cited on page [114](#).
- [ABCR16] Marcelo Arenas, Elena Botoeva, Diego Calvanese, and Vladislav Ryzhikov. Knowledge base exchange: The case of OWL 2 QL. *Artif. Intell.*, 238:11–62, 2016. Cited on page [55](#).
- [ABM99] Carlos Areces, Patrick Blackburn, and Maarten Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in Lecture Notes in Computer Science, pages 307–321, Madrid, Spain, 1999. Springer. Cited on page [131](#).
- [AFH12] Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Moving Arrows and Four Model Checking Results. In *Logic, Language, Information and Computation*, volume 7456 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2012. Cited on pages [8](#), [128](#), and [136](#).
- [AFH13] Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Tableaux for Relation-Changing Modal Logics. In *Frontiers of Combining Systems - 9th International Symposium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings*, pages 263–278, 2013. Cited on pages [128](#) and [139](#).
- [AFH14] Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Swap Logic. *Logic Journal of the IGPL*, 22(2):309–332, 2014. Cited on pages [128](#), [133](#), and [136](#).
- [AFH15] Carlos Areces, Raul Fervari, and Guillaume Hoffmann. Relation-Changing Modal Operators. *Logic Journal of the IGPL*, 23(4):601–627, 2015. Cited on pages [128](#), [129](#), [136](#), [137](#), [138](#), and [139](#).
- [AFHM16] Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Relation-Changing Logics as Fragments of Hybrid Logics. In *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016*, pages 16–29, 2016. Cited on page [11](#).

- [AFHM17] Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Undecidability of Relation-Changing Modal Logics. In *Dynamic Logic. New Trends and Applications - First International Workshop, DALI 2017, Brasilia, Brazil, September 23-24, 2017, Proceedings*, pages 1–16, 2017. Cited on pages 9, 11, and 128.
- [AFHM18] Carlos Areces, Raul Fervari, Guillaume Hoffmann, and Mauricio Martel. Satisfiability for Relation-Changing Logics. *To appear in Journal of Logic and Computation*, 2018. Cited on page 11.
- [AG08] Renzo Angles and Claudio Gutiérrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, 2008. Cited on page 100.
- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. Cited on page 1.
- [ANv98] Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philosophical Logic*, 27(3):217–274, 1998. Cited on pages 3, 14, 19, and 30.
- [AtC07] Carlos Areces and Balder ten Cate. Hybrid Logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, pages 821–868. Elsevier, 2007. Cited on pages 127 and 130.
- [AvDFS14] Carlos Areces, Hans van Ditmarsch, Raul Fervari, and François Schwarzentruber. Logics with Copy and Remove. In *Logic, Language, Information, and Computation*, volume 8652 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 2014. Cited on page 128.
- [AvDFS15] Carlos Areces, Hans van Ditmarsch, Raul Fervari, and François Schwarzentruber. The Modal Logic of Copy and Remove. *To Appear in Information and Computation, special issue of WoLLIC 2014*, 2015. Cited on page 128.
- [Baa91] Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. IJCAI*, pages 446–451, 1991. Cited on pages 100 and 103.
- [Bar13] Pablo Barceló. Querying graph databases. In *Proc. PODS*, pages 175–188, 2013. Cited on page 100.
- [BBP18] Pablo Barceló, Gerald Berger, and Andreas Pieris. Containment for rule-based ontology-mediated queries. In *Proc. PODS*, 2018. Cited on page 105.
- [BCM⁺07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Press, 2nd edition, 2007. Cited on pages 15 and 99.
- [BCOŠ14] Meghyn Bienvenu, Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Nested regular path queries in description logics. In *Proc. KR*, 2014. Cited on page 100.

- [BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. Cited on pages 2, 17, 127, and 139.
- [BGO14] Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014. Cited on pages 7 and 19.
- [BHLS17] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. Cited on pages 3, 15, 99, 103, and 125.
- [BHLW16] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First order-rewritability and containment of conjunctive queries in Horn description logics. In *Proc. IJCAI*, pages 965–971, 2016. Cited on pages 55, 57, and 105.
- [BKL⁺16] Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. Inseparability and conservative extensions of description logic ontologies: A survey. In *Proc. of Reasoning Web*, volume 9885 of *LNCS*, pages 27–89. Springer, 2016. Cited on pages 5, 19, 56, and 66.
- [BKR14] Pierre Bourhis, Markus Krötzsch, and Sebastian Rudolph. How to best nest regular path queries. In *Proc. DL*, volume 1193 of *CEUR Workshop Proceedings*, pages 404–415, 2014. Cited on page 101.
- [BKR⁺16] Elena Botoeva, Roman Kontchakov, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. Games for query inseparability of description logic knowledge bases. *Artif. Intell.*, 234:78–119, 2016. Cited on page 56.
- [BL16] Pierre Bourhis and Carsten Lutz. Containment in monadic disjunctive datalog, MMSNP, and expressive description logics. In *Proc. KR*, pages 207–216. AAAI Press, 2016. Cited on pages 105 and 125.
- [BLMS11] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011. Cited on page 99.
- [BLR⁺16] Elena Botoeva, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. Query-based entailment and inseparability for *ALC* ontologies. In *Proc. IJCAI*, pages 1001–1007, 2016. Cited on pages 56, 66, and 87.
- [BLW12] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. Query containment in description logics reconsidered. In *Proc. KR*. AAAI Press, 2012. Cited on page 105.
- [BLW13] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First order-rewritability of atomic queries in Horn description logics. In *Proc. IJCAI*, pages 754–760, 2013. Cited on pages 80 and 81.
- [BM03] Patrick Blackburn and Maarten Marx. Constructive interpolation in hybrid logic. *J. Symb. Log.*, 68(2):463–480, 2003. Cited on page 139.

- [BMRT11] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. IJCAI*, pages 712–717, 2011. Cited on page 99.
- [BO15] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. Reasoning Web*, volume 9203 of *LNCS*, pages 218–307. Springer, 2015. Cited on pages 55, 59, and 99.
- [BOŠ13] Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Conjunctive regular path queries in lightweight description logics. In *Proc. IJCAI*, pages 761–767. IJCAI/AAAI, 2013. Cited on page 101.
- [BOŠ15a] Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Navigational queries based on frontier-guarded datalog: Preliminary results. In *Proc. AMW*, volume 1378 of *CEUR Workshop Proceedings*, 2015. Cited on page 100.
- [BOŠ15b] Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Regular path queries in lightweight description logics: Complexity and algorithms. *J. Artif. Intell. Res.*, 53:315–374, 2015. Cited on page 100.
- [BR15] Meghyn Bienvenu and Riccardo Rosati. Query-based comparison of OBDA specifications. In *Proc. DL*, volume 1350. ceur-ws.org, 2015. Cited on page 56.
- [BS95] Patrick Blackburn and Jerry Seligman. Hybrid Languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. Cited on pages 127 and 131.
- [BtCLW14] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014. Cited on pages 7, 99, 103, and 125.
- [BtCS15] Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, 2015. Cited on pages 4, 7, 15, 19, and 23.
- [BtCV15] Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. Interpolation with decidable fixpoint logics. In *Proc. of LICS*, pages 378–389. IEEE Computer Society, 2015. Cited on page 20.
- [BvB07] Patrick Blackburn and Johan van Benthem. Modal Logic: A Semantic Perspective. In *Handbook of Modal Logic*, pages 1–84. Elsevier, 2007. Cited on page 127.
- [BvBW06] Patrick Blackburn, Johan van Benthem, and Frank Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006. Cited on page 17.
- [Bü60] Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. Cited on page 1.

- [Bü62] Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962. Cited on page 1.
- [CDL⁺07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reas.*, 39(3):385–429, 2007. Cited on page 55.
- [CDL⁺13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 195:335–360, 2013. Cited on page 117.
- [CDLN01] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. In *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier and MIT Press, 2001. Cited on pages 100 and 103.
- [CDLV00] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Proc. KR*, pages 176–185, 2000. Cited on page 100.
- [CEO07] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proc. AAAI*, pages 391–396, 2007. Cited on page 100.
- [CEO09] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In *Proc. IJCAI*, pages 714–720, 2009. Cited on page 100.
- [CEO14] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics via alternating tree-automata. *Inf. Comput.*, 237:12–55, 2014. Cited on page 100.
- [CGK08] Andrea Cali, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. KR*, pages 70–80, 2008. Cited on page 99.
- [Chu36] Alonzo Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, pages 40–41, 1936. Cited on page 2.
- [CK90] Chen Chung Chang and H. Jerome Keisler. *Model Theory*. Elsevier, 3rd edition, 1990. Cited on page 30.
- [CKS81] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981. Cited on page 49.
- [CMW87] Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. A graphical query language supporting recursion. In *Proc. SIGMOD*, pages 323–330, 1987. Cited on page 100.
- [Cod70] Edgar Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, 1970. Cited on page 2.

- [Com69] Stephen D. Comer. Classes without the amalgamation property. *Pacific Journal of Mathematics*, 28:309–318, 1969. Cited on page 23.
- [DGS93] Răzvan Diaconescu, Joseph A. Goguen, and Petros Stefanec. Logical support for modularisation. In Gerard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130, 1993. Cited on page 4.
- [DH00] Giovanna D’Agostino and Marco Hollenberg. Logical questions concerning the μ -calculus: Interpolation, Lyndon and Łoś-Tarski. *J. Symb. Log.*, 65(1):310–332, 2000. Cited on page 22.
- [DL15] Giovanna D’Agostino and Giacomo Lenzi. Bisimulation quantifiers and uniform interpolation for guarded first order logic. *Theor. Comput. Sci.*, 563:75–85, 2015. Cited on page 22.
- [EGOŠ08] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus. Query answering in the description logic Horn-*ALCQI*. In *Proc. JELIA*, volume 5293 of *LNCS*, pages 166–179. Springer, 2008. Cited on pages 56 and 62.
- [EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. of FOCS*, pages 368–377. IEEE Computer Society, 1991. Cited on pages 40 and 78.
- [Elg61] Calvin Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the AMS*, 98:21–52, 1961. Cited on page 1.
- [End01] Herbert Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition, 2001. Cited on pages 2 and 13.
- [EOŠ⁺12] Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-*ALCQI* plus rules. In *Proc. AAAI*, 2012. Cited on page 55.
- [FdRS03] Massimo Franceschet, Maarten de Rijke, and Bernd-Holger Schlingloff. Hybrid logics on linear structures: Expressivity and complexity. In *TIME-ICTL 2003, Cairns, Queensland, Australia*, pages 166–173, 2003. Cited on page 135.
- [Fer14] Raul Fervari. *Relation-Changing Modal Logics*. PhD thesis, Universidad Nacional de Córdoba, Argentina, 2014. Cited on pages 8, 129, 136, 137, and 138.
- [FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979. Cited on page 101.
- [Gab99] Dov Gabbay. *Fibring Logics*. Oxford logic guides. Clarendon Press, 1999. Cited on page 129.
- [Gab08] Dov Gabbay. Introducing reactive Kripke semantics and arc accessibility. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, pages 292–341, 2008. Cited on page 127.

- [Gab13] Dov Gabbay. *Reactive Kripke Semantics*. Cognitive Technologies. Springer, 2013. Cited on page 127.
- [GKP02] Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient algorithms for processing XPath queries. In *Proc. VLDB*, pages 95–106, 2002. Cited on page 101.
- [GKV97] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997. Cited on pages 3 and 21.
- [GKVQ09] Nina Gierasimczuk, Lena Kurzen, and Fernando R. Velázquez-Quesada. Learning and teaching as a game: A sabotage approach. In Xiangdong He, John F. Horty, and Eric Pacuit, editors, *LORI*, volume 5834 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2009. Cited on page 128.
- [GLL09] Stefan Göller, Markus Lohrey, and Carsten Lutz. PDL with intersection and converse: satisfiability and infinite-state model checking. *J. Symb. Log.*, 74(1):279–314, 2009. Cited on pages 101 and 124.
- [GLW06] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I damage my ontology? A case for conservative extensions in description logic. In *Proc. of KR*, pages 187–197. AAAI Press, 2006. Cited on pages 5, 19, 20, 48, and 87.
- [GLWZ06] Silvio Ghilardi, Carsten Lutz, Frank Wolter, and Michael Zakharyashev. Conservative extensions in modal logic. In *Advances in Modal Logic 6*, pages 187–207, 2006. Cited on pages 5 and 19.
- [GM93] Michael J. C. Gordon and Thomas F. Melham, editors. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993. Cited on page 4.
- [GO99] Erich Grädel and Martin Otto. On logics with two variables. *Theoretical Computer Science*, 224(1):73 – 113, 1999. Cited on page 3.
- [GO06] Valentin Goranko and Martin Otto. Model theory of modal logic. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, pages 249–330. Elsevier, 2006. Cited on page 30.
- [GO14] Erich Grädel and Martin Otto. The freedoms of (guarded) bisimulation. In Alexandru Baltag and Sonja Smets, editors, *Johan van Benthem on Logic and Information Dynamics*, pages 3–31. Springer, 2014. Cited on page 30.
- [GOPŠ12] Georg Gottlob, Giorgio Orsi, Andreas Pieris, and Mantas Šimkus. Datalog and its extensions for semantic web databases. In *Proc. Reasoning Web*, volume 7487 of *LNCS*, pages 54–77. Springer, 2012. Cited on page 99.
- [GOR97] Erich Grädel, Martin Otto, and Eric Rosen. Undecidability results on two-variable logics. In *Proc. of STACS*, volume 1200 of *LNCS*, pages 249–260. Springer-Verlag, 1997. Cited on page 3.

- [GP92] Valentin Goranko and Solomon Passy. Using the universal modality: Gains and questions. *Journal of Logic and Computation*, 2(1):5–30, 1992. Cited on page 130.
- [GPT13] Georg Gottlob, Andreas Pieris, and Lidia Tendera. Querying the guarded fragment with transitivity. In *Proc. ICALP II*, volume 7966 of *LNCS*, pages 287–298. Springer, 2013. Cited on pages 100 and 104.
- [GR99] Erich Grädel and Eric Rosen. On preservation theorems for two-variable logic. *Math. Log. Q.*, 45:315–325, 1999. Cited on page 3.
- [Grä89] Erich Grädel. Dominoes and the complexity of subclasses of logical theories. *Ann. Pure Appl. Logic*, 43(1):1–30, 1989. Cited on page 89.
- [Grä99] Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999. Cited on pages 3, 14, 19, 52, and 104.
- [GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. Cited on page 1.
- [GW99] Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *Proc. of LICS*, pages 45–54. IEEE Computer Society, 1999. Cited on pages 52, 76, and 114.
- [HM02] Eva Hoogland and Maarten Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70(3):373–409, 2002. Cited on pages 22 and 23.
- [HMS07] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3), 2007. Cited on page 56.
- [ILS14] Yazmín Ibáñez-García, Carsten Lutz, and Thomas Schneider. Finite model reasoning in Horn description logics. In *Proc. KR*, 2014. Cited on page 56.
- [JLM⁺17] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, Thomas Schneider, and Frank Wolter. Conservative Extensions in Guarded and Two-Variable Fragments. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 108:1–108:14, 2017. Cited on page 11.
- [JLMS17] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, and Thomas Schneider. Query Conservative Extensions in Horn Description Logics with Inverse Roles. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1116–1122, 2017. Cited on page 11.
- [JLMS18] Jean Christoph Jung, Carsten Lutz, Mauricio Martel, and Thomas Schneider. Querying the Unary Negation Fragment with Regular Path Expressions. In *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*, pages 15:1–15:18, 2018. Cited on page 11.

- [Kaz09] Yevgeny Kazakov. Consequence-driven reasoning for Horn-*ALCQI* ontologies. In *Proc. IJCAI*, pages 2040–2045, 2009. Cited on page 56.
- [KG13] Ilianna Kollia and Birte Glimm. Optimizing SPARQL query answering over OWL ontologies. *J. Artif. Intell. Res.*, 48:253–303, 2013. Cited on page 55.
- [Kie06] Emanuel Kieronski. On the complexity of the two-variable guarded fragment with transitive guards. *Inf. Comput.*, 204(11):1663–1703, 2006. Cited on page 52.
- [KL07] Adila Krisnadhi and Carsten Lutz. Data complexity in the \mathcal{EL} family of description logics. In *Proc. LPAR*, volume 4790 of *LNCS*, pages 333–347. Springer, 2007. Cited on page 117.
- [KLWW09] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Formal properties of modularisation. In H. Stuckenschmidt, S. Spaccapietra, and C. Parent, editors, *Modular Ontologies*, volume 5445 of *LNCS*, pages 25–66. Springer, 2009. Cited on pages 4, 20, 23, and 87.
- [KLWW12] Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter. The logical difference for the lightweight description logic \mathcal{EL} . *J. Artif. Intell. Res.*, 44:633–708, 2012. Cited on pages 55 and 56.
- [KPS⁺09] Roman Kontchakov, Luca Pulina, Ulrike Sattler, Thomas Schneider, P. Selmer, Frank Wolter, and Michael Zakharyashev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *Proc. IJCAI*, pages 836–840, 2009. Cited on page 56.
- [KRH07] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for Horn description logics. In *Proc. AAAI*, pages 452–457, 2007. Cited on page 56.
- [KWZ10] Roman Kontchakov, Frank Wolter, and Michael Zakharyashev. Logic-based ontology comparison and module extraction, with an application to DL-Lite. *Artif. Intell.*, 174:1093–1141, 2010. Cited on page 55.
- [KZ14] Roman Kontchakov and Michael Zakharyashev. An introduction to description logics and query rewriting. In *Proc. Reasoning Web*, volume 8714 of *LNCS*, pages 195–244. Springer, 2014. Cited on page 99.
- [LP97] Harry Lewis and Christos Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 2nd edition, 1997. Cited on page 1.
- [Lut08] Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. IJCAR*, volume 5195 of *LNCS*, pages 179–193. Springer, 2008. Cited on pages 100 and 117.
- [LW10] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic \mathcal{EL} . *J. Symb. Comput.*, 45(2):194–228, 2010. Cited on pages 56, 66, 87, and 89.
- [LW11] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. of IJCAI*, pages 989–995. IJCAI/AAAI, 2011. Cited on pages 19, 22, and 32.

- [LW12] Carsten Lutz and Frank Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. KR*, 2012. Cited on pages 56 and 63.
- [LW17] Carsten Lutz and Frank Wolter. The data complexity of description logic ontologies. *Logical Methods in Computer Science*, 13(4), 2017. Cited on page 63.
- [LWW07] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In *Proc. IJCAI*, pages 453–458, 2007. Cited on pages 5, 19, and 87.
- [Mar15] Mauricio Martel. On the Undecidability of Relation-Changing Logics. Master’s thesis, Universidad Nacional de Río Cuarto, Argentina, 2015. Cited on pages 9 and 128.
- [McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521 – 530, 1966. Cited on page 1.
- [Mor75] Michael Mortimer. On languages with two variables. *Math. Log. Q.*, 21(1):135–140, 1975. Cited on pages 3, 14, 19, and 52.
- [MP17] Nico Matentzoglou and Bijan Parsia. BioPortal Snapshot 30 March 2017 (data set), 2017. <http://doi.org/10.5281/zenodo.439510>. Cited on page 57.
- [MSV15] Johannes Marti, Fatemeh Seifan, and Yde Venema. Uniform interpolation for coalgebraic fixpoint logic. In *CALCO*, volume 35 of *LIPICs*, pages 238–252. Schloss Dagstuhl, 2015. Cited on page 22.
- [ORŠ11] Magdalena Ortiz, Sebastian Rudolph, and Mantas Šimkus. Query answering in the Horn fragments of the description logics *SHOIQ* and *SROIQ*. In *Proc. IJCAI*, pages 1039–1044, 2011. Cited on page 100.
- [Pap94] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. Cited on page 1.
- [Pit92] Andrew M. Pitts. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. of Symbolic Logic*, 57, 1992. Cited on page 22.
- [PLC⁺08] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *Journal on Data Semantics*, 10:133–173, 2008. Cited on pages 55 and 99.
- [Pra07] Ian Pratt-Hartmann. Complexity of the guarded two-variable fragment with counting quantifiers. *J. Log. Comput.*, 17(1):133–155, 2007. Cited on page 52.
- [Rab69] Michael Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969. Cited on page 1.
- [Roh06] Philipp Rohde. *On games and logics over dynamically changing structures*. PhD thesis, RWTH Aachen, 2006. Cited on page 139.

- [RRV17] Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular queries on graph databases. *Theory Comput. Syst.*, 61(1):31–83, 2017. Cited on pages [102](#) and [125](#).
- [Sch91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. IJCAI*, pages 466–471, 1991. Cited on page [100](#).
- [Sch03] Philippe Schnoebelen. Oracle circuits for branching-time model checking. In *Proc. ICALP*, volume 2719 of *LNCS*, pages 790–801. Springer, 2003. Cited on page [121](#).
- [Sch07] Thomas Schneider. *The Complexity of Hybrid Logics over Restricted Frame Classes*. PhD thesis, University of Jena, 2007. Cited on page [135](#).
- [Sco62] Dana Scott. A decision method for validity of sentences in two variables. *Journal of Symbolic Logic*, 27:1962, 1962. Cited on pages [14](#) and [19](#).
- [Spa93] Edith Spaan. *Complexity of modal logics*. PhD thesis, ILLC, University of Amsterdam, 1993. Cited on page [130](#).
- [ST04] Wiesław Szwaast and Lidia Tendera. The guarded fragment with transitive guards. *Ann. Pure Appl. Logic*, 128(1-3):227–276, 2004. Cited on pages [100](#) and [104](#).
- [Sto74] Larry Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. Massachusetts Institute of Technology, Project MAC, 1974. Cited on page [2](#).
- [tC05] Balder ten Cate. *Model theory for extended modal languages*. PhD thesis, University of Amsterdam, 2005. ILLC Dissertation Series DS-2005-01. Cited on page [131](#).
- [tCF05] Balder ten Cate and Massimo Franceschet. On the complexity of hybrid logics with binders. volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer Verlag, 2005. Cited on page [135](#).
- [tCS13] Balder ten Cate and Luc Segoufin. Unary negation. *Logical Methods in Computer Science*, 9(3), 2013. Cited on pages [3](#), [7](#), [14](#), [100](#), [101](#), [104](#), [106](#), [116](#), [121](#), and [125](#).
- [Tob01] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001. Cited on page [63](#).
- [TSCS15] Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Sem.*, 33:30–49, 2015. Cited on page [55](#).
- [Tur37] Alan Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2:230–265, 1937. Cited on page [2](#).

- [Var82] Moshe Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 137–146. ACM, 1982. Cited on page 2.
- [Var96] Moshe Y. Vardi. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, pages 149–184, 1996. Cited on page 3.
- [Var98] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *Proc. ICALP*, volume 1443 of *LNCS*, pages 628–641. Springer, 1998. Cited on pages 39, 41, 76, and 79.
- [vB05] Johan van Benthem. An Essay on Sabotage and Obstruction. In *Mechanizing Mathematical Reasoning*, pages 268–276, 2005. Cited on pages 8 and 127.
- [vDvdHK07] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Synthese Library. Springer, 2007. Cited on pages 8 and 127.
- [Vis96] Albert Visser. Uniform interpolation and layered bisimulation. In *Gödel '96 (Brno, 1996)*, volume 6 of *Lecture Notes in Logic*, pages 139–164. Springer, 1996. Cited on pages 20 and 22.
- [Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal μ -calculus. *Bulletin of the Belgian Mathematical Society*, 8(2), 2001. Cited on page 38.
- [WWT⁺14] Kewen Wang, Zhe Wang, Rodney W. Topor, Jeff Z. Pan, and Grigoris Antoniou. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Comput. Intell.*, 30(2):205–232, 2014. Cited on page 55.
- [ZCN⁺15] Yujiao Zhou, Bernardo Cuenca Grau, Yavor Nenov, Mark Kaminski, and Ian Horrocks. PAGOdA: Pay-as-you-go ontology query answering using a Datalog reasoner. *J. Artif. Intell. Res.*, 54:309–367, 2015. Cited on page 55.