Free University of Bozen-Bolzano

PhD Thesis

Temporal Ontology-Mediated Queries

Complexity and Rewritability Checking

Candidate
Anton Gnatenko

Supervisor
Alessandro Artale

2026

Call it nonsense if you like, but I've heard nonsense, compared with which this would be as sensible as a dictionary!

Lewis Carroll, Through the Looking Glass

Acknowledgements

Every student complains about their professor, but I have always been the odd one out. I am deeply grateful to my supervisor Alessandro Artale for his kindness, for his support and trust without judgement, and for tolerating my attempts to discuss "le logiche descrittive" in very bad Italian.

I would also like to thank my collaborators, who have been my advisors throughout. Heartfelt thanks to Vladislav Ryzhikov and Michael Zakharyaschev for their patience and their help in difficult moments. Working with them has given me a model for how to think and write about science, which I can only hope to approach. I am equally grateful to Camille Bourgaux and Michaël Thomazo for hosting me in Paris and for a wonderful collaboration. This thesis has further benefited from the constructive feedback and deep insight of Przemysław Wałęga and Frank Wolter.

Upon arriving in Bolzano, I was fortunate to find true and rare friendship. I cannot overestate how happy I am to have met my dear friends Ale, Andrea, Andrei, and Mahum, and to have shared both joyful and difficult times with them. I was also very lucky to have amazing colleagues: Andrea, Davide, Francesco, Marco, Nicola, Ognjen, Oliver and Tiziano, as well as all the krdbeers, who made the office feel less like a workplace than a home, and who will remain friends beyond its walls. I owe special thanks to my friend Andreas and to the welcoming community of Cortina sulla Strada del Vino, which I have come to call "my village".

Above all, I thank my family for their love and encouragement, and Angie for her love, patience, and care through these four long years.

Abstract

We study the complexity and rewritability of ontology-mediated queries (OMQs), that is, database queries enriched with semantic information. Our focus is on temporal data, where database facts are timestamped and ontologies are specified in description logics or datalog extended with operators of linear temporal logic.

For queries mediated by such ontologies, we analyse the complexity of evaluation in two scenarios: when both the query and the database are given as input, and when the query is fixed while the database varies. These correspond to the combined and data complexity perspectives in query answering.

Related is the problem of rewritability checking—the task of deciding whether an OMQ can be rewritten into a standard database query, so that query answering with respect to an ontology is "packed" into a finite formula executable by a conventional database engine. In this thesis, we study this problem for ontology languages where rewritability is non-trivial, since it is not always guaranteed.

Our results are both positive and negative. For $\mathcal{TEL}^{\circ}$, a temporal extension of the description logic $\mathcal{EL}$ with the operators $\bigcirc/\bigcirc^{-}$ (next/previous time), we prove that OMQ answering is undecidable, but identify two new fragments where it is tractable. The proofs and algorithms rely on a novel correspondence between $\mathcal{TEL}^{\circ}$ ontologies and conjunctive grammars, a generalisation of context-free grammars by the operation of intersection.

To further clarify the expressive power of our ontology languages, we introduce a logic in the temporal *DL-Lite* family, and show that rewritability into standard languages is undecidable precisely due to the interaction between the temporal and description logic components.

Finally, for temporal datalog, we obtain a complete and decidable classification of linear connected queries with $\bigcirc/\bigcirc^{-}$, from which we derive procedures for checking rewritability into various standard languages. In contrast, we prove that rewritability becomes undecidable when operators $\Diamond/\Diamond^{-}$ (once in the future/past) are allowed.

Publications contributing to this thesis

1. Alessandro Artale, Anton Gnatenko, Vladislav Ryzhikov, Michael Zakharyaschev. *A Decidable Temporal DL-Lite Logic with Undecidable First-Order and Datalog-rewritability of Ontology-Mediated Atomic Queries (Extended Abstract).* International Workshop on Description Logics (DL), 2023.
2. Alessandro Artale, Anton Gnatenko, Vladislav Ryzhikov, Michael Zakharyaschev. *On Deciding the Data Complexity of Answering Linear Monadic Datalog Queries with LTL Operators* (*Extended Abstract*). International Workshop on Description Logics (DL), 2024.
3. Alessandro Artale, Anton Gnatenko, Vladislav Ryzhikov, Michael Zakharyaschev. *On Deciding the Data Complexity of Answering Linear Monadic Datalog Queries with LTL Operators.* International Conference on Database Theory (ICDT), 2025.
4. Camille Bourgaux, Anton Gnatenko and Michaël Thomazo. *Analysing Temporal Reasoning in Description Logics Using Formal Grammars* (*Extended Abstract*). International Workshop on Description Logics (DL), 2025. ⋆ Best paper ⋆
5. Camille Bourgaux, Anton Gnatenko and Michaël Thomazo. *Analysing Temporal Reasoning in Description Logics Using Formal Grammars.* European Conference on Artificial Intelligence (ECAI), 2025. ⋆ Outstanding paper ⋆

Publications contributing to a good impression

1. Anton Gnatenko, Oliver Kutz, Nicolas Troquard. *Building an ontology of computational complexity.* Joint Ontology Workshops (JOWO), 2024. ⋆ Best paper ⋆
2. Anton Gnatenko, Oliver Kutz, Nicolas Troquard. *Modelling and Mining Knowledge About Computational Complexity.* International Conference on Knowledge Engineering and Knowledge Management (EKAW), 2024.
3. Anton Gnatenko, Oliver Kutz, Nicolas Troquard. *Ontological Modelling Principles for Computational Complexity.* Applied Ontology, to appear in 2026.

Contents

Contents **vii**

1 Introduction **1**
1.1 Relevance and scope 1
1.2 Temporal logic, sea battles, and data 3
1.3 Ontology-mediated queries 5
1.4 Background 8
1.5 Contributions and outline 12

2 Preliminaries **15**
2.1 Basics 15
2.2 Formal grammars 23
2.3 Databases and query languages 27
2.4 Ontology languages 34
2.5 Temporal conjunctive queries 41
2.6 Ontology-mediated queries 43

3 Reasoning in temporal description logic $\mathcal{TEL}^{\circ}$ **47**
3.1 Overview 47
3.2 The $\mathcal{TEL}^{\circ}$ language 50
3.3 Known results on $\mathcal{TEL}^{\circ}$ 54
3.4 Future $\mathcal{TEL}^{\circ}$ and unary conjunctive grammars . . . 56
3.5 Linear $\mathcal{TEL}^{\circ}$ and context-free grammars 61

3.6 Query answering relative to $\mathcal{TEL}^{\circ}$ TBoxes 64
3.7 Discussion 69

4 A logic with undecidable rewritability problem 71
4.1 Overview 71
4.2 The *TDL-Lite*^{4} language 74
4.3 Deciding first-order rewritability for *DL-Lite*^{4} queries 75
4.4 Query answering relative to *TDL-Lite*^{4} ontologies . 79
4.5 Rewritability of *TDL-Lite*^{4} queries is undecidable . 82
4.6 Discussion 92

5 On rewritability of *datalog*$^{\circ\diamond}$ queries 95
5.1 Overview 95
5.2 Additional preliminaries 100
5.3 Rewritability is undecidable for $datalog^{\diamond}_{lm}$ 102
5.4 Automata-theoretic tools for connected $datalog^{\circ}_{lm}$ . . 106
5.5 Decidability for connected $datalog^{\circ}_{lm}$ 113
5.6 Discussion 117

6 Conclusions 119

A Proofs for Chapter 2 123
A.1 Proofs for Section 2.5 123
A.2 Proofs for Section 2.6 125

B Proofs for Chapter 3 131
B.1 Two definitions of ultimate periodicity 131
B.2 Proofs for Section 3.2 136
B.3 Additional notation and conventions 138
B.4 Proofs for Section 3.4 139
B.5 Proofs for Section 3.5 151
B.6 Proofs for Section 3.6 156

C Proofs for Chapter 4 173
C.1 Proofs for Section 4.3 173
C.2 Proofs for Section 4.4 175

D Proofs for Chapter 5 **179**
D.1 Proofs for Section 5.2 179
D.2 Proofs for Section 5.3 179
D.3 Proofs for Section 5.4 184
D.4 Proofs for Section 5.5 190

Bibliography **199**

Chapter 1

Introduction

1.1 Relevance and scope

During my bachelor studies, I took a course in statistics. "One day", the professor told us, "you will graduate from here. And then all of you will have to work with big data".

Indeed, data is the core resource of the computing industry. The amount of data being produced and stored worldwide is enormous, and enterprises across industries rely heavily on querying this data for analysis.

At the same time, in practice, data is often fragmented or incomplete, limiting its usefulness. The framework of ontology-mediated queries addresses this by introducing a semantic layer that unifies heterogeneous sources and enables inference over missing facts [Xiao et al., 2018]. This approach is gradually making its way to industry: successful pilot projects were conducted with applications to predictive maintenance in Siemens, a German industrial automation company [Kharlamov et al., 2014], and to data integration in Statoil (now Equinor), Norway's leading energy company [Kharlamov et al., 2017]. More recently, Novo Nordisk, a Danish pharmaceuti-

cal company, adopted the ontological approach to standardise and interlink R&D data, improving reuse and query response times in pharmaceutical research [Tan et al., 2025].

Orthogonally to that, the industry faces the challenge of managing data that evolves over time. Many application domains are inherently temporal: clinical data shows patient's conditions and the effects of treatments progressing over time; financial market prices fluctuate; contracts and legal regulations hold during particular periods; observations in scientific experiments are timestamped. In many cases, it is important to store and analyse the whole history of changes in its entirety [Tansel et al., 1993, Snodgrass, 1999]. Data models that account for this, such as dynamic networks, are successfully used for representing a variety of processes and systems, including virus transmission [Husein et al., 2019, Wu et al., 2022], transport networks [Ran and Boyce, 1996], social media [Skyrms and Pemantle, 2000], supply chains [Xu et al., 2019], and power grids [Schäfer et al., 2018].

When one wants to work with heterogeneous and incomplete temporal data, the two challenges reinforce each other: semantic reasoning is usually harder in the temporal setting [Artale et al., 2017].

This thesis focuses on the theoretical foundations of ontology-mediated query answering over temporal data. Our study concerns the computational complexity of key reasoning tasks, in particular the inference of implicit facts by leveraging semantics, and query answering relying both on explicit and inferred information. The results that we present, on the one hand, delineate the limits of applicability of the framework in temporal settings, showing cases where reasoning becomes computationally intractable or even undecidable. On the other hand, we also identify cases where efficient reasoning is possible.

In the remainder of this chapter, Section 1.2 provides a historical account of temporal reasoning, Section 1.3 then examines in more

detail the problems addressed in this thesis, and Section 1.5 outlines our contributions and the structure of the thesis.

1.2 Temporal logic, sea battles, and data

Questions about the logical status of statements about time are much older than modern logic. A classical example is Aristotle's sea battle: if one says today that there will be a sea battle tomorrow, in what sense is this statement true or false already today, before tomorrow has arrived? (See also Lowe [1980]) More generally, the puzzle is whether the truth of statements is fixed by the world as it is now, or affected by how events will in fact unfold. This problem intrigued philosophers for centuries and reappeared in later discussions, including medieval debates on future truth and necessity and early modern accounts such as those of Leibniz [Kretzmann et al., 1982, Murray, 1995]. In the twentieth century, these questions were taken up in a more systematic way within mathematical logic, moving from isolated philosophical examples to formal systems. At the general level, Tarski's work on truth, logical consequence, and model-theoretic semantics provided the technical machinery for studying such systems with mathematical precision: logical formulas could be evaluated in structures, and time itself could be represented as part of that structure [Tarski, 1943, 1983, Etchemendy, 1988]. One early formal system specifically aimed at temporal reasoning was proposed by Łoś [1947], who treated truth as relative to a temporal position: instead of saying simply that a formula is true, one could say that it is true at a given moment or interval of time [Tkaczyk and Jarmużek, 2019]. Around the same period, Łukasiewicz [1988] proposed a many-valued logic in which statements about an open future need not be simply true or false [Woleński, 2019].

A more direct starting point for what is now called temporal logic is the tense logic of Prior [1967], who introduced modal operators for the past and the future and built systematic proof systems around them, using them to formalise arguments about time and to

discuss metaphysical questions about the nature of temporal truth. His work showed that the temporal reading of such operators can be treated with tools similar to those used in modal logic, and that it is possible to reason formally about time without reducing everything to arithmetic on timestamps. Subsequent work refined the semantic side, for example by giving possible worlds semantics and by relating tense operators to more fine-grained notions such as *until* and *since* [Kamp, 1968].

In computer science, temporal logic appeared in a particularly visible way through the work of Pnueli, who proposed linear temporal logic, *LTL*, as a specification language for programs that react to their environment rather than simply compute a function on an input [Pnueli, 1977]. In this view, formulas of temporal logic describe properties of infinite runs, such as that a request is always eventually followed by a response, or that something bad never happens. This line of work led to temporal logics becoming a standard formalism in program verification and model checking, and it also triggered a large body of research on the expressivity and complexity of temporal logics over various classes of structures. For our purposes, an important point is that temporal logics turned out to be rather robust: they support useful specification patterns while still allowing for algorithmic procedures for verification.

This thesis is on ontologies and database theory, in particular on reasoning with temporal data. In that setting, temporal logic has long been viewed as a natural language for talking about changing information: rather than looking only at one database snapshot, one can ask questions about the whole history of the data, for example whether something eventually happens, whether it continues to hold over time, or whether one event is always followed by another [Chomicki and Toman, 1998]. This gave rise to a substantial body of work on temporal query languages and temporal ontology languages, and on their expressive power, evaluation, and translation into more standard database formalisms [Snodgrass, 1987, Chomicki, 1994, Chomicki et al., 2001, Artale and Franconi, 2005, Lutz et al., 2008, Baader et al., 2013]. We therefore turn next to the general

framework of ontology-mediated queries.

1.3 Ontology-mediated queries

An ontology-mediated query is a query accompanied by an *ontology*, a formal description of the domain suitable for automated reasoning. Typically, an ontology introduces a vocabulary of concepts (classes of objects) and relations among them, together with logical constraints on how they may interact. This enables users to formulate queries directly in the ontology's vocabulary while relying on the intended semantics of its terms. The system then computes the answer by reasoning with the ontology [Artale et al., 2009, Xiao et al., 2018].

For example, suppose we are given the following facts:[1]

Andrea is a researcher in 2025.
Andrea is Tiziano's friend in 2025.

They can be represented using relation symbols $\mathsf{Researcher}$ and $\mathsf{friendOf}$, respectively, as

$$\mathsf{Researcher}(\mathrm{Andrea}, 2025) \tag{1.1}$$

$$\mathsf{friendOf}(\mathrm{Andrea}, \mathrm{Tiziano}, 2025) \tag{1.2}$$

where the last argument captures the so-called *timestamp* of the fact.

Further, suppose we would like to have an ontology that encodes the following semantic property of the friendship relation [Deacon and Mercury, 1986]:

Friends will be friends.

We may do it by means of a first-order formula augmented with the temporal operator $\Box$ ("always in the future") of linear temporal logic [Prior, 1967, Pnueli, 1977, Manna and Pnueli, 1991]:

$$\forall x \, \forall y \, . \, \mathsf{friendOf}(x, y) \longrightarrow \Box \, \mathsf{friendOf}(x, y) \tag{1.3}$$

[1] Any resemblance to actual persons is deliberate.

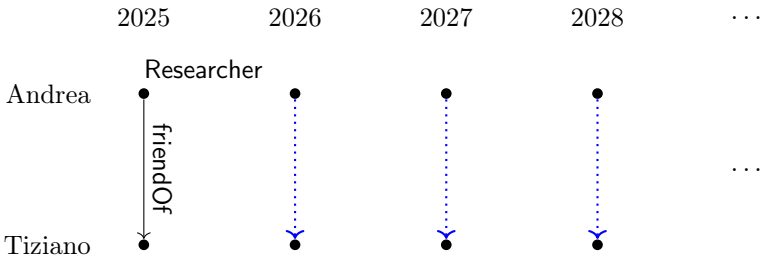


Figure 1.1: Given facts are shown by the node label and the solid arrow, while dotted arrows represent facts that are entailed by the axiom (1.3).

Now, the following queries can be formulated:

Who is a researcher in 2025?
Who is a researcher in 2028?
Who is Tiziano's friend in 2028?

Answering the first query amounts to a simple look-up, while for the second we do not have enough information. The third query is more interesting. Despite having no explicit data on Tiziano's friends in 2028, we conclude from the fact (1.2), the axiom (1.3), and the fact that 2028 is in the future of 2025 that Andrea is still a friend of Tiziano in that year. Figure 1.1 illustrates the matter.

We formulate our ontologies using relational formulas extended with temporal operators similarly to the formula (1.3). The full list of operators we will use is as follows:

$\bigcirc$	(at the next moment)	$\bigcirc^-$	(at the previous moment)
$\Diamond$	(eventually in the future)	$\Diamond^-$	(some time in the past)
$\Box$	(always in the future)	$\Box^-$	(always in the past)

Our temporal databases are finite sets of timestamped relational facts of the form (1.1)-(1.2). An ontology-mediated query $Q = (\mathcal{O}, q)$

is a temporal relational formula q accompanied by an ontology $\mathcal{O}$. Intuitively, answers to Q over a temporal database $\mathcal{D}$ are all the answers to q that are entailed by the facts in $\mathcal{D}$ and the axioms of $\mathcal{O}$.

We will consider two basic scenarios of query answering and two respective measures of computational complexity for this problem.

(*i*) Many queries, many databases. The most general setting is when both the ontology-mediated query and the database are given as input, and the task is to compute the set of answers to the query with respect to the ontology and the database. This framework captures the situation in which queries are posed dynamically and databases are not fixed in advance, so that both parts of the input may vary arbitrarily. The natural complexity measure for this scenario is *combined complexity*, which accounts for the joint contribution of the size of the database, the query, and the ontology. Results in this setting provide upper and lower bounds on the difficulty of ontology-mediated query answering as a computational task.

(*ii*) One query, many databases. In some applications, however, the same ontology-mediated query must be evaluated repeatedly over changing data. The relevant complexity measure in this setting is *data complexity*, which isolates the cost of query answering as a function of the database size, treating the ontology and the query as fixed constants. This viewpoint is especially natural in practice, where the data are typically by far the largest part of the input.

From the data-complexity perspective, a natural question is whether an ontology-mediated query can be rewritten as an equivalent standard SQL query. In that case, the ontology reasoning can be compiled into the query itself and delegated to an existing database

engine, benefiting from its optimisation mechanisms. Although such rewritings are often difficult to compute, scenario (*ii*) makes them particularly attractive, since the cost of rewriting can be amortised over many future evaluations [Gottlob et al., 2011].

This thesis studies combined and data complexity of answering ontology-mediated queries, and also the decidability and complexity of *rewritability checking*, that is, deciding whether a given ontology-mediated query admits a rewriting into a suitable target query language.[2]

1.4 Background

The complexity of ontology-mediated query answering over ordinary, non-temporal data is by now understood in considerable detail. In *datalog*, ontologies are given by Horn rules that iteratively derive new facts from existing ones [Ceri et al., 1989, 1990]. The outcome of this reasoning process is called a (canonical) *model*: a complete picture of the facts that follow from the data and the ontology. Query answering then asks whether a given fact is true in that model. For datalog, this task is ExpTime-complete in combined complexity and P-complete in data complexity. Well-behaved fragments are also understood: linear datalog drops to P-completeness in combined and NL-completeness in data complexity, while the non-recursive fragment captures standard database queries such as select-project-join queries, or, from the theoretical point of view, existential positive first-order formulas [Abiteboul et al., 1995].

This leads to the optimisation question whether a datalog ontology-mediated query can be replaced by an equivalent non-recursive one. In the datalog literature, this property is called *boundedness*: intuitively, there is a uniform, data-independent bound on the number of rule applications needed to compute all possible inferences. In general, however, boundedness is undecidable, even for

[2] In practice, the task is to find a rewriting—but readers with a practical mindset will likely have abandoned the text by this point.

restricted fragments such as linear datalog over binary relations. Decidability reappears only under stronger restrictions, for example in monadic datalog, where all derived predicates are unary [Cosmadakis et al., 1988, van der Meyden, 2000]. Another important restriction is *connectedness*, which requires every rule to describe one connected piece of data rather than several unrelated components. This restriction significantly simplifies the analysis of datalog queries, but by itself it does not restore decidability of boundedness, although its exact effect in low arities, especially over binary schemas, is not fully understood [Hillebrand et al., 1995].

Binary schemas are important because they provide a natural setting for graph-shaped data. *Description logics* (DLs) were designed for this setting: they speak about *concepts*, representing classes of objects, and *roles*, representing binary relations between them [Baader et al., 2010]. Unlike datalog, DLs are interpreted under the open-world assumption, where the database gives only a partial description of reality and the ontology may force the existence of additional, unnamed elements. This changes the shape of models: reasoning no longer just infers new relations between objects, but may have to enlarge the domain as well. At one end, logics such as $\mathcal{ALC}$ and $\mathcal{SROIQ}$ allow rich Boolean structure, role hierarchies, and constraints on the in- and outdegree, at the price of high complexity. At the tractable end, the *DL-Lite* family was designed so that ontology-mediated queries are always first-order rewritable, which is the terminological analogy of boundedness in the DL literature [Artale et al., 2009]. Between these extremes lie the Horn description logics, most notably $\mathcal{EL}$, which retain polynomial-time query answering while allowing substantially richer ontologies than *DL-Lite* [Baader et al., 2005, 2008].

For rewritability, $\mathcal{EL}$ is especially important because it remains close in spirit to datalog. Despite the open-world semantics, $\mathcal{EL}$ ontologies can be translated into datalog programs that preserve query answers. This creates a clear parallel between the two settings, allowing the transfer of results and techniques. First-order rewritability and rewritability to linear datalog for $\mathcal{EL}$-based queries

was shown to be ExpTime-complete [Lutz and Sabellek, 2017, 2022]. For more expressive logics such as $\mathcal{ALC}$, translations to monadic disjunctive datalog and then to constraint satisfaction problems (CSPs) make it possible to study rewritability via the theory of CSP definability. This yields 2NExpTime-completeness of first-order rewritability and a 3NExpTime procedure for datalog rewritability, with a 2NExpTime lower bound [Bienvenu et al., 2014, Feier et al., 2019].

Some of these ideas were later transferred to the temporal setting. On the datalog side, temporal extensions such as $datalog_{1S}$ [Chomicki and Imieliński, 1988], which augments datalog with the successor function on integers, and TEMPLOG [Abadi and Manna, 1989, Baudinet, 1989], which adds Prior-style temporal operators and is expressively equivalent to $datalog_{1S}$, were introduced to capture reasoning over both relations and time. More recently, $datalog_{MTL}$ extended this line of work to facts interpreted over intervals [Wałęga et al., 2019, 2020]. In all these formalisms, one again asks for the model generated by repeated rule application, except that facts are now distributed over time. In the closed-world setting of datalog, however, the temporal dimension remains relatively tame: data complexity rises to PSpace-completeness, while combined complexity, for $datalog_{1S}$ and TEMPLOG, stays at the level of ExpTime. What is missing, however, is the analogue of the boundedness theory, which was not systematically studied for these temporal datalog languages.

What was studied instead is a notable model-theoretic property, namely *ultimate periodicity* [Chomicki and Imieliński, 1988]. Intuitively, a temporal model is ultimately periodic if, after some finite initial segment, the pattern of derived facts repeats with a fixed period. This still allows infinitely many time points, but ensures that the infinite set of consequences admits a finite representation. For $datalog_{1S}$, and therefore also for TEMPLOG, ultimate periodicity was established as a general property of models under the closed-world semantics. This is not an analogue of boundedness in a direct sense, but it does provide a useful regularity condition for the analysis of

queries through automata-theoretic techniques.

Temporal description logics bring the same questions into the open-world setting, where models are much less constrained. Existing results mainly concern lightweight temporal DLs, especially variants close to *DL-Lite*, and identify fragments for which low data complexity is guaranteed, often via rewritability into temporal first-order query languages [Artale et al., 2014, 2021]. More recent work studies rewritability as a decision problem, for instance for pure linear temporal logic and for temporal *DL-Lite* ontologies that admit suitable projections to LTL. There, ultimately periodic models again play an important technical role, with automata and algebraic methods for regular languages as the main tools [Kurucz et al., 2023, Artale et al., 2022]. These results, however, remain confined to rather restricted fragments close to *DL-Lite*.

At the other end, temporal extensions of expressive DLs such as $\mathcal{ALC}$ quickly become computationally intractable, and in many cases even undecidable, because they can encode complex temporal structures such as Turing machine computations or tiling problems [Artale and Franconi, 2005]. In that setting, the CSP-based route that is so useful for rewritability questions in the atemporal case is far less developed, and at present it does not provide a comparable approach to rewritability checking. A natural intermediate case is temporal $\mathcal{EL}$, since in the atemporal setting $\mathcal{EL}$ occupies exactly the point where expressive power and tractable reasoning remain well balanced. Here a key result of Gutiérrez-Basulto et al. [2016b] established a close connection between temporal $\mathcal{EL}$ ontologies and $datalog_{1S}$, echoing the atemporal connection between $\mathcal{EL}$ and datalog. The role of ultimate periodicity is much more delicate, however: unlike in temporal datalog under closed-world semantics, it was not known whether the relevant temporal $\mathcal{EL}$ models are always ultimately periodic, and the translation to $datalog_{1S}$ was obtained only under that assumption. Thus, the parallel with the atemporal $\mathcal{EL}$ -datalog connection remained conditional. Moreover, even when such a translation is available, the lack of rewritability results for temporal datalog limits the transfer of techniques and results.

Taken together, these results leave a fairly clear picture. In the atemporal setting, the connection between datalog and $\mathcal{EL}$ supports both complexity analysis and rewritability theory. In the temporal setting, the datalog side provides robust complexity bounds and a general periodicity property, while the DL side offers rewritability results mainly for fragments close to *DL-Lite*. For temporal $\mathcal{EL}$, the connection to temporal datalog is known only under an additional periodicity assumption, and there is still no corresponding rewritability theory on the temporal datalog side to match the atemporal one.

1.5 Contributions and outline

Our main concern is how combining the relational and temporal dimensions within a single ontology language changes the computational behaviour of ontology-mediated queries and the prospects for their rewritability. Upon providing the technical background in Chapter 2, we investigate in the next three chapters the theory of query answering and query rewriting for temporal extensions of $\mathcal{EL}$, *DL-Lite*, and datalog.

Chapter 3: Temporal $\mathcal{EL}$. We start with $\mathcal{TEL}^{\circ}$, a temporal extension of the description logic $\mathcal{EL}$ with operators $\bigcirc/\bigcirc^{-}$. This ontology language was introduced by Gutiérrez-Basulto et al. [2016a], who left open whether query answering is decidable. To answer this question, we establish a correspondence between (fragments of) $\mathcal{TEL}^{\circ}$ and some specific kinds of formal grammars, in particular, conjunctive grammars (context-free grammars equipped with the operation of intersection). This connection implies that $\mathcal{TEL}^{\circ}$ does not possess the property of ultimate periodicity of models, and further leads to undecidability of query answering in $\mathcal{TEL}^{\circ}$, thus closing the original question negatively. On the other hand, the same link with grammars allows to establish decidability of query answering for some new interesting fragments of $\mathcal{TEL}^{\circ}$, and to reuse for this purpose

existing tools and algorithms known for conjunctive grammars.

Chapter 4: Temporal *DL-Lite*. The results on $\mathcal{TEL}^{\circ}$ suggest that even a "minimalist" two-dimensional language can be expressive enough to make basic reasoning undecidable. On the other hand, decidability for other languages known from literature usually relies on strong syntactic restrictions which make the reasoning almost one-dimensional. Our interest in this chapter lies in finding an example of a language where basic reasoning is decidable, but where the interaction of the two dimensions is strong enough to render it difficult. We design a non-Horn logic in the temporal *DL-Lite* family that allows a very limited form of temporal reasoning with binary relations. We show that answering ontology-mediated atomic queries in this logic can be done in polynomial space, but determining whether it is rewritable in some standard query languages for temporal databases is undecidable. On the other hand, we show (via a translation into monadic disjunctive datalog) that the same problem is decidable when the temporal operators are removed from the logic, highlighting that the undecidability result is a product of two dimensions.

Chapter 5: Temporal datalog. We then restrict our attention to Horn logics, and consider (fragments of) TEMPLOG. We obtain a complete and decidable classification of linear connected queries that use operators $\bigcirc/\bigcirc^{-}$ and derive complexity bounds for rewritability checking to various query languages. In contrast, we prove undecidability of such rewritability when queries are allowed to use the eventuality operators $\Diamond/\Diamond^{-}$, essentially reusing the ideas of Chapter 4.

Chapter 3 is based on joint work with Camille Bourgaux and Michaël Thomazo [Bourgaux et al., 2025], while Chapters 4 and 5 draw on collaborations with Alessandro Artale, Vladislav Ryzhikov, and Michael Zakharyaschev [Artale et al., 2023, 2025]. Discussions

of related work appear in the introductory sections of the individual chapters, and the results are summarised in their concluding sections, followed by a general overview of our contributions and directions for future work in the final Chapter 6.

An important aspect of our results is how we measure input size, which in turn depends on how timestamps are represented. Most earlier work on temporal reasoning assumes a unary encoding of integers. Under this assumption, a temporal database is viewed as a sequence of ordinary database snapshots—one for each moment in time, without gaps—making automata-theoretic and related methods particularly natural to apply. Our results adopt this standard assumption, which does not hinder a meaningful theoretical comparison of ontology languages. Recently, however, a more practical setting of binary encoding has received considerable attention, e.g. in [Furia and Spoletini, 2014, Brandt et al., 2018, Wałęga et al., 2019]. In some cases, encoding integers in binary (and thus exponentially more succinctly) merely causes the corresponding exponential increase in the complexity of reasoning, for example for bounded metric temporal logic [Bouyer et al., 2008]. In other cases, reasoning with binary-encoded numbers can in fact yield procedures that are practically more efficient than their unary counterparts [Wałęga et al., 2020]. We briefly return to this issue when discussing future research directions in Chapter 6.

Beyond the intuitive ideas, most of our formal proofs rely on induction and extensive case analyses. To keep the exposition clear, we present examples, proof sketches, and some short proofs in the main text, while longer formal arguments are deferred to the appendix. Bibliographic references and cross-references to chapters, sections, theorems, and lemmas are all clickable.[3]

[3]That is, in the digital version; printed copies are generally less responsive.

Chapter 2

Preliminaries

We introduce in this chapter the notions and results—both classical and recent—that will be used in the subsequent chapters. In Section 2.1, we cover the basic notions of theoretical computer science: formal languages, machines, and complexity classes, and in Section 2.2—formal grammars. In particular, we define and discuss conjunctive grammars that we will work with in Chapter 3. Then, we introduce temporal databases and query languages, in Section 2.3, and logics for reasoning about temporal data, in Section 2.4. Finally, we cover temporal conjunctive queries and ontology-mediated queries, in Sections 2.5 and 2.6, respectively.

As is usual for such material, the exposition is formal and dry; the reader is advised to rehydrate regularly.

2.1 Basics

Notation. We use the standard denotations $\mathbb{Z}$ and $\mathbb{N}$ for sets of integers resp. non-negative integers. Letters m, n, k, ℓ always stand for integers. In general, sets are denoted with uppercase letters and their elements with lowercase letters. Given a set A, with A^n

we denote the nth Cartesian product of A, defined inductively as $A^1 = A$, and $A^n = A^{n-1} \times A$ for $n > 1$. Elements of A^n are *tuples* of length n. The symbol $\overline{a}$ stands for a *tuple* $(a_1, \ldots, a_n)$, and $|\overline{a}| = n$ is its length. Tuples of numbers are usually called *vectors*. We also consider the empty tuple () of length 0, and $A^0 = \{()\}$. Sometimes, it is convenient to associate a tuple $\overline{a}$ with the set $\{a_1, \ldots, a_n\}$. Furthermore, if f is a function $A \to B$ and $\overline{a} \in A^n$, we write $f(\overline{a})$ for the tuple $(f(a_1), \ldots, f(a_n)) \in B^n$.

Formal languages. Any finite set $\mathfrak{A}$ is called an *alphabet*. A sequence $w = a_1 a_2 \ldots a_n$, where each $a_i \in \mathfrak{A}$, is a *word* over $\mathfrak{A}$; its length $|w|$ is n. The empty sequence is denoted by ε, with $|\varepsilon| = 0$. We write $\mathfrak{A}^*$ for the set of all words over $\mathfrak{A}$, including ε. A *language* over $\mathfrak{A}$ is any subset $L \subseteq \mathfrak{A}^*$. For words u and v, their concatenation is denoted by uv.

Finite automata. A *finite automaton* over an alphabet $\mathfrak{A}$ is a quintuple $\mathcal{A} = (S, \mathfrak{A}, s_0, T, F)$, where:

- S is a finite set of *states*,
- $s_0 \in S$ is the *initial state*,
- $T \subseteq S \times \mathfrak{A} \times S$ is the *transition relation*,
- $F \subseteq S$ is the set of *accepting states*.

The size of $\mathcal{A}$ is $|\mathcal{A}| = |S| + |T|$.

A *run* of $\mathcal{A}$ on a word $w = a_1 a_2 \ldots a_n \in \mathfrak{A}^*$ is a sequence of states $\rho = s_0 s_1 \ldots s_n$ such that $(s_{i-1}, a_i, s_i) \in T$ for every $i \in 1, \ldots, n$. A run ρ is *accepting* if its final state s_n lies in F. A word $w \in \mathfrak{A}^*$ is *accepted* by $\mathcal{A}$ if $\mathcal{A}$ admits an accepting run on w. The *language* recognised by $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of all accepted words:

$$L(\mathcal{A}) = \{w \in \mathfrak{A}^* \mid \mathcal{A} \text{ has an accepting run on } w\}.$$

An automaton is called *deterministic* if its transition relation T is a total function $S \times \mathfrak{A} \to S$. Every nondeterministic automaton can be transformed into a deterministic one that recognises the same language. This is done by the subset construction [Rabin and Scott, 1959], which may incur an exponential blow-up in the number of states, and this bound is tight.

Turing machines and decidability. A *Turing machine* is a sextuple $M = (S, \mathfrak{A}, \mathfrak{B}, s_0, s_h, \Delta)$, where:

- S is a finite set of states,
- $\mathfrak{A}$ is an input alphabet,
- $\mathfrak{B} \supseteq \mathfrak{A} \cup \{0, 1\}$ is a tape alphabet,
- $s_0, s_h \in S$ are the initial and halting states,
- $\Delta \subseteq S \times \mathfrak{B} \times S \times \mathfrak{B} \times \{L, R\}$ is a transition relation.

We assume that there is no transition $(s, \beta, s', \beta', D) \in \Delta$ for $s = s_h$. The machine is *deterministic* if Δ is a partial function $S \times \mathfrak{B} \to S \times \mathfrak{B} \times \{L, R\}$, and *nondeterministic* otherwise.

Fix a *blank symbol* $\sqcup \in \mathfrak{B} \setminus \mathfrak{A}$. A *configuration* of M is a triple (s, t, i), where $s \in S$ is the current state, $t \colon \mathbb{Z} \to \mathfrak{B}$ is the tape contents, and $i \in \mathbb{Z}$ is the head position. For an input word $w = a_0 \dots a_{n-1} \in \mathfrak{A}^*$, the *initial configuration* of M on w is $(s_0, t_w, 0)$, where

$$t_w(j) = \begin{cases} a_j, & 0 \leqslant j < n, \\ \sqcup, & \text{otherwise.} \end{cases}$$

A *computation* of M on w is a sequence of configurations

$$(s_0, t_0, i_0), (s_1, t_1, i_1), \dots, (s_m, t_m, i_m) \tag{2.1}$$

such that (s_0, t_0, i_0) is the initial configuration on w and, for each $\ell < m$, if

$$(s_\ell, t_\ell(i_\ell), s_{\ell+1}, b_\ell, D_\ell) \in \Delta,$$

then $t_{\ell+1}$ agrees with t_ℓ at every position except possibly i_ℓ, where $t_{\ell+1}(i_\ell) = b_\ell$, and

$$i_{\ell+1} = \begin{cases} i_\ell - 1, & D_\ell = L, \\ i_\ell + 1, & D_\ell = R. \end{cases}$$

The number m is the *length* of the computation. A computation is *halting* if $s_m = s_h$.

Every deterministic Turing machine M computes a partial function $F_M \colon \mathfrak{A}^* \to \mathfrak{B}^*$. Namely, for $w \in \mathfrak{A}^*$, we define $F_M(w) = u$ if M has a (unique) halting computation on w which ends in a configuration (s_h, t, i) with $i = 0$, and there exists a word $u = b_0 \dots b_{r-1} \in \mathfrak{B}^*$ such that

$$t(j) = \begin{cases} b_j, & 0 \leqslant j < r, \\ \sqcup, & \text{otherwise.} \end{cases}$$

If no such halting computation exists, then $F_M(w)$ is undefined.

For a language $L \subseteq \mathfrak{A}^*$, let $\chi_L \colon \mathfrak{A}^* \to \{0, 1\}$ be its characteristic function. We say that a deterministic Turing machine M *decides* L if $F_M = \chi_L$. In that case we also write

$$L(M) = \{w \in \mathfrak{A}^* \mid F_M(w) = 1\}.$$

We will use nondeterministic Turing machines only for deciding languages. In that setting, a nondeterministic machine *accepts* an input word w if it has a halting computation on w whose output is 1. It *decides* a language $L \subseteq \mathfrak{A}^*$ if it accepts exactly the words in L and *halts* on every input word w, i.e. there exists $m_0 \in \mathbb{N}$ such that every computation of M on w has length at most m_0.

For languages, one can equivalently work with two halting states s_a and s_r instead of a single halting state and a binary output: entering s_a corresponds to output 1, and entering s_r to output 0.

Boolean circuits. Let B be a set of Boolean functions. A *Boolean circuit* over B with n inputs and k outputs is a directed acyclic graph

whose nodes are called *gates*, with n designated *input gates* labelled by the variables $x_1, \ldots, x_n$ and k designated and ordered *output gates*. Each non-input gate v is labelled by a function $f_v \in B$ of some arity $r \geqslant 0$, and has exactly r incoming edges.

Given an input $\overline{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$, the value of each gate is defined inductively:

- an input gate labelled by x_i has value a_i,
- if a non-input gate v is labelled by a function f_v and its predecessors have values $b_1, \ldots, b_r$, then v has value $f_v(b_1, \ldots, b_r)$.

The circuit computes the function

$$C\colon \{0,1\}^n \to \{0,1\}^k,$$

whose value on input $\overline{a}$ is the tuple of values of the output gates.

The *standard basis* is $\{\wedge, \vee, \neg\}$. We will also consider, for each $m \geqslant 2$, the modulo-m gates MOD_m, defined by

$$\mathrm{MOD}_m(b_1, \ldots, b_r) = 1 \iff \sum_{i=1}^{r} b_i \equiv 0 \pmod{m}.$$

A family $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits with one output gate *decides* a language $L \subseteq \{0,1\}^*$ if it computes its characteristic function. This is extended to languages over arbitrary finite alphabets via a suitable binary encoding.

Complexity. For a function $f\colon \mathbb{N} \to \mathbb{N}$, a Turing machine M *runs in time* f if there is $c \in \mathbb{N}$ such that on every input word of length n, any of its computations has length at most $c \cdot f(n)$. It *uses space* f if $0 \leqslant i < c \cdot f(n)$ in every configuration (s, t, i) of its computations on every input word of length n.

We denote by $\mathrm{DTime}(f)$ and $\mathrm{DSpace}(f)$ the classes of languages decidable by deterministic Turing machines in time resp. space f. Their nondeterministic analogues are denoted by $\mathrm{NTime}(f)$ and $\mathrm{NSpace}(f)$.

Further, let $\exp(x, 0) = x$ and $\exp(x, m+1) = 2^{\exp(x,m)}$ for all $m \in \mathbb{N}$. The classes used in this thesis are:

$$\begin{aligned}
\mathrm{P} &= \bigcup_{k\in\mathbb{N}} \mathrm{DTime}(n^k),\\
\mathrm{PSpace} &= \bigcup_{k\in\mathbb{N}} \mathrm{DSpace}(n^k),\\
\mathrm{ExpTime} &= \bigcup_{k\in\mathbb{N}} \mathrm{DTime}(2^{n^k}),\\
m\text{-}\mathrm{ExpTime} &= \bigcup_{k\in\mathbb{N}} \mathrm{DTime}(\exp(n^k, m)),\\
m\text{-}\mathrm{ExpSpace} &= \bigcup_{k\in\mathbb{N}} \mathrm{DSpace}(\exp(n^k, m)),\\
\mathrm{L} &= \bigcup_{c\in\mathbb{N}} \mathrm{DSpace}(c \log n),\\
\mathrm{NP} &= \bigcup_{k\in\mathbb{N}} \mathrm{NTime}(n^k),\\
m\text{-}\mathrm{NExpTime} &= \bigcup_{k\in\mathbb{N}} \mathrm{NTime}(\exp(n^k, m)),\\
\mathrm{NL} &= \bigcup_{c\in\mathbb{N}} \mathrm{NSpace}(c \log n).
\end{aligned}$$

We also use coNP, the class of languages whose complements lie in NP. By the theorem of Savitch [1970], nondeterministic polynomial space coincides with polynomial space, and more generally m-nondeterministic exponential space coincides with m-exponential space.

We also use circuit complexity classes. Let C be a Boolean circuit. Its *size* is the number of gates of C, its *depth* is the length of a longest directed path from an input gate to an output gate, and its *fan-in* is the maximum number of incoming edges of any gate.

A family of Boolean circuits $\{C_n\}_{n\in\mathbb{N}}$, where C_n has n inputs, has *polynomial size* if there exist $c, k \in \mathbb{N}$ such that C_n has size at

most $c \cdot n^k$ for all n. It has *constant depth* if there exists $d \in \mathbb{N}$ such that C_n has depth at most d for all n. It has *logarithmic depth* if there exists $c \in \mathbb{N}$ such that C_n has depth at most $c \cdot \log n$ for all n. It has *fan-in* r if every gate in every C_n has fan-in at most r.

Then, the class AC^0 consists of all languages decided by polynomial-size, constant-depth Boolean circuit families over the standard basis, with unbounded fan-in $\wedge$- and $\vee$-gates. The class ACC^0 is defined analogously, with the addition of unbounded fan-in MOD_m gates for all $m \geqslant 2$. Finally, NC^1 consists of all languages decided by polynomial-size, logarithmic-depth Boolean circuit families over the standard basis with fan-in 2. The following inclusions hold:

$$\mathrm{AC}^0 \subsetneq \mathrm{ACC}^0 \subseteq \mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \\ \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSpace} \subseteq \mathrm{ExpTime}. \quad (2.2)$$

All these inclusions are conjectured to be strict, though this has been proved only for the first one [Razborov, 1987, Smolensky, 1987]. We also have

$$m\text{-ExpTime} \subseteq m\text{-ExpSpace}$$

and

$$m\text{-ExpTime} \subsetneq (m+1)\text{-ExpTime},$$
$$m\text{-ExpSpace} \subsetneq (m+1)\text{-ExpSpace}$$

for all m.

Reductions and hardness. A *many-one reduction* from a language $L \subseteq \mathfrak{A}^*$ to a language $L' \subseteq \mathfrak{A}^*$ is a function $f\colon \mathfrak{A}^* \to \mathfrak{A}^*$ such that, for every $w \in \mathfrak{A}^*$, $w \in L \iff f(w) \in L'$.

Such a function is *computable in polynomial time* if there exists a deterministic Turing machine M with $F_M = f$ that runs in time p for some polynomial p. Analogously, f is *computable in logarithmic space* if there exists a machine M with $F_M = f$ that uses space $c \log n$ on every input word of length n, for some constant c.

Let $\mathcal{C}$ be a complexity class. A language L is *$\mathcal{C}$-hard* with respect to polynomial-time (logarithmic-space, constant-depth) reductions, if for every language $L' \in \mathcal{C}$ there is a reduction from L' to L computable in polynomial time (resp. in logarithmic space or by a constant-depth, polynomial-size Boolean circuit family). It is *$\mathcal{C}$-complete* if it is $\mathcal{C}$-hard and belongs to $\mathcal{C}$.

In what follows, we will understand NC^1- and L-hardness with respect to constant-depth circuit reductions; NL- and P-hardness with respect to logarithmic-space reductions; and hardness for larger classes with respect to polynomial-time reductions.

Decision problems. A *decision problem* is specified by giving a class of input objects and a property to be tested on those objects. Formally, this corresponds, under a suitable encoding, to a language $L \subseteq \mathfrak{A}^*$ consisting of all words that encode objects with the required property. Not every word in $\mathfrak{A}^*$ must be a valid encoding, but such details are usually omitted when they are clear from the context. The notions of decidability, hardness and completeness are thus transferred from languages to decision problems.

Graph 3-colourability is the following decision problem. Given an undirected graph $G = (V, E)$, decide whether there exists a map $c\colon V \to \{1, 2, 3\}$ such that $c(u) \neq c(v)$ for every edge $\{u, v\} \in E$. This problem is NP-complete [Arora and Barak, 2009]. The lower bounds already hold for connected graphs.

Counter machines. A *k-counter machine* (or *Minsky machine*) is a quadruple $M = (S, k, s_0, \delta)$, where:

- S is a finite set of states,
- k is the number of counters,
- $s_0 \in S$ is the initial state,

- $\delta\colon S\times\{0,1\}^k \to S\times\{-1,0,1\}^k$ is a partial transition function such that, if $\delta(s,\overline{p}\,) = (s',\overline{\varepsilon}\,)$, then $p_i = 0$ implies $\varepsilon_i \geqslant 0$ for all $i \in \{1,\dots,k\}$.

A *configuration* of M is a pair $(s,\overline{c}\,)$ with $s \in S$ and $\overline{c} \in \mathbb{N}^k$. Define $\mathrm{sgn}\colon \mathbb{N} \to \{0,1\}$ by

$$\mathrm{sgn}(n) = \begin{cases} 0, & n = 0, \\ 1, & n > 0. \end{cases} \tag{2.3}$$

A *computation* of M on input $\overline{c}^{\,0}$ is a sequence of configurations

$$(s_0,\overline{c}^{\,0}),(s_1,\overline{c}^{\,1}),\dots,(s_m,\overline{c}^{\,m}) \tag{2.4}$$

such that for each $i < m$ we have $\delta(s_i,\mathrm{sgn}(\overline{c}^{\,i})) = (s_{i+1},\overline{\varepsilon}^{\,i})$ and $\overline{c}^{\,i+1} = \overline{c}^{\,i} + \overline{\varepsilon}^{\,i}$. The number m is the *length* of the computation.

We say that M *halts* on input $\overline{c}^{\,0}$ if there exists a computation (2.4) ending in a configuration $(s_m,\overline{c}^{\,m})$ with $\delta(s_m,\mathrm{sgn}(\overline{c}^{\,m}))$ undefined. In that case, $\overline{c}^{\,m}$ is the output of M on $\overline{c}^{\,0}$. We say that M *halts* if it halts on input $\overline{0} = (0,\dots,0)$.

Theorem 2.1 (Minsky [1967])**.** *It is undecidable whether a given* 2*-counter machine halts.*

2.2 Formal grammars

We introduce conjunctive grammars [Okhotin, 2001, 2013], and their subclasses of context-free and regular grammars.

Syntax. A *conjunctive grammar* is a quadruple $G = (N,\mathfrak{A},\mathcal{S},R)$, where N and $\mathfrak{A}$ are disjoint alphabets of *nonterminals* and *terminals*, respectively, $\mathcal{S} \in N$ is a distinguished *start symbol*, and R is a finite set of *rules* of the form:

$$\mathcal{N} \quad\to\quad \alpha_1 \;\&\; \dots \;\&\; \alpha_n \tag{2.5}$$

with $\mathcal{N} \in N$, $n \geqslant 1$, and $\alpha_i \in (N \cup \mathfrak{A})^*$. Each α_i is called a *conjunct*. If a grammar has a unique conjunct in every rule, then it is a *context-free grammar*, and if further this conjunct has form either ε or $c\mathcal{N}'$, for $c \in \mathfrak{A}$, $\mathcal{N}' \in N$, then it is a *regular grammar* [Chomsky, 1956, Hopcroft et al., 2006]. When the start symbol is not specified, we write $G = (N, \mathfrak{A}, R)$. Several rules with the same left-hand side $\mathcal{N}$ can also be written as a single rule (with | used to separate the right-hand sides):

$$\mathcal{N} \quad \to \quad \alpha_1^1 \;\&\; \ldots \;\&\; \alpha_{n_1}^1 \mid \cdots \mid \alpha_1^m \;\&\; \ldots \;\&\; \alpha_{n_m}^m$$

The *size* $|G|$ of G is the number of symbols needed to write it down.

Semantics. Intuitively, the semantics of conjunctive grammars extends that of context-free grammars with *intersection*: given rule (2.5), apply "in parallel" context-free rules $\mathcal{N} \to \alpha_i$ and take the intersection of the generated languages. Formally, for every $X \in N \cup \mathfrak{A}$ and every word $w \in \mathfrak{A}^*$, let $X(w)$ denote the proposition "w has property X". A *derivation* in G is a finite sequence $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_m$ such that $\mathcal{G}_0 = \{c(c) \mid c \in \mathfrak{A}\}$, and, for each $i \in \{0, \ldots, m-1\}$, the set $\mathcal{G}_{i+1}$ is obtained from $\mathcal{G}_i$ by applying the following *deduction rule*: add a single fact $\mathcal{N}(w)$ such that there exists a grammar rule

$$\mathcal{N} \to \alpha_1 \;\&\; \cdots \;\&\; \alpha_n$$

in R, where each α_i is of the form

$$\alpha_i = X_1^i \ldots X_{k_i}^i \qquad (k_i \geqslant 0,\ X_j^i \in N \cup \mathfrak{A}),$$

and there exist words $u_j^i \in \mathfrak{A}^*$ such that

$$u_1^1 \ldots u_{k_1}^1 = \cdots = u_1^n \ldots u_{k_n}^n = w,$$

while

$$X_1^i(u_1^i), \ldots, X_{k_i}^i(u_{k_i}^i) \in \mathcal{G}_i \qquad \text{for every } i \in \{1, \ldots, n\}.$$

For $X \in N \cup \mathfrak{A}$ and $w \in \mathfrak{A}^*$, we write $G \vdash X(w)$ if there exists a derivation $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_m$ with $X(w) \in \mathcal{G}_m$.

For each $\mathcal{N} \in N$, define

$$L_G(\mathcal{N}) = \{w \in \mathfrak{A}^* \mid G \vdash \mathcal{N}(w)\},$$

and the *language* of G by $L(G) = L_G(\mathcal{S})$.

We call a language $L \subseteq \mathfrak{A}^*$ *conjunctive*, (*context-free*, *regular*) if $L = L(G)$ for a conjunctive (respectively, context-free or regular) grammar G. A language is regular if and only if it is recognised by a finite-state automaton [Arora and Barak, 2009].

Example 2.2 (Okhotin [2013])**.** The language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$, which is not context-free [Hopcroft et al., 2006], is generated by $G = (\{\mathcal{S}, \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}, \{a, b, c\}, \mathcal{S}, R)$ where R contains:

$$\begin{array}{lcl} \mathcal{S} & \to & \mathcal{A}\mathcal{B} \,\&\, \mathcal{D}\mathcal{C} \\ \mathcal{A} & \to & a\mathcal{A} \mid \varepsilon \\ \mathcal{C} & \to & c\mathcal{C} \mid \varepsilon \end{array} \qquad \begin{array}{lcl} \mathcal{B} & \to & b\mathcal{B}c \mid \varepsilon \\ \mathcal{D} & \to & a\mathcal{D}b \mid \varepsilon \end{array}$$

Indeed, from $\mathcal{A}\mathcal{B}$ one obtains $\{a^m b^n c^n \mid m, n \in \mathbb{N}\}$, and from $\mathcal{D}\mathcal{C}$ one gets $\{a^k b^k c^l \mid k, l \in \mathbb{N}\}$. The intersection filters out 'unbalanced' words, forcing $m = n = k = l$. ⊣

The membership problem for conjunctive grammars is P-complete.

Theorem 2.3 (Okhotin [2003, 2011])**.** *Checking whether $w \in L(G)$, for a given conjunctive grammar G and $w \in \mathfrak{A}^*$, is* P*-complete.*

Parikh images. A set $\mathcal{L} \subseteq \mathbb{Z}^n$ is *linear* if

$$\mathcal{L} = \{\overline{b} + k_1\overline{p}_1 + \cdots + k_l\overline{p}_l \mid k_1, \ldots, k_l \in \mathbb{N}\}$$

for some $\overline{b} \in \mathbb{Z}^n$, called *offset*, and $\overline{p}_1, \ldots, \overline{p}_l \in \mathbb{Z}^n$, called *periods*. A *semilinear* set is a union of finitely many linear sets:

$$S = \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_m$$

Given an alphabet $\mathfrak{A} = \{c_1, \ldots, c_n\}$ enumerated in a fixed order, let $\#c_i(w)$ denote the number of occurrences of c_i in $w \in \mathfrak{A}^*$. The *Parikh image* $p(w)$ of a word w is a vector $\overline{u} \in \mathbb{Z}^n$ such that $u_i = \#c_i(w)$, for all $1 \leqslant i \leqslant n$. The Parikh image $p(L)$ of a language L is the set $\{p(w) \mid w \in L\}$. It is easy to see that when L is regular, $p(L)$ is semilinear. A deeper result is the following.

Theorem 2.4 (Parikh [1966])**.** *If L is context-free, then $p(L)$ is semilinear. Moreover, for any semilinear $S \subseteq \mathbb{N}^n$ there exists a regular language L' such that $S = p(L')$.*

A unary language L can be seen as a set of numbers given in unary that coincides with its Parikh image: $c^n \in L$ if and only if $n \in p(L)$. Theorem 2.4 implies that a unary language is regular iff its Parikh image is semilinear, and thus unary context-free languages are regular. However, unary conjunctive languages may not be regular.

Example 2.5 (Jeż [2008])**.** For the following grammar G, the language $L_G(\mathcal{N}_1) = \{c^{4^n} \mid n \in \mathbb{N}\}$ is not regular:

$$\begin{array}{lcl} \mathcal{N}_1 & \to & \mathcal{N}_1\mathcal{N}_3 \;\&\; \mathcal{N}_2\mathcal{N}_2 \mid c \\ \mathcal{N}_2 & \to & \mathcal{N}_1\mathcal{N}_1 \;\&\; \mathcal{N}_2\mathcal{N}_6 \mid cc \\ \mathcal{N}_3 & \to & \mathcal{N}_1\mathcal{N}_2 \;\&\; \mathcal{N}_6\mathcal{N}_6 \mid ccc \\ \mathcal{N}_6 & \to & \mathcal{N}_1\mathcal{N}_2 \;\&\; \mathcal{N}_3\mathcal{N}_3 \end{array}$$

To understand the rules above, associate every word c^n with the number n, and each $\mathcal{N}_i$ with the set $\{i \cdot 4^n \mid n \in \mathbb{N}\}$. Since $c^n c^k = c^{n+k}$, concatenation of words corresponds to the summation of the respective numbers. The expression $\mathcal{N}_1\mathcal{N}_3 \,\&\, \mathcal{N}_2\mathcal{N}_2$ encodes the equation $4^m + 3 \cdot 4^k = 2 \cdot 4^l + 2 \cdot 4^s$, which holds if and only if $m = k = l = s$, when both sides become equal to 4^{k+1}. This newly obtained number is assigned, by the first rule, to the set of $\mathcal{N}_1$. ⊣

Building on the idea behind Example 2.5, Jeż and Okhotin [2010] devised grammars encoding Turing machine computations, leading to:

Theorem 2.6 (Jeż and Okhotin [2010])**.** *Given a unary conjunctive grammar G, it is undecidable whether $L(G)$ is (i) empty, (ii) finite, or (iii) regular.*

2.3 Databases and query languages

2.3.1 Syntactic notions.

A *signature* $\Sigma = (\mathit{Rel}, \mathit{Const})$ is given by a set Rel of *relation symbols* R with associated arities $m \geqslant 0$, and a set Const, of *constants*, which we assume to be disjoint from $\mathbb{Z}$. The notation $R(\overline{o})$ always implies that the arity of R is $|\overline{o}|$. Syntactic objects (formal expressions of a certain kind or finite sets of them) are built from relation symbols, constants, other symbols (e.g., variables, connectives, temporal operators, etc.) and integers. For any such object $\mathfrak{O}$, its *size* $|\mathfrak{O}|$ is the number of symbols required to write it down, where every occurrence of a symbol counts as one, and integers are encoded in *unary* (see the related discussion in Section 1.5).

2.3.2 Databases

A *database* D over a signature Σ is a finite set of *facts* of the form $R(a_1, \ldots, a_m)$, with $a_1, \ldots, a_m \in \mathit{Const}$. A *temporal database* $\mathcal{D}$, instead, is a finite set of *timestamped facts* of the form $R(a_1, \ldots, a_m, \ell)$, with $\ell \in \mathbb{Z}$ and $R \in \mathit{Rel}$ of arity m. We write $\min \mathcal{D}$ and $\max \mathcal{D}$ for the minimal resp. maximal timestamps appearing in $\mathcal{D}$, and $\mathrm{tem}(\mathcal{D})$ for the integer segment $[\min \mathcal{D}, \max \mathcal{D}]$. The constants that appear in D or $\mathcal{D}$ are called *objects*, the set of all objects of a (temporal) database is denoted by $\mathrm{obj}(D)$ resp. $\mathrm{obj}(\mathcal{D})$. In general, the letters D and $\mathcal{D}$ always denote non-temporal resp. temporal databases. We omit the word "temporal" where it does not cause confusion.

Homomorphisms. A homomorphism from D as above to another database D' is a mapping $h\colon \mathit{Const} \to \mathit{Const}$ such that

$$R(a_1, \ldots, a_m) \in D \qquad \text{implies} \qquad R(h(a_1), \ldots, h(a_m)) \in D'$$

Such a mapping is a homomorphism between two temporal databases, $\mathcal{D}$ and $\mathcal{D}'$, if for any $\ell \in \mathbb{Z}$

$$R(a_1, \ldots, a_m, \ell) \in \mathcal{D} \qquad \text{implies} \qquad R(h(a_1), \ldots, h(a_m), \ell) \in \mathcal{D}'$$

Queries and complexity measures. *Database queries* of the form $Q(\overline{x}, \overline{t})$ are formulated using query languages that we define in the following sections. For each of these languages we define the relation $\mathcal{D} \models Q(\overline{a}, \overline{k})$, saying "$\mathcal{D}$ *satisfies* $Q(\overline{a}, \overline{k})$". Here $\overline{a} \in \mathrm{obj}(\mathcal{D})^{|\overline{x}|}$ and $\overline{k} \in \mathrm{tem}(\mathcal{D})^{|\overline{t}|}$. *Query answering* is the problem of deciding whether $\mathcal{D} \models Q(\overline{a}, \overline{k})$ holds. We distinguish two complexity measures for this problem: for *combined complexity*, the size of the input is $|Q| + |\mathcal{D}| + |\overline{a}| + |\overline{k}|$, and for *data complexity*, it is $|\mathcal{D}| + |\overline{a}| + |\overline{k}|$, while Q is fixed.

2.3.3 First-order query languages with numerical predicates

We define several languages based on first-order logic. The first of them is FO($<, \equiv$) [Immerman, 1999].

Syntax of FO($<, \equiv$). Let $\mathsf{X} = \{x, y, \ldots\}$ and $\mathsf{T} = \{t, t', \ldots\}$ be countably infinite lists of *object variables* and *time variables*, respectively. We use Greek letters ξ, η to refer to variables when we do not want to specify their sort. We write $\overline{\xi}$ for a tuple of variables of length $|\overline{\xi}|$. An *object term* o is either an object variable $x \in \mathsf{X}$ or a constant $a \in \mathit{Const}$, while a *time term* τ is either a time variable $t \in \mathsf{T}$ or an integer $k \in \mathbb{Z}$.

Formulas of $\mathrm{FO}(<, \equiv)$ are defined according to the following BNF:

$$\varphi ::= \top \mid R(\overline{o}, \tau) \mid \tau < \tau' \mid \tau \equiv_m \tau' \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi \mid \exists t.\varphi$$

where $R \in \mathit{Rel}$ is of arity $|\overline{o}|$ and $m > 0$. An expression $R(\overline{o}, \tau)$ is called an *atom*. It is a *ground atom* when it is a fact, i.e. when $\overline{o} \subseteq \mathit{Const}$ and $\tau \in \mathbb{Z}$. We use the standard notions of free and bound variables and the abbreviations:

$$\begin{aligned} \varphi \vee \psi &\equiv \neg(\neg\varphi \wedge \neg\psi) & \forall \xi.\varphi &\equiv \neg\exists\xi.\neg\varphi \\ \varphi \to \psi &\equiv \neg\varphi \vee \psi & \bot &\equiv \neg\top \end{aligned} \tag{2.6}$$

We write $\varphi(\overline{\xi})$ to highlight that $\overline{\xi}$ are exactly the free variables of φ. A *sentence* is a formula without free variables. We also write $\exists\overline{\xi}$ and $\forall\overline{\xi}$ as shorthands for $\exists\xi_1 \ldots \exists\xi_n$ resp. $\forall\xi_1 \ldots \forall\xi_n$.

Given formula $\varphi(\overline{x}, \overline{x}', \overline{t}, \overline{t}')$, with $|\overline{x}| = n$ and $|\overline{t}| = m$, and two tuples $\overline{o}$ and $\overline{\tau}$ of object and time terms of respective lengths, we write $\varphi(\overline{o}, \overline{x}', \overline{\tau}, \overline{t}')$ for a formula obtained from $\varphi(\overline{x}, \overline{x}', \overline{t}, \overline{t}')$ by substituting each free occurrence of x_i by o_i and each free occurrence of t_j by τ_j, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$.

Semantics of $\mathrm{FO}(<, \equiv)$. Fix a temporal database $\mathcal{D}$ and assume all object and time terms belong to $\mathsf{X} \cup \mathsf{T} \cup \mathrm{obj}(\mathcal{D}) \cup \mathrm{tem}(\mathcal{D})$. An *assignment* is a function $\mathfrak{a} \colon \mathsf{X} \cup \mathsf{T} \to \mathrm{obj}(\mathcal{D}) \cup \mathrm{tem}(\mathcal{D})$ that maps object variables to objects and time variables to timestamps. By a slight abuse of notation, we extend $\mathfrak{a}$ to all object and time terms, so that $\mathfrak{a}(a) = a$ and $\mathfrak{a}(k) = k$ for $a \in \mathrm{obj}(\mathcal{D})$ and $k \in \mathrm{tem}(\mathcal{D})$. For $\alpha \in \mathrm{obj}(\mathcal{D}) \cup \mathrm{tem}(\mathcal{D})$, we write $\mathfrak{a}[\alpha/\xi]$ to denote the assignment that maps ξ to α and coincides with $\mathfrak{a}$ otherwise. We define the relation $\models$ ("satisfy") as follows.

$$\mathcal{D}, \mathfrak{a} \models \top \text{ for any } \mathcal{D} \text{ and } \mathfrak{a} \tag{2.7}$$

$$\mathcal{D}, \mathfrak{a} \models R(o_1, \ldots, o_m, \tau) \iff R(\mathfrak{a}(o_1), \ldots, \mathfrak{a}(o_m), \mathfrak{a}(\tau)) \in \mathcal{D} \tag{2.8}$$

$$\mathcal{D}, \mathfrak{a} \models \tau < \tau' \iff \mathfrak{a}(\tau) < \mathfrak{a}(\tau') \tag{2.9}$$

$$\mathcal{D}, \mathfrak{a} \models \tau \equiv_m \tau' \iff \mathfrak{a}(\tau) \equiv \mathfrak{a}(\tau') \pmod{m} \tag{2.10}$$

$$\mathcal{D}, \mathfrak{a} \models \neg\varphi \iff \text{it is not the case that } \mathcal{D}, \mathfrak{a} \models \varphi \tag{2.11}$$

$$\mathcal{D}, \mathfrak{a} \models \varphi_1 \wedge \varphi_2 \iff \mathcal{D}, \mathfrak{a} \models \varphi_1 \text{ and } \mathcal{D}, \mathfrak{a} \models \varphi_2 \tag{2.12}$$

$$\mathcal{D}, \mathfrak{a} \models \exists x.\varphi \iff \mathcal{D}, \mathfrak{a}[a/x] \models \varphi \text{ for some } a \in \mathrm{obj}(\mathcal{D}) \tag{2.13}$$

$$\mathcal{D}, \mathfrak{a} \models \exists t.\varphi \iff \mathcal{D}, \mathfrak{a}[k/t] \models \varphi \text{ for some } k \in \mathrm{tem}(\mathcal{D}) \tag{2.14}$$

Note that $\mathcal{D}, \mathfrak{a} \models \varphi$ holds or fails simultaneously for all assignments when φ is a sentence. Accordingly, we write simply $\mathcal{D} \models \varphi$.

FO(<) is a fragment of FO($<, \equiv$) that disallows the modulo comparisons $\tau \equiv_m \tau'$. Note that all other order relations, i.e. $>, \leqslant, \geqslant$, as well as equality, increment and decrement functions $+1$ and -1 on time variables, and constants $\min \mathcal{D}$ and $\max \mathcal{D}$ are definable in FO($<$). We will need two more extensions of FO($<$) that in fact subsume FO($<, \equiv$).

FO(<, MOD) extends FO($<$) with quantifiers $\exists^N t$, for all $N > 1$, that are defined by setting $\mathcal{D}, \mathfrak{a} \models \exists^N t.\varphi(\bar{\xi}, t)$ whenever the cardinality of $\{k \in \mathrm{tem}(\mathcal{D}) \mid \mathcal{D}, \mathfrak{a} \models \varphi(\bar{\xi}, k)\}$ is divisible by N. Note that $t \equiv 0 \pmod{N}$ is definable as $\exists^N t'.(t' < t)$.

FO(RPR) extends FO($<$) with *relational primitive recursion* [Compton and Laflamme, 1990]. The latter is a limited form of recursion, the analogue of primitive recursion on numbers, that allows to define new predicates by unfolding them step by step along the linear order on timestamps. Our definition follows that in Artale et al. [2022]. We can construct formulas $\varphi(\eta, \xi_1, \ldots, \xi_n)$ such as

$$\left[\begin{array}{l} P_1(\xi_1, s) \equiv \theta_1(\xi_1, s, P_1(z_1, s-1), \ldots, P_n(z_n, s-1)) \\ \ldots \\ P_n(\xi_n, s) \equiv \theta_n(\xi_n, s, P_1(z_1, s-1), \ldots, P_n(z_n, s-1)) \end{array}\right] \psi(\eta, \bar{\xi})$$

where the part of φ within $[\ldots]$ defines recursively, via the FO(RPR) formulas θ_i, the interpretations of the relations P_i in the FO(RPR) formula ψ. The recursion starts at $t = 0$ assuming that all $P_i(\xi_i, -1)$

are false. Thus, the truth value of $P_i(\xi_i, 0)$ is computed by substituting $\bot$ for all $P_i(\xi_i, -1)$. We allow the relation symbols P_i to occur in only one recursive definition $[\dots]$, so it makes sense to write $\mathcal{D} \models P_i(\overline{\alpha}, k)$, for any tuple $\overline{\alpha} \subseteq \mathrm{obj}(\mathcal{D}) \cup \mathrm{tem}(\mathcal{D})$ and $k \in \mathrm{tem}(\mathcal{D})$, if the computed value is "true". Using thus defined truth-values, we compute inductively the truth-relation $\mathcal{D} \models \psi(\beta, \overline{\alpha})$, and so $\mathcal{D} \models \varphi(\beta, \overline{\alpha})$, as usual in first-order logic.

Note that FO(RPR) is a two-sorted logic, while the original definition by Compton and Laflamme [1990] features time variables *only*. To overcome this, Artale et al. [2022] employ the following trick: enumerate $\mathrm{obj}(\mathcal{D}) = \{a_1, \dots, a_n\}$ and associate each a_i with its number $i \in \mathrm{tem}(\mathcal{D})$. The latter set can be extended, if needed, by adding dummy facts $R(a_i, i)$, where R does not appear in $\mathcal{D}$. With this at hand, we regard the object variables in FO(RPR) as syntactic sugar over the original logic by Compton and Laflamme [1990].

A **first-order query** is any formula $\varphi(\overline{x}, \overline{t})$ as introduced above. Table 2.1 shows, in the first three columns, the data complexity of the answering problem for such queries, i.e. the problem of checking whether $\mathcal{D} \models \varphi(\overline{a}, \overline{k})$, for the four languages we defined. Recall that, for data complexity, φ is fixed and the input is $\mathcal{D}$, $\overline{a} \in \mathrm{obj}(\mathcal{D})^{|\overline{x}|}$ and $\overline{k} \in \mathrm{tem}(\mathcal{D})^{|\overline{t}|}$.

The expressiveness, i.e. the ability to define sets of the form $\{(\mathcal{D}, \overline{a}, \overline{k}) \mid \mathcal{D} \models \varphi(\overline{a}, \overline{k})\}$, strictly increases from FO(<) to FO(<, ≡), and then to FO(<, MOD), and is strongly believed to further increase with FO(RPR) [Immerman, 1999].

FO. The standard one-sorted first-order logic, FO, is the fragment of FO(<, ≡) without time variables. First-order formulas are naturally interpreted over non-temporal databases. A formula φ is *preserved* under homomorphisms if $D \models \varphi(\overline{a})$ implies $D' \models \varphi(h(\overline{a}))$ for any homomorphism h from D to D'. Every such φ is equivalent to a *positive existential formula* of the form $\exists \overline{y}.\psi(\overline{x}, \overline{y})$, where ψ is a

FO variants				*datalog*(S, <)	
<	<, ≡	<, MOD	RPR	linear	full
AC^0		ACC^0	NC^1-comp.	NL-comp.	P-comp.

Table 2.1: Data complexity for query languages.

quantifier free-formula built from atoms using $\wedge$ and $\vee$ only. Such formulas correspond to *unions of conjunctive queries*, the basic class of queries in database theory [Abiteboul et al., 1995]. This motivates the vision of the logics introduced above as basic query languages for temporal databases, although no result is known on the shape of formulas preserved under homomorphisms.

2.3.4 Datalog

Datalog is a recursive query language for relational databases [Ceri et al., 1989, 1990]. We consider a slight modification tailored to temporal databases, which we denote *datalog*(S, <), where S stands for the *successor relation* on $\mathbb{Z}$: $\mathrm{S}(k, m)$ is true whenever $m = k + 1$.

Syntax. A *rule* of *datalog*(S, <) has the form

$$C(\overline{x}, t_0) \leftarrow R_1(\overline{u}_1, t_1) \wedge \cdots \wedge R_s(\overline{u}_s, t_s) \wedge \psi(t_0, t_1, \ldots, t_s) \quad (2.15)$$

where $\overline{x} \subseteq \overline{u}_1 \cup \cdots \cup \overline{u}_s$, R_i and C are relation symbols and ψ is a conjunction of atoms of the forms $\mathrm{S}(t, t')$ and $t < t'$ for $t, t' \in \{t_0, t_1, \ldots, t_s\}$.

The part of the rule to the left of the arrow is called its *head* and the right-hand side its *body*. A *program*, Π, is a finite set of rules. The relation symbols that appear in rule heads of a program Π are called *intensional*, or IDBs, while the rest are *extensional*, or EDBs. The respective signatures are denoted $IDB(\Pi)$ and $EDB(\Pi)$. *Recursive*

rules are those that contain IDB atoms in their bodies, the rest are called *initialisation rules*. When a program Π is given, we assume that all database facts are given using EDB symbols only. A program is *linear* if every rule contains at most one intensional atom in the body.

Semantics. Note that every such rule ρ can be read right-to-left as an FO($<$) formula $\rho(\overline{\xi})$. We say that $\mathcal{D}$ *satisfies* a rule $\rho(\overline{\xi})$ of the form (2.15), if $\mathcal{D} \models \forall \overline{\xi} \,.\, \rho(\overline{\xi})$. Furthermore, we write $(\Pi, \mathcal{D}) \models R(\overline{a}, \ell)$ if $R(\overline{a}, \ell) \in \mathcal{D}'$, for every $\mathcal{D}'$ that satisfies every rule of Π and such that $\mathrm{obj}(\mathcal{D}) = \mathrm{obj}(\mathcal{D}')$, $\mathcal{D} \subseteq \mathcal{D}'$.

Queries. A *datalog*$(\mathrm{S}, <)$ *query* is a pair $Q = (\Pi, G)$, where $G \in \mathit{IDB}(\Pi)$ is called the *goal predicate*. It is called *linear* when so is Π. We write $\mathcal{D} \models Q(\overline{a}, \ell)$ whenever $(\Pi, \mathcal{D}) \models G(\overline{a}, \ell)$.

For a fixed query, checking whether $\mathcal{D} \models Q(\overline{a}, \ell)$ is P-complete and NL-complete for linear queries. To see this, recall that standard datalog is the version of *datalog*$(\mathrm{S}, <)$ without time variables and the respective conditions ψ. Every *datalog*$(\mathrm{S}, <)$ rule (2.15) can be regarded as a rule of datalog if one sees variables t_0 and t_i's as object variables and adds the relation symbols S and $<$ to the signature. Such a rule can be interpreted over a non-temporal database $D_{\mathcal{D}}$, with the domain $\mathrm{obj}(\mathcal{D}) \cup \mathrm{tem}(\mathcal{D})$, defined over a signature where arities of relation symbols are increased by one and containing the facts of $\mathcal{D}$ plus the facts $\mathrm{S}(\ell, \ell + 1)$ and $(\ell < \ell')$ for all appropriate $\ell, \ell + 1, \ell' \in \mathrm{tem}(\mathcal{D})$. It is not hard to see that $D_{\mathcal{D}}$ can be constructed from $\mathcal{D}$ by a constant-depth polynomial-size circuit. Therefore, answering *datalog*$(\mathrm{S}, <)$ queries essentially amounts to answering datalog queries, thus inheriting the complexity of the latter [Gottlob and Papadimitriou, 2003] and completing Table 2.1.

Note that the relation $<$ can be expressed using S and recursion in *datalog*$(\mathrm{S}, <)$ as follows:

$$(t < t') \leftarrow \mathrm{S}(t, t')$$
$$(t < t') \leftarrow \mathrm{S}(t, t'') \wedge (t'' < t')$$

However, making $<$ an IDB relation may break the linearity of certain linear *datalog*$(\mathrm{S}, <)$ programs. We also observe that *datalog*$(\mathrm{S}, <)$ queries are preserved under homomorphisms, just like standard datalog queries.

2.4 Ontology languages

The query languages of the previous section are meant to formulate queries for evaluation over temporal databases. In this section, we introduce logics for reasoning over structures with potentially infinite domains and timelines. These logics handle time via *temporal operators* rather than numerical predicates, allowing one to adapt the intuition and the methods developed for the propositional *linear temporal logic*, *LTL* [Prior, 1967, Pnueli, 1977, Manna and Pnueli, 1991]. We start by introducing first-order temporal logic, also known as *quantified temporal logic*, $\mathcal{QTL}$ [Gabbay et al., 1994, 2005].

Syntax of $\mathcal{QTL}$. We use again the set $\mathsf{X} = \{x, y, \dots\}$ of variables and the notion of an object term $o \in \mathsf{X} \cup \mathit{Const}$. A $\mathcal{QTL}$ formula is an expression φ defined by the following BNF:

$$\varphi \ ::= \ \top \ \mid \ R(\bar{o}) \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \exists x.\varphi \ \mid \ \mathcal{O}\varphi \tag{2.16}$$

where $\mathcal{O} \in \{\bigcirc, \bigcirc^-, \Diamond, \Diamond^-, \Box, \Box^-\}$. We write $\mathcal{O}^n$, $\mathcal{O} \in \{\bigcirc, \Diamond, \Box\}$, for a sequence of n symbols $\mathcal{O}$ if $n > 0$, of $|n|$ symbols $\mathcal{O}^- \in \{\bigcirc^-, \Diamond^-, \Box^-\}$ if $n < 0$, and an empty sequence if $n = 0$. As before, we use the abbreviations (2.6).

Semantics of $\mathcal{QTL}$ are defined for *temporal structures*. A structure $\mathcal{I} = (\Delta_\mathcal{I}, \cdot^\mathcal{I})$ over a signature Σ is defined by specifying a non-empty *domain* $\Delta_\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$ that associates each relation symbol $R \in \mathit{Rel}$ of arity m with a relation $R^\mathcal{I} \subseteq \Delta_\mathcal{I}^m$ and each constant $a \in \mathit{Const}$ with a domain element $a^\mathcal{I} \in \Delta_\mathcal{I}$. Note that a *propositional* symbol P (i.e. of arity 0) is interpreted as either

$\varnothing$ or $\{()\}$. We associate these with Boolean values $\bot$ (false) and $\top$ (true), respectively.

A *temporal structure* over Σ is a family of structures $\mathfrak{I} = (\mathcal{I}_\ell)_{\ell\in\mathbb{Z}}$ that share the same signature, domain $\Delta_\mathfrak{I}$, and the interpretation of constants. The indices ℓ are called *timestamps*. For clarity, we denote individual structures $\mathcal{I}_\ell$ by $\mathfrak{I}|_\ell$ and interpretation functions $\cdot^{\mathcal{I}_\ell}$ by $\cdot^{\mathfrak{I},\ell}$, and write $\mathfrak{I} = (\Delta_\mathfrak{I}, \cdot^\mathfrak{I})$. Since $a^{\mathfrak{I},\ell} = a^{\mathfrak{I},\ell'}$ for all $a \in \mathit{Const}$ and $\ell, \ell' \in \mathbb{Z}$, we write simply $a^\mathfrak{I}$. The domain elements that do not correspond to constants are called *anonymous*.

A homomorphism from $\mathcal{I}$ to another structure $\mathcal{I}'$ is a mapping $h\colon \Delta_\mathcal{I} \to \Delta_{\mathcal{I}'}$, such that $h(a^\mathcal{I}) = a^{\mathcal{I}'}$ and $(d_1, \dots, d_m) \in R^\mathcal{I}$ implies $(h(d_1), \dots h(d_m)) \in R^{\mathcal{I}'}$. For temporal structures, a homomorphism from $\mathfrak{I}$ to $\mathfrak{I}'$ is such an h that is a homomorphism from $\mathfrak{I}|_\ell$ to $\mathfrak{I}'|_\ell$ for every $\ell \in \mathbb{Z}$.

The formulas of $\mathcal{QTL}$ are interpreted over temporal structures *relative* to a timestamp. Given a temporal structure $\mathfrak{I}$, an *assignment* is a function $\mathfrak{a}\colon \mathsf{X} \to \Delta_\mathfrak{I}$. Additionally, we will assume that $\mathfrak{a}(a) = a^\mathfrak{I}$ for $a \in \mathit{Const}$. We define the satisfaction relation $\models$ as follows:

$$\mathfrak{I}, \ell, \mathfrak{a} \models \top \text{ for any } \mathfrak{I}, \ell \text{ and } \mathfrak{a} \tag{2.17}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models R(o_1, \dots, o_m) \iff (\mathfrak{a}(o_1), \dots, \mathfrak{a}(o_m)) \in R^{\mathfrak{I},\ell} \tag{2.18}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \neg\varphi \iff \text{it is not the case that } \mathfrak{I}, \ell, \mathfrak{a} \models \varphi \tag{2.19}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \varphi_1 \wedge \varphi_2 \iff \mathfrak{I}, \ell, \mathfrak{a} \models \varphi_1 \text{ and } \mathfrak{I}, \ell, \mathfrak{a} \models \varphi_2 \tag{2.20}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \exists x.\varphi \iff \mathfrak{I}, \ell, \mathfrak{a}[d/x] \models \varphi \text{ for some } d \in \Delta_\mathfrak{I} \tag{2.21}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \bigcirc\varphi \iff \mathfrak{I}, \ell + 1, \mathfrak{a} \models \varphi \tag{2.22}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \bigcirc^-\varphi \iff \mathfrak{I}, \ell - 1, \mathfrak{a} \models \varphi \tag{2.23}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \Diamond\varphi \iff \mathfrak{I}, \ell', \mathfrak{a} \models \varphi \text{ for some } \ell' > \ell \tag{2.24}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \Diamond^-\varphi \iff \mathfrak{I}, \ell', \mathfrak{a} \models \varphi \text{ for some } \ell' < \ell \tag{2.25}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \Box\varphi \iff \mathfrak{I}, \ell', \mathfrak{a} \models \varphi \text{ for all } \ell' > \ell \tag{2.26}$$

$$\mathfrak{I}, \ell, \mathfrak{a} \models \Box^-\varphi \iff \mathfrak{I}, \ell', \mathfrak{a} \models \varphi \text{ for all } \ell' < \ell \tag{2.27}$$

When φ is a sentence, we omit the redundant assignment symbol and write just $\mathfrak{I}, \ell \models \varphi$.

Models and entailments. We say that a temporal structure $\mathfrak{I}$ is a *model* of a $\mathcal{QTL}$ sentence φ, if $\mathfrak{I}, 0 \models \varphi$. Furthermore, $\mathfrak{I}$ is a model of a fact $R(\overline{a}, \ell)$, if $\mathfrak{I}, \ell \models R(\overline{a})$ *and* $a_i^{\mathfrak{I}} \neq a_j^{\mathfrak{I}}$ when $a_i \neq a_j$. The latter requirement is a standard convention in logic-based approaches to databases known as the *unique name assumption* [Brachman and Levesque, 2004].

A $\mathcal{QTL}$ *ontology* $\mathcal{O}$ is a finite set of $\mathcal{QTL}$ sentences called *axioms*. A (temporal) *knowledge base* (KB) is a pair $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ of an ontology and a temporal database. The reason why we distinguish between ontologies and databases is that we will actually be working with constant-free ontologies, so that databases will be our only source of ground atoms.

A temporal structure $\mathfrak{M}$ is a *model* of $\mathcal{O}$ if it is a model of every $\varphi \in \mathcal{O}$; it is a model of a KB $(\mathcal{O}, \mathcal{D})$ if it is additionally a model of every fact in $\mathcal{D}$ and $a^{\mathfrak{I}} \neq b^{\mathfrak{I}}$ for all $a, b \in \mathrm{obj}(\mathcal{D})$, $a \neq b$. A model $\mathfrak{M}$ of $(\mathcal{O}, \mathcal{D})$ is called *flat* when $\Delta_{\mathfrak{M}} = \{a^{\mathfrak{M}} \mid a \in \mathrm{obj}(\mathcal{D})\}$. Consistency checking is the problem of deciding whether a given KB has a model.

In general, the notation $\mathfrak{M} \models$ "X" means that $\mathfrak{M}$ is a model of "X". Then, we say that $\mathcal{O}$ *entails* a sentence φ, written $\mathcal{O} \models \varphi$, if $\mathfrak{M} \models \mathcal{O}$ implies $\mathfrak{M} \models \varphi$. A knowledge base $(\mathcal{O}, \mathcal{D})$ *entails* a fact $R(\overline{a}, \ell)$, written $(\mathcal{O}, \mathcal{D}) \models R(\overline{a}, \ell)$, if $\mathfrak{M} \models (\mathcal{O}, \mathcal{D})$ implies $\mathfrak{M} \models R(\overline{a}, \ell)$.

Linear temporal logic (*LTL*) corresponds to the propositional fragment of $\mathcal{QTL}$. That is, *LTL* formulas are those $\mathcal{QTL}$ formulas that are built purely from propositional symbols. Consistency checking for *LTL* knowledge bases is PSPACE-complete [Manna and Pnueli, 1991]. Note that a ground atom $R(\overline{a})$ can be seen as a proposition, and quantifier-free $\mathcal{QTL}$ formulas over ground atoms — as *LTL* formulas. Sometimes, it is also convenient to see constant-free $\mathcal{QTL}$ formulas built with a unique variable x and

interpreted over a database with a unique object as *LTL* formulas.

The universal fragment of $\mathcal{QTL}$ is that of formulas of the form $\forall\overline{\xi}\,.\,\varphi(\xi)$, where φ is quantifier-free. Suppose $\mathcal{O}$ given in the universal fragment and fix a finite set $C \subseteq \mathit{Const}$. Then, $\mathcal{O}$ can be grounded to an *LTL* ontology $\mathcal{O}'$ by substituting variables in its formulas in all possible ways with constants from C and removing the quantifiers. Thus, every KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ can be grounded to an *LTL* KB $\mathcal{K}' = (\mathcal{O}', \mathcal{D})$, where the ground atoms in $\mathcal{D}$ are seen as propositions. Then, $\mathcal{K}$ is consistent whenever $\mathcal{K}'$ is consistent [Ebbinghaus et al., 1994, Baader et al., 2010].

We note that already for the non-temporal fragment of $\mathcal{QTL}$, first-order logic, consistency checking is undecidable. We will use $\mathcal{QTL}$ as an umbrella language, relying on its semantics to define temporal extensions of two other relational formalisms: description logics [Baader et al., 2010] and datalog [Ceri et al., 1989, 1990]. We use the conventional syntax of these languages, although both can be seen as fragments of $\mathcal{QTL}$.

2.4.1 Temporal description logics

Temporal description logics (TDLs) are defined over a fixed signature $\Sigma = (\mathsf{N_C} \cup \mathsf{N_R}, \mathsf{N_I})$, where $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ are countably infinite sets of *concept names*, *role names*, and *individual names*. Concept names have arity 1, and written in capital case, while role names have arity 2, and written in lowercase. We will be mostly interested in two TDLs: temporal $\mathcal{EL}$ [Artale et al., 2007, Gutiérrez-Basulto et al., 2016a] and temporal *DL-Lite* [Artale et al., 2014]. The definition below follows that Artale et al. [2022] and provides a common syntactic ground for these two logics.

Syntax. A *role* r is either a role name p or its *inverse* p^-, with r^{--} identified with r. A *concept* C is either a concept name A or an expression of the form $\exists r.A$, called an *existential restriction*. A

temporalised concept (respectively, role) has the form $\mathcal{O}^*C$ (respectively, $\mathcal{O}^*r$), where $\mathcal{O}^*$ denotes an arbitrary sequence of temporal operators $\bigcirc, \bigcirc^-, \Diamond, \Diamond^-, \Box, \Box^-$.

A *concept (role) inclusion* takes the form:

$$\vartheta_1 \sqcap \ldots \sqcap \vartheta_k \sqsubseteq \vartheta_{k+1} \sqcup \ldots \sqcup \vartheta_{k+m} \tag{2.28}$$

where the ϑ_i are temporalised concepts (respectively, roles). By definition, an empty *body* (the left-hand side) amounts to $\top$ and an empty *head* (the right-hand side) to $\bot$. An inclusion is *Horn* when its head contains at most one disjunct and no operators $\Diamond$ or $\Diamond^-$, and *non-Horn*, otherwise. Horn formulas enjoy several well-known model-theoretic properties [Horn, 1951, Vardi, 1982]. In particular, every consistent Horn knowledge base $\mathcal{K}$ has a *canonical model* $\mathfrak{M}$, in the sense that for every model $\mathfrak{N}$ of $\mathcal{K}$ there exists a homomorphism from $\mathfrak{M}$ to $\mathfrak{N}$. We will formulate this property more precisely for particular sublanguages in Chapters 3 and 4.

A *TBox*, $\mathcal{T}$, is a finite set of concept inclusions, and an *RBox*, $\mathcal{R}$, is a finite set of role inclusions. A TDL ontology $\mathcal{O}$ is a union $\mathcal{T} \cup \mathcal{R}$.

We denote by $\mathsf{N_C}(\mathcal{O})$ and $\mathsf{N_R}(\mathcal{O})$, respectively, the sets of concept and role names appearing in $\mathcal{O}$. Additionally, we write $\mathrm{Rol}(\mathcal{O})$ for the set $\{p, p^- \mid p \in \mathsf{N_R}(\mathcal{O})\}$.

Semantics of concept and role inclusions is defined via a translation to $\mathcal{QTL}$. For every concept name A and role name p, let

$$A^\dagger = A(x) \qquad (\exists r.A)^\dagger = \exists y\,.\, r^\dagger(x,y) \wedge A(y) \tag{2.29}$$

$$p^\dagger = p(x,y) \qquad (p^-)^\dagger = p(y,x) \tag{2.30}$$

For a temporalised concept (role), we naturally assume:

$$(\mathcal{O}^*C)^\dagger = \mathcal{O}^*C^\dagger \qquad (\mathcal{O}^*r)^\dagger = \mathcal{O}^*r^\dagger \tag{2.31}$$

Then, a concept inclusion α of the form (2.28) is interpreted as a $\mathcal{QTL}$ formula $\alpha^\dagger$:

$$\Box\Box^- \left[\forall x\,.\, \vartheta_1^\dagger \wedge \ldots \wedge \vartheta_k^\dagger \rightarrow \vartheta_{k+1}^\dagger \vee \ldots \vee \vartheta_{k+m}^\dagger\right]$$

and a role inclusion β as a formula $\beta^\dagger$:

$$\Box\Box^- \, [\forall x \forall y \, . \, \vartheta_1^\dagger \, \wedge \, \ldots \, \wedge \, \vartheta_k^\dagger \, \rightarrow \, \vartheta_{k+1}^\dagger \, \vee \, \ldots \, \vee \, \vartheta_{k+m}^\dagger]$$

We write $\mathfrak{I}, \ell \models \alpha$ and $\mathfrak{I}, \ell \models \beta$ whenever $\mathfrak{I}, \ell \models \alpha^\dagger$ and $\mathfrak{I}, \ell \models \beta^\dagger$, respectively. The definitions of models and entailments are also imported from $\mathcal{QTL}$.

Note an important subtlety of these definitions. In $\mathcal{QTL}$, we write $\mathfrak{I} \models \varphi$ (i.e. $\mathfrak{I}$ is a model of φ) when $\mathfrak{I}, 0 \models \varphi$. For a concept (role) inclusion θ, however, we write $\mathfrak{I} \models \theta$ when $\mathfrak{I} \models \theta^\dagger$. Since $\theta^\dagger = \Box\Box^-\psi$, $\mathfrak{I} \models \theta$ amounts to $\mathfrak{I}, \ell \models \psi$ for *every* $\ell \in \mathbb{Z}$.

For the logic defined above, knowledge base consistency is still undecidable [Artale et al., 2014]. Even when the temporal operators are dropped, the same problem is NExpTime-hard [Lutz and Sattler, 2002]. To obtain computationally tractable fragments, the syntax is restricted in one or another way, giving rise to various (temporal) description logic*s*, in plural. For example, (temporal) $\mathcal{EL}$ is the logic of concept inclusions only, where the right-hand side is limited to exactly one disjunct and inverse roles are forbidden [Baader et al., 2005]. The (temporal) *DL-Lite* family [Artale et al., 2009, 2014] features various logics where the existential restrictions are limited to the form $\exists r$, with $(\exists r)^\dagger(x) = \exists y \, . \, r(x, y)$.

As we show in Chapter 4, satisfiability is decidable in ExpSpace for temporal *DL-Lite* knowledge bases with TBoxes and restricted RBoxes. Further restrictions on TBoxes lead to lower complexity, and sometimes permit a richer RBox. In particular, some logics make use of so called *rigid* role names, i.e. such p that $p \sqsubseteq \Box\Box^- p$. In Chapter 3, we consider the temporal version of $\mathcal{EL}$, $\mathcal{TEL}^\circ$, that unites lightweight TBoxes with $\bigcirc/\bigcirc^-$ operators and rigid role names, and prove decidability for some of its fragments.

2.4.2 Temporal datalog

Temporal versions of the recursive query language datalog, such as $\mathit{datalog}_{1\mathrm{S}}$ [Chomicki and Imieliński, 1988] and TEMPLOG [Abadi and Manna, 1989, Baudinet, 1989], were introduced in the late 1980s. Essentially, $\mathit{datalog}_{1\mathrm{S}}$ is $\mathit{datalog}(\mathrm{S})$ interpreted over the whole $\mathbb{Z}$ instead of $\mathrm{tem}(\mathcal{D})$. On the other hand, TEMPLOG is based on temporal operators. We limit our attention to operators $\bigcirc, \bigcirc^-, \Diamond, \Diamond^-$ to keep the queries being preserved by homomorphisms. Thus, we introduce $\mathit{datalog}^{\bigcirc\Diamond}$, that we study later in Chapter 5.

Syntax. A $\mathit{datalog}^{\bigcirc\Diamond}$ *rule* has the form:

$$P(\overline{x}) \leftarrow \mathcal{O}^* R_1(\overline{u}_1) \wedge \cdots \wedge \mathcal{O}^* R_s(\overline{u}_s) \tag{2.32}$$

where $\overline{x} \subseteq \overline{u}_1 \cup \cdots \cup \overline{u}_s$, R_i and P are relation symbols and $\mathcal{O}^*$ is an arbitrary sequence of temporal operators $\bigcirc, \bigcirc^-, \Diamond$ and $\Diamond^-$. The part of the rule to the left of the arrow is called its *head* and the right-hand side — its *body*.

The definitions given in Section 2.3.4 apply. Additionally, a program is *monadic*, if all its IDBs are of arity at most 1, and *binary* if they are of arity at most 2.

Semantics. A rule ρ of the form (2.32) is interpreted as a $\mathcal{QTL}$ formula $\rho^\dagger$:

$$\Box\Box^- \, [\forall \overline{u}_1 \ldots \forall \overline{u}_s \, . \, \mathcal{O}^* R_1(\overline{u}_1) \wedge \cdots \wedge \mathcal{O}^* R_s(\overline{u}_s) \to P(\overline{x})] \tag{2.33}$$

We conventionally refer to ontologies in $\mathit{datalog}^{\bigcirc\Diamond}$, i.e. finite sets of its rules, as *programs*.

Observe that for a $\mathit{datalog}^{\bigcirc\Diamond}$ program Π, any knowledge base $(\Pi, \mathcal{D})$ is always consistent. Moreover, any $\mathfrak{M} \models (\Pi, \mathcal{D})$ can be safely restricted to $\mathrm{obj}(\mathcal{D})$ making it a flat model. It follows that $(\Pi, \mathcal{D}) \models R(\overline{a}, \ell)$ iff $\mathfrak{M} \models R(\overline{a}, \ell)$ for every *flat* model of $(\Pi, \mathcal{D})$.

The relational fragment of *datalog*$^{\bigcirc\Diamond}$ is just datalog [Ceri et al., 1989, 1990]. We will also need *disjunctive datalog*, where disjunction (possibly empty) is allowed in the rule heads:

$$P_1(\overline{x}_1) \vee \cdots \vee P_k(\overline{x}_k) \leftarrow R_1(\overline{u}_1) \wedge \cdots \wedge R_s(\overline{u}_s) \tag{2.34}$$

and all the rest is defined analogously.

2.5 Temporal conjunctive queries

We distinguish a fragment of $\mathcal{QTL}$ that we will see as a query language. The formulas of this fragment generalise the shape of *datalog*$^{\bigcirc\Diamond}$ rule bodies, and will be employed for analysing *datalog*$^{\bigcirc\Diamond}$ queries.

Syntax. A *temporal conjunctive query* (temporal CQ) is a positive primitive $\mathcal{QTL}$ formula, i.e. a formula of the form $q(\overline{x}) = \exists \overline{u} \,.\, \varphi(\overline{x}, \overline{u})$, where φ, the *body* of the query, is defined by the following BNF:

$$\varphi ::= R(\overline{z}) \mid (\varphi \wedge \varphi) \mid \mathcal{O}\varphi \tag{2.35}$$

with $R \in \mathit{Rel}$ and $\overline{z} \subseteq \overline{x} \cup \overline{u}$ and $\mathcal{O}$ being any of $\bigcirc, \bigcirc^-, \Diamond$ and $\Diamond^-$. We call $\overline{x}$ the *answer variables* of $q(\overline{x})$.

Semantics. We define semantics of primitive positive formulas over databases (instead of structures) as follows:[1]

$$\mathcal{D}, \ell \models R(a_1, \ldots, a_m) \iff R(a_1, \ldots, a_m, \ell) \in \mathcal{D} \tag{2.36}$$

$$\mathcal{D}, \ell \models \varphi_1 \wedge \varphi_2 \iff \mathcal{D}, \ell \models \varphi_1 \text{ and } \mathcal{D}, \ell \models \varphi_2 \tag{2.37}$$

$$\mathcal{D}, \ell \models \bigcirc\varphi \iff \mathcal{D}, \ell + 1 \models \varphi \tag{2.38}$$

$$\mathcal{D}, \ell \models \Diamond\varphi \iff \mathcal{D}, \ell' \models \varphi \text{ for some } \ell' > \ell \tag{2.39}$$

and symmetrically for $\bigcirc^-$ and $\Diamond^-$. Given a temporal database $\mathcal{D}$, a timestamp $\ell \in \mathbb{Z}$, and a query $q(\overline{x}) = \exists \overline{u} \,.\, \varphi(\overline{x}, \overline{u})$, we say that

[1] Note that it is still defined relative to a timestamp, as usual for $\mathcal{QTL}$.

$\mathcal{D}, \ell \models q(\overline{a})$ if there exist $\overline{b} \in \mathrm{obj}(\mathcal{D})^{|\overline{u}|}$ such that $\mathcal{D}, \ell \models \varphi(\overline{a}, \overline{b})$.

Temporal CQ answering problem is that of checking whether $\mathcal{D}, \ell \models q(\overline{a})$. Answering temporal CQs is not harder than that for non-temporal conjunctive queries [Abiteboul et al., 1995].

Proposition 2.7. *For a temporal CQ $q(\overline{x})$, checking whether $\mathcal{D}, \ell \models q(\overline{a})$ is in AC^0 for data complexity.*

Proof (Sketch). We show that any such q is FO(<)-rewritable in the sense that there exists an FO(<) formula $\psi(\overline{x}, t)$ such that for all ℓ and $\overline{a}$ as above $\mathcal{D}, \ell \models q(\overline{a})$ iff $\mathcal{D} \models \psi(\overline{a}, \ell)$. The result then follows from the data complexity of evaluating FO(<) formulas (cf. Table 2.1).

We outline the construction of the rewriting. Fix any $q(\overline{x}) = \exists \overline{u} \, . \, \varphi(\overline{x}, \overline{u})$. The main issue with the construction is that temporal subformulas of φ, say $\Diamond \varkappa$, may be true for $\overline{a}$ at $\ell \in \mathrm{tem}(\mathcal{D})$, because $\mathcal{D}, \ell' \models \varkappa(\overline{a})$ for $\ell' \notin \mathrm{tem}(\mathcal{D})$. Had that not been the case, we could construct the rewriting for q straightforwardly by structural induction on φ. To overcome this, let N be the number of temporal operators in φ. We use a property that for all tuples $\overline{a}$ of objects from $\mathrm{obj}(\mathcal{D})$ and subformulas $\varkappa$ of φ, we have

$$\begin{array}{l} \mathcal{D}, r + N + 1 \models \varkappa(\overline{a}) \iff \mathcal{D}, \ell \models \varkappa(\overline{a}) \text{ for all } \ell > r + N \\ \mathcal{D}, l - N - 1 \models \varkappa(\overline{a}) \iff \mathcal{D}, \ell \models \varkappa(\overline{a}) \text{ for all } \ell < l - N \end{array} \tag{2.40}$$

Thus, for any subformula $\varkappa(\overline{z})$ of φ, we construct, by induction, the formulas $\psi_\varkappa(\overline{z}, t)$ and $\psi^i_\varkappa(\overline{z})$ for $i \in [-N-1, \dots, -1] \cup [1, \dots, N+1]$, so that for any $\mathcal{D}$, $\mathrm{tem}(\mathcal{D}) = [l, r]$, and any objects $\overline{a} \in \mathrm{obj}(\mathcal{D})^{|\overline{z}|}$,

$$\begin{array}{lcll} \mathcal{D}, \ell \models \varkappa(\overline{a}) & \iff & \mathcal{D} \models \psi_\varkappa(\overline{a}, \ell) & \text{for } \ell \in [l, r], \\ \mathcal{D}, (r + i) \models \varkappa(\overline{a}) & \iff & \mathcal{D} \models \psi^i_\varkappa(\overline{a}) & \text{for } 1 \leqslant i \leqslant N + 1, \\ \mathcal{D}, (l + i) \models \varkappa(\overline{a}) & \iff & \mathcal{D} \models \psi^i_\varkappa(\overline{a}) & \text{for } -N - 1 \leqslant i \leqslant -1. \end{array}$$

For the base case, we set $\psi_R(\overline{z}, t) = R(\overline{z}, t)$ and $\psi^i_R(\overline{z}) = \bot$. For an induction step, e.g., $\psi^{-1}_{\bigcirc\varkappa}(\overline{z}) = \psi_\varkappa(\overline{z}, \min \mathcal{D})$, $\psi^{N+1}_{\bigcirc\varkappa}(\overline{z}) = \psi^{N+1}_\varkappa(\overline{z})$,

$\psi^i_{\bigcirc\varkappa}(\overline{z}) = \psi^{i+1}_{\varkappa}(\overline{z})$ for all other i, and finally $\psi_{\bigcirc\varkappa}(\overline{z}, t) = \exists t'((t' = t+1) \wedge \psi_{\varkappa}(\overline{z}, t')) \vee ((t = \max \mathcal{D}) \wedge \psi^1_{\varkappa}(\overline{z}))$.[2] The required rewriting of q is then the formula $\exists \overline{u} \,.\, \psi_\varphi(\overline{x}, \overline{u}, t)$. □

2.6 Ontology-mediated queries

An atomic *ontology-mediated query* is a pair $Q = (\mathcal{O}, R)$ of an ontology $\mathcal{O}$ and a relation symbol R. The problem of *query answering relative to an ontology* is the problem of deciding whether $(\mathcal{O}, \mathcal{D}) \models R(\overline{a}, \ell)$. As before, we distinguish two complexity measures for this problem, *combined* and *data complexity*, which assume the size of the input to be $|\mathcal{O}| + |\mathcal{D}| + |\overline{a}| + |\ell|$ and $|\mathcal{D}| + |\overline{a}| + |\ell|$, respectively. Note that we measure the size of ℓ as $|\ell|$, assuming, as mentioned before, a unary encoding of numbers.

We can define general, non-atomic ontology-mediated queries, where R is replaced by an arbitrary $\mathcal{QTL}$ formula φ. In the setting of $\mathcal{QTL}$, this is not a true generalisation: we use a relation symbol R_φ that does not appear in $(\mathcal{O}, \varphi)$ and convert the latter to an equivalent atomic one $(\mathcal{O} \cup \{\forall \overline{x} \,.\, \varphi(\overline{x}) \to R_\varphi(\overline{x})\}, R_\varphi)$. In restricted fragments of $\mathcal{QTL}$ atomic queries may be less general: e.g. in temporal datalog the above translation works only for formulas whose structure is that of rule bodies. However, we stick to atomic queries, since we will have enough to worry about already in this case.

Relations to consistency checking. For ontology languages that admit the construct $\bot$, atomic query answering relative to an ontology and consistency checking are mutually reducible by constant-depth circuits. For the forward direction, take a relation symbol R' of the same arity as R and not appearing in $(\mathcal{O}, \mathcal{D})$. Then $(\mathcal{O}, \mathcal{D}) \models R(\overline{a}, \ell)$ if and only if the KB $(\mathcal{O} \cup \{\forall \overline{x} \,.\, R'(\overline{x}) \wedge R(\overline{x}) \to \bot\}, \mathcal{D} \cup \{R'(\overline{a}, \ell)\})$ is *not* consistent. Conversely, take a propositional symbol P not appearing in $(\mathcal{O}, \mathcal{D})$. Then $(\mathcal{O}, \mathcal{D})$ is *not* consistent

[2] Recall from Section 2.3.3 that $\min \mathcal{D}$ and $\max \mathcal{D}$ and the function $+1$ are definable in FO(<).

if and only if $(\mathcal{O}, \mathcal{D}) \models P(0)$; indeed, any model of $(\mathcal{O}, \mathcal{D})$ (if one exists) may interpret P as true or false at 0.

Without negation, however, the situation is different: for instance, every knowledge base in $\mathit{datalog}^{\bigcirc\Diamond}$ is trivially satisfiable, whereas query answering w.r.t. $\mathit{datalog}^{\bigcirc\Diamond}$ programs is nontrivial.

2.6.1 Rewritings and rewritability

We now link the "plain" queries of Section 2.3 and the "fancy" ontology-mediated queries. A *rewriting language* $\mathcal{L}$ is one of FO(<), FO(<, ≡), FO(<, MOD), FO(RPR) or (linear) $\mathit{datalog}(\mathrm{S}, <)$. We say that an ontology-mediated query $Q = (\mathcal{O}, R)$ is *$\mathcal{L}$-rewritable* if there exists a query ψ_Q of $\mathcal{L}$, such that for every temporal database $\mathcal{D}$, $(\mathcal{O}, \mathcal{D}) \models R(\overline{a}, \ell)$ if and only if $\mathcal{D} \models \psi_Q(\overline{a}, \ell)$. This ψ_Q is called an *$\mathcal{L}$-rewriting* of Q.

It follows that the data complexity of answering $\mathcal{L}$-rewritable ontology-mediated queries is at most that of query answering for $\mathcal{L}$. The *$\mathcal{L}$-rewritability* problem arises: given an ontology-mediated query $Q = (\mathcal{O}, R)$, decide whether Q is $\mathcal{L}$-rewritable.

Database schemas. In some settings, the assumption is made that some relation symbols are reserved for the use in ontologies and do not occur in databases. Given a signature $\Sigma = (\mathit{Rel}, \mathit{Const})$, a *database schema* is any subset $\mathit{Rel}' \subseteq \mathit{Rel}$. When $\mathit{Rel}' \neq \mathit{Rel}$, we write queries as triples $Q = (\mathcal{O}, \mathit{Rel}', R)$ to emphasise that rewritability is considered only with respect to databases defined over the signature $\Sigma = (\mathit{Rel}', \mathit{Const})$. Note that R and the relation symbols of $\mathcal{O}$ may belong to $\mathit{Rel} \setminus \mathit{Rel}'$.

All the definitions above are projected to non-temporal ontology languages and databases in the expected way. We note that the target rewriting languages in this case are FO and (linear) datalog.

2.6.2 Answering linear *datalog*$^{\bigcirc\diamond}$ queries

We prove a result about the linear fragment of *datalog*$^{\bigcirc\diamond}$ that will be useful in Chapters 3 and 5. A *datalog*$^{\bigcirc\diamond}$ *query* has the form $(\Pi, \mathit{EDB}(\Pi), G)$, where Π is a *datalog*$^{\bigcirc\diamond}$ program and G the goal predicate. We will write just (Π, G), as the restriction of the database schema to $\mathit{EDB}(\Pi)$ is a convention for datalog.

The following theorem[3] builds on results of a similar kind obtained for temporal description logics [Artale et al., 2013a, 2014].

Theorem 2.8. *Answering linear datalog*$^{\bigcirc\diamond}$ *queries is* PSPACE*-complete and* NL*-complete, respectively, for combined and data complexity.*

Proof. (*Sketch*) The lower bounds come from the case of the standard, non-temporal linear datalog [Gottlob and Papadimitriou, 2003]. For the upper bound, fix a linear query (Π, G). Without loss of generality, we assume that temporalised IDB atoms in rule bodies of Π have the form $\bigcirc^k C(\overline{y})$, where $|k| \leqslant 1$ (the cases of $\diamond/\diamond^-$ and consecutive $\bigcirc/\bigcirc^-$ can be expressed via recursion). Given a temporal database $\mathcal{D}$, tuples of objects $\overline{c}$ and $\overline{a}$ from $\mathrm{obj}(\mathcal{D})$, and $\ell \in \mathbb{Z}$, we write $C(\overline{c}) \leftarrow_{\ell,k} D(\overline{a})$ if Π has a rule $C(\overline{x}) \leftarrow \varphi(\overline{x}, \overline{y}, \overline{u}) \wedge \bigcirc^k D(\overline{y})$ such that $\mathcal{D}, \ell \models \exists \overline{u} \,.\, \varphi(\overline{c}, \overline{a}, \overline{u})$. Analogously, we write $C(\overline{c}) \leftarrow_\ell$ if there is an initialisation rule $C(\overline{x}) \leftarrow \varphi(\overline{x}, \overline{u})$ and $\mathcal{D}, \ell \models \exists \overline{u} \,.\, \varphi(\overline{c}, \overline{u})$. Then, given a linear *datalog*$^{\bigcirc\diamond}$ query (Π, G), we have $\mathcal{D}, \Pi, \ell \models G(\overline{a})$ if and only if $G = C_0$, $\overline{a} = \overline{c}^0$, and there exists a sequence

$$C_0(\overline{c}^0) \leftarrow_{k_0} C_1(\overline{c}^1) \leftarrow_{k_1} \cdots \leftarrow_{k_{n-1}} C_n(\overline{c}^n) \tag{2.41}$$

such that $C_i \in \mathit{IDB}(\Pi)$, tuples $\overline{c}^i$ are from $\Delta_{\mathcal{D}}$, $\ell_0 = \ell$, $\ell_{i+1} = \ell_i + k_i$ for $0 \leqslant i < n$, $C_i(\overline{c}^i) \leftarrow_{\ell_i, k_i} C_{i+1}(\overline{c}^{i+1})$, and $C_n(\overline{c}^n) \leftarrow_{\ell_n}$. Let $\mathrm{tem}(\mathcal{D}) = [l, r]$. Note that a rule $C(\overline{c}) \leftarrow_{\ell,k} D(\overline{a})$ either holds or does not hold simultaneously for all $\ell > r + N$, where N is the

[3] Many thanks to Vladislav Ryzhikov for promptly proving this theorem at a critical moment.

number of temporal operators in Π, and, similarly, for all $\ell < l - N$. Now, consider a sequence (2.41) and ℓ, where $\ell > r + N$ (the case for $\ell < l - N$ is analogous). Any loop of the form $C_i(\overline{c}^i) \leftarrow_{k_i} \cdots \leftarrow_{k_{j-1}} C_j(\overline{c}^j)$ in it with $C_i(\overline{c}^i) = C_j(\overline{c}^j)$ can be removed as long as in the resulting sequence

$$C_0(\overline{c}^0) \leftarrow_{k_0} C_1(\overline{c}^1) \leftarrow_{k_1} \cdots \leftarrow_{k_{i-1}} C_i(\overline{c}^i) \leftarrow_{k_j} C_{j+1}(\overline{c}^{j+1}) \cdots \leftarrow_{k_{n-1}} C_n(\overline{c}^n), \quad (2.42)$$

the sum of all k_t remains $\geqslant 0$. This allows us to convert any sequence (2.41) to a sequence with the same C_0 in the beginning, the same C_n in the end, and where all ℓ_i do not exceed $\ell + O(|IDB(\Pi)| \cdot |\mathrm{obj}(\mathcal{D})|^m)$, for m equal to the maximal arity of a relation in $IDB(\Pi)$. This means that, for $\ell \in \mathrm{tem}(\mathcal{D})$, we can check $\mathcal{D}, \Pi, \ell \models G(\overline{a})$ using timestamps in the range $[l - O(|\Pi| \cdot |\mathrm{obj}(\mathcal{D})|^m), r + O(|\Pi| \cdot |\mathrm{obj}(\mathcal{D})|^m)]$. The existence of such a derivation can be checked nondeterministically in space polynomial in $|\Pi|$, but just logarithmic in $|\mathcal{D}|$, so in NL, for data complexity, and in PSPACE, for combined. □

Chapter 3

Reasoning in temporal description logic $\mathcal{TEL}^{\bigcirc}$

3.1 Overview

In this chapter, we study the temporal description logic $\mathcal{TEL}$, introduced by Artale et al. [2007] and Gutiérrez-Basulto et al. [2016a] as a fusion of the description logic $\mathcal{EL}$ and the temporal logic *LTL*. Concept inclusions of $\mathcal{TEL}$ are of the positive Horn fragment, i.e. their right-hand sides contain exactly one disjunct, and feature temporal operators $\bigcirc/\bigcirc^{-}$ and $\Diamond/\Diamond^{-}$ applied to concepts. Role inclusions are not allowed, except that one may declare some roles as *rigid*, i.e. that the relations they model do not change over time.

Example 3.1. Imagine that Alice is a professor in 2025, denoted $\mathsf{Prof}(\text{Alice}, 2025)$. Professorship is permanent and requires advising students, who in three years become doctors. Being an advisor of a doctor makes one proud, and proud professors are happy. This is

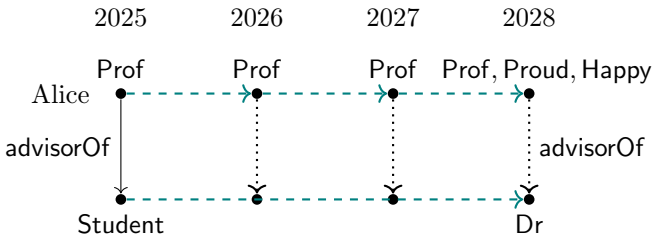


Figure 3.1: Representation of some inferences for Example 3.1. Dashed lines represent the temporal evolution of a given element, while dotted lines represent a relation whose existence is known due to role rigidity.

formalised in $\mathcal{TEL}$ as follows (using a rigid role $\mathsf{advisorOf}$):

$$\mathsf{Prof} \sqsubseteq \bigcirc\mathsf{Prof} \quad \mathsf{Prof} \sqcap \mathsf{Proud} \sqsubseteq \mathsf{Happy} \quad \mathsf{Student} \sqsubseteq \bigcirc^3\mathsf{Dr}$$
$$\mathsf{Prof} \sqsubseteq \exists \mathsf{advisorOf}.\mathsf{Student} \quad \exists \mathsf{advisorOf}.\mathsf{Dr} \sqsubseteq \mathsf{Proud}$$

Figure 3.1 provides a graphical representation of some information about Alice that can be inferred from $\mathsf{Prof}(\mathrm{Alice}, 2025)$ and the above $\mathcal{TEL}$ TBox. In particular, Alice is happy in 2028. ⊣

Despite the decidability of basic reasoning problems for both *LTL* and $\mathcal{EL}$, their combination in $\mathcal{TEL}$ quickly leads to undecidability of entailment checking [Artale et al., 2007] and query answering [Gutiérrez-Basulto et al., 2016a]. The authors of the latter proposed to limit the temporal operators to only $\bigcirc$ and $\bigcirc^-$, giving rise to the language $\mathcal{TEL}^{\bigcirc}$, but left open the question of whether that restriction by itself is enough to regain decidability of query answering and introduced additional syntactic constraints, based on some form of acyclicity (either on the description logics side, or on the temporal side). All these constraints enforce a crucial property: the existence of models where the evolution over time of any given object is, after some initial segment, periodic. This property was called *ultimate periodicity* by Gutiérrez-Basulto et al. [2016a].

We link temporal reasoning with $\mathcal{TEL}^{\circ}$-ontologies to the study of associated formal languages, which allows us to close the open question of Gutiérrez-Basulto et al. [2016a] and obtain additional results. After formally introducing $\mathcal{TEL}^{\circ}$ in Section 3.2, we discuss known decidability results for its fragments in Section 3.3. The sections that follow present our contributions.

We first consider the case of $\mathcal{TEL}^{\circ}_{future}$, where concept inclusions only allow to derive novel information about the future (or present) and not the past. For this fragment, we show in Section 3.4 that the task of deciding whether a concept inclusion of the form $A \sqsubseteq \bigcirc^n B$ is entailed by an ontology can be reduced to deciding whether n belongs to the Parikh image [Parikh, 1966] of a unary conjunctive language, and vice-versa. Such languages are defined by *conjunctive grammars*, introduced by Okhotin [2001] as a generalisation of context-free grammars which allows for a conjunction operation in rules.

We then turn our attention to $\mathcal{TEL}^{\circ}_{lin}$, a temporal extension of linear $\mathcal{EL}$ [Dimartino et al., 2016], and obtain in Section 3.5 a similar reduction from reasoning in $\mathcal{TEL}^{\circ}_{lin}$ to deciding whether the Parikh image of a context-free grammar over a binary alphabet fulfils some property.

In Section 3.6, we exploit the previous correspondences to obtain results for $\mathcal{TEL}^{\circ}$ and fragments thereof. First, we close negatively the question of whether all $\mathcal{TEL}^{\circ}$ ontologies enjoy ultimate periodicity, and complete the complexity picture of $\mathcal{TEL}$ given by Gutiérrez-Basulto et al. [2016a]: answering atomic queries mediated by $\mathcal{TEL}^{\circ}$ ontologies is undecidable. Second, we provide results for the new fragments $\mathcal{TEL}^{\circ}_{future}$ and $\mathcal{TEL}^{\circ}_{lin}$ we introduce. As for $\mathcal{TEL}^{\circ}_{future}$, we prove that answering atomic ontology-mediated queries is solvable in polynomial time (in combined and data complexity), but becomes undecidable if $\mathcal{TEL}^{\circ}_{future}$ is extended with rigid concept names (that do not change their interpretation in time) or the universal concept $\top$. Regarding $\mathcal{TEL}^{\circ}_{lin}$, we prove that $\mathcal{TEL}^{\circ}_{lin}$ ontologies enjoy ultimate periodicity, and derive complexity results for ontology-mediated queries.

3.2 The $\mathcal{TEL}^{\bigcirc}$ language

We recall the syntax and semantics of $\mathcal{TEL}^{\bigcirc}$ from Gutiérrez-Basulto et al. [2016a]. As in Section 2.4.1, let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be the sets of concept names, role names, and individual names, respectively. We additionally partition $\mathsf{N_R}$ into *rigid role names* $\mathsf{N_R^{rig}}$ and *local role names* $\mathsf{N_R^{loc}}$.

Syntax. A $\mathcal{TEL}^{\bigcirc}$ *TBox* $\mathcal{T}$ is a finite set of *concept inclusions* of the form

$$A \sqsubseteq \bigcirc^n B \qquad A \sqcap A' \sqsubseteq B \qquad \exists r.A \sqsubseteq B \qquad A \sqsubseteq \exists r.B \tag{3.1}$$

where $A, A', B \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $n \in \mathbb{Z}$.

Here, $\bigcirc^n$ is just an abbreviation of a sequence of n symbols $\bigcirc$, when $n > 0$, of $|n|$ symbols $\bigcirc^-$, when $n < 0$, and of an empty sequence, when $n = 0$. Equivalently, we can say that n is given in unary: the crucial property is that each appearance of n contributes to the size of the TBox as $|n|$ (see Remark 3.25). Therefore, when $n = 0$, $n = 1$, or $n = -1$, we simply write $A \sqsubseteq B$, $A \sqsubseteq \bigcirc B$, and $A \sqsubseteq \bigcirc^- B$, respectively.

We will further consider two fragments of $\mathcal{TEL}^{\bigcirc}$. The *future* fragment, $\mathcal{TEL}^{\bigcirc}_{future}$, is obtained by setting $n \geqslant 0$. The *linear* fragment, $\mathcal{TEL}^{\bigcirc}_{lin}$, disallows concept inclusions of the form $A \sqcap A' \sqsubseteq B$.

We denote by $\mathsf{N_C}(\mathcal{T})$, $\mathsf{N_R}(\mathcal{T})$, $\mathsf{N_R^{rig}}(\mathcal{T})$, and $\mathsf{N_R^{loc}}(\mathcal{T})$, respectively, the sets of concept names, role names, and rigid and local role names appearing in $\mathcal{T}$.

Semantics. Concept inclusions are interpreted over temporal structures as given in Section 2.4. We remind the reader of the definition of their meaning via the translation $\cdot^\dagger$ to $\mathcal{QTL}$ formulas:

$$(A \sqsubseteq \bigcirc^n B)^\dagger = \Box\Box^- [\forall x \,.\, A(x) \to \bigcirc^n B(x)] \tag{3.2}$$

$$(A \sqcap A' \sqsubseteq B)^\dagger = \Box\Box^- [\forall x \,.\, A(x) \wedge A'(x) \to B(x)] \tag{3.3}$$

$$(\exists r.A \sqsubseteq B)^\dagger = \Box\Box^- [\forall x \,.\, (\exists y \,.\, r(x,y) \wedge A(y)) \to B(x)] \tag{3.4}$$

$$(A \sqsubseteq \exists r.B)^{\dagger} = \Box\Box^{-}\,[\forall x\,.\,A(x) \rightarrow \exists y\,.\,r(x,y) \wedge B(y)] \tag{3.5}$$

Additionally, for every $r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}}$ we define a $\mathcal{QTL}$ formula

$$\theta_r = \Box\Box^{-}\,[\forall x \forall y\,.\,r(x,y) \rightarrow \Box\Box^{-} r(x,y)]$$

Then, a TBox $\mathcal{T}$ is translated to a $\mathcal{QTL}$-ontology $\mathcal{T}^{\dagger}$ as follows:

$$\mathcal{T}^{\dagger} = \{\alpha^{\dagger} \mid \alpha \in \mathcal{T}\} \cup \{\theta_r \mid r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}}(\mathcal{T})\} \tag{3.6}$$

Recall further that a temporal structure $\mathfrak{I}$ is a *model* of $\mathcal{T}$ if it is a model of $\mathcal{T}^{\dagger}$, and $\mathcal{T}$ *entails* α, written $\mathcal{T} \models \alpha$, whenever $\mathcal{T}^{\dagger}$ entails $\alpha^{\dagger}$.

Note that *rigid concept names* can be defined in a similar fashion as rigid role names: $\Box\Box^{-}\,[\forall x\,.\,A(x) \rightarrow \Box\Box^{-} A(x)]$. This can be simulated with $A \sqsubseteq \bigcirc A$ and $A \sqsubseteq \bigcirc^{-1} A$ in $\mathcal{TEL}^{\circ}$ and $\mathcal{TEL}^{\circ}_{lin}$, but not in $\mathcal{TEL}^{\circ}_{future}$.

Canonical models. For every $\mathcal{TEL}^{\circ}$ TBox $\mathcal{T}$ and database $\mathcal{D}$, there exists a *canonical model* $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ of $(\mathcal{T},\mathcal{D})$, i.e. such that for any model $\mathfrak{M}$ of $(\mathcal{T},\mathcal{D})$, there is a homomorphism from $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ to $\mathfrak{M}$. To see this, consider the following construction. Let N_{N} be an infinite countable set of *named nulls* disjoint from N_{I}, and set $\Delta_{\mathfrak{M}_{(\mathcal{T},\mathcal{D})}} = \mathsf{N}_{\mathsf{I}} \cup \mathsf{N}_{\mathsf{N}}$. Following Artale et al. [2013b], we represent $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ as its *atomic diagram* [Hodges, 1993], i.e. a potentially infinite set $\mathcal{M}$ of timestamped facts built from N_{R}, N_{C} and $\mathsf{N}_{\mathsf{I}} \cup \mathsf{N}_{\mathsf{N}}$ such that for every $n \in \mathbb{Z}$, $d \in A^{\mathfrak{M}_{(\mathcal{T},\mathcal{D}),n}}$ if and only if $A(d,n) \in \mathcal{M}$ and $(d,e) \in r^{\mathfrak{M}_{(\mathcal{T},\mathcal{D}),n}}$ if and only if $r(d,e,n) \in \mathcal{M}$. Formally, $\mathcal{M}$ is defined over an extended signature $(\mathsf{N}_{\mathsf{C}} \cup \mathsf{N}_{\mathsf{R}}, \mathsf{N}_{\mathsf{I}} \cup \mathsf{N}_{\mathsf{N}})$; when $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ is considered as a structure over the original signature, the nulls become anonymous.

We define $\mathcal{M} = \bigcup_{i \geqslant 0} \mathcal{M}_i$ where $\mathcal{M}_0 = \mathcal{D}$ and $\mathcal{M}_{i+1}$ is built from $\mathcal{M}_i$ by applying a rule of one of the following forms, assuming that the rule application is fair (i.e., if a rule can be applied, it is eventually applied):

(i) if $r(a,b,n) \in \mathcal{M}_i$, $r \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}}$ and there is $k \in \mathbb{Z}$ such that $r(a,b,k) \notin \mathcal{M}_i$, then $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{r(a,b,k) \mid k \in \mathbb{Z}\}$;

(ii) if $A(a,n) \in \mathcal{M}_i$, $A \sqsubseteq \bigcirc^k B \in \mathcal{T}$ and $B(a,n+k) \notin \mathcal{M}_i$, then $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{B(a,n+k)\}$;

(iii) if $A(a,n), A'(a,n) \in \mathcal{M}_i$, $A \sqcap A' \sqsubseteq B \in \mathcal{T}$ and $B(a,n) \notin \mathcal{M}_i$, then $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{B(a,n)\}$;

(iv) if $r(a,b,n), A(b,n) \in \mathcal{M}_i$, $\exists r.A \sqsubseteq B \in \mathcal{T}$ and $B(a,n) \notin \mathcal{M}_i$, then $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{B(a,n)\}$;

(v) if $A(a,n) \in \mathcal{M}_i$, $A \sqsubseteq \exists r.B \in \mathcal{T}$ and there is no $b \in \mathsf{N}_\mathsf{N}$ such that $r(a,b,n), B(b,n) \in \mathcal{M}_i$, then $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{r(a,b,n), B(b,n)\}$ for some $b \in \mathsf{N}_\mathsf{N}$ which does not occur in $\mathcal{M}_i$.

It is not hard to see that $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ is indeed a canonical model.

Lemma 3.2. *$\mathfrak{M}_{(\mathcal{T},\mathcal{D})} \models (\mathcal{T},\mathcal{D})$ and for every $\mathfrak{M} \models (\mathcal{T},\mathcal{D})$, there is a homomorphism $h : \mathsf{N}_\mathsf{I} \cup \mathsf{N}_\mathsf{N} \mapsto \Delta_\mathfrak{M}$ from $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ to $\mathfrak{M}$.*

Derivations. We will use the following derivation system for $\mathcal{TEL}^{\circ}$. Given a $\mathcal{TEL}^{\circ}$ TBox $\mathcal{T}$ and a database $\mathcal{D}$, we write $(\mathcal{T},\mathcal{D}) \vdash A(a,n)$ if there exists a derivation of $A(a,n)$ from $\mathcal{D} \cup \mathcal{T}$ using rules of the form (3.7)–(3.11). Formally, (3.7)–(3.11) are rule schemata, where the symbols a, b range over $\mathsf{N}_\mathsf{I} \cup \mathsf{N}_\mathsf{N}$, the symbols n, k range over $\mathbb{Z}$, the symbols A, A', B range over N_C, while r ranges over N_R and ρ over $\mathsf{N}_\mathsf{R}^{\mathsf{rig}}$. An instance of a rule schema is obtained by replacing these symbols with concrete individual names, integers, concept names, and role names of the appropriate kind.

A derivation of $A(a,n)$ from $(\mathcal{T},\mathcal{D})$ is then a sequence $(\mathcal{F}_0,\dots,\mathcal{F}_m)$ such that $\mathcal{F}_0 = \mathcal{D} \cup \mathcal{T}$, $A(a,n) \in \mathcal{F}_m$, and for every $1 \leqslant i \leqslant m$, the set $\mathcal{F}_i$ is obtained from $\mathcal{F}_{i-1}$ by choosing an instance of one of the rule schemata (3.7)–(3.11) whose left-hand side is contained in $\mathcal{F}_{i-1}$, and then adding to $\mathcal{F}_{i-1}$ the facts on its right-hand side.

$$\rho(a,b,n) \qquad \vdash \qquad \rho(a,b,k) \qquad (3.7)$$

$$A(a,n),\ A \sqsubseteq \bigcirc^k B \quad \vdash \quad B(a,n+k) \tag{3.8}$$

$$A(a,n),\ A'(a,n),\ A \sqcap A' \sqsubseteq B \quad \vdash \quad B(a,n) \tag{3.9}$$

$$r(a,b,n),\ A(b,n),\ \exists r.A \sqsubseteq B \quad \vdash \quad B(a,n) \tag{3.10}$$

$$A(a,n),\ A \sqsubseteq \exists r.B \quad \vdash \quad r(a,b,n),\ B(b,n) \tag{3.11}$$

In (3.11), the individual name $b \in \mathsf{N_N}$ is required to be new, that is, b must not occur in $\mathcal{F}_{i-1}$. Note further that an instance of (3.7) may use an arbitrary $k \in \mathbb{Z}$.

The next proposition is easily shown using the notion of canonical models.

Proposition 3.3. *For every $\mathcal{TEL}^\circ$ TBox $\mathcal{T}$, database $\mathcal{D}$, $A, B \in \mathsf{N_C}$, $a \in \mathsf{N_I}$, and $n, k \in \mathbb{Z}$:*

- $(\mathcal{T}, \mathcal{D}) \models A(a,n)$ *iff* $(\mathcal{T}, \mathcal{D}) \vdash A(a,n)$, *and*
- $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ *iff* $(\mathcal{T}, \{A(a,k)\}) \vdash B(a,k+n)$.

Example 3.4 (Example 3.1 continued)**.** *Below is a derivation of* $\mathsf{Happy}(\mathrm{Alice}, 2028)$*, from the TBox $\mathcal{T}$ that contains the concept inclusions from Example 3.1 and $\mathcal{D} = \{\mathsf{Prof}(\mathrm{Alice}, 2025)\}$, with* $\mathsf{advisorOf} \in \mathsf{N_R^{rig}}$*. The form of the rule applied is given as a subscript to $\vdash$ and the left-hand side is left implicit.*

$$\vdash_{(3.8)} \mathsf{Prof}(\mathrm{Alice}, 2026)$$
$$\vdash_{(3.8)} \mathsf{Prof}(\mathrm{Alice}, 2027)$$
$$\vdash_{(3.8)} \mathsf{Prof}(\mathrm{Alice}, 2028)$$
$$\vdash_{(3.11)} \mathsf{advisorOf}(\mathrm{Alice}, b, 2025), \mathsf{Student}(b, 2025)$$
$$\vdash_{(3.8)} \mathsf{Dr}(b, 2028)$$
$$\vdash_{(3.7)} \mathsf{advisorOf}(\mathrm{Alice}, b, 2028)$$
$$\vdash_{(3.10)} \mathsf{Proud}(\mathrm{Alice}, 2028)$$
$$\vdash_{(3.9)} \mathsf{Happy}(\mathrm{Alice}, 2028)$$

By Proposition 3.3, it holds that $(\mathcal{T}, \mathcal{D}) \models \mathsf{Happy}(\mathrm{Alice}, 2028)$ *and* $\mathcal{T} \models \mathsf{Prof} \sqsubseteq \bigcirc^3 \mathsf{Happy}$.

Query answering. We are interested in answering atomic ontology-mediated queries of the form $(\mathcal{T}, A)$, where $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}$ TBox, and A a concept name. Thus, *query answering relative to $\mathcal{TEL}^{\circ}$ TBoxes* is the problem of deciding whether $(\mathcal{T}, \mathcal{D}) \models A(a, n)$. Recall that we distinguish between *combined complexity*, where the size of the input is $|\mathcal{T}| + |\mathcal{D}| + |A(a, n)|$, and *data complexity*, where $\mathcal{T}$ is fixed.

3.3 Known results on $\mathcal{TEL}^{\circ}$

We survey the existing results for $\mathcal{TEL}^{\circ}$. Although Gutiérrez-Basulto et al. [2016a] left open the question of whether query answering relative to $\mathcal{TEL}^{\circ}$ TBoxes is decidable, they formulated an important sufficient condition for decidability: namely, the *ultimate periodicity* of a TBox, defined using the notion of *canonical quasimodels*.

Since we do not use quasimodels in this work, we rephrase this property using the sets of numbers $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$. A TBox $\mathcal{T}$ is *ultimately periodic* if for every $A, B \in \mathsf{N_C}(\mathcal{T})$, the set $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$ is *semilinear*.

Theorem 3.5 (Gutiérrez-Basulto et al. [2016a])**.** *Query answering relative to ultimately periodic $\mathcal{TEL}^{\circ}$ TBoxes is in* PSPACE*, for data complexity.*

The property of ultimate periodicity is a semantic one, it concerns the shape of canonical models and it is not immediately clear whether such a property holds for a given TBox. Gutiérrez-Basulto et al. [2016a] proposed three syntactic conditions on concept inclusions that guaranteed ultimate periodicity of TBoxes.

Limited use of rigid role names. The first condition allows using rigid role names only in concept inclusions of the form $\exists r.B \sqsubseteq A$. Thus, no rigid "connections" to nulls are created in derivations, and TBoxes become "rewritable" to propositional *LTL* ontologies. Query

answering in this case is PSPACE-complete for combined complexity, and P-complete for data.

Temporal acyclicity. The second condition disallows derivations to loop through the same concepts along the temporal dimension. More formally, let $\lambda\colon \mathsf{N_C} \to \mathbb{N}$ be a ranking of concept names. A TBox $\mathcal{T}$ is *temporally acyclic* if the following conditions hold:

- for $A \sqsubseteq \bigcirc^n B \in \mathcal{T}$, we have $\lambda(A) < \lambda(B)$;
- for $A \sqcap A' \sqsubseteq B \in \mathcal{T}$, we have $\lambda(A) = \lambda(A') = \lambda(B)$;
- for $\exists r.A \sqsubseteq B \in \mathcal{T}$ or $A \sqsubseteq \exists r.B \in \mathcal{T}$, we have $\lambda(A) = \lambda(B)$.

As a result, no new fact can be derived further than $\mathrm{poly}(|\mathcal{T}|)$ timestamps away from $\mathrm{tem}(\mathcal{D})$, and query answering relative to such TBoxes becomes P-complete for data and combined complexity.

Relational acyclicity. Alternatively, acyclicity may be imposed on the relational dimension. A TBox $\mathcal{T}$ is *relationally acyclic* if the following conditions hold:

- for $A \sqsubseteq \exists r.B \in \mathcal{T}$ or $\exists r.B \sqsubseteq A \in \mathcal{T}$, we have $\lambda(A) < \lambda(B)$;
- for $A \sqcap A' \sqsubseteq B \in \mathcal{T}$, we have $\lambda(A) = \lambda(A') = \lambda(B)$;
- for $A \sqsubseteq \bigcirc^n B \in \mathcal{T}$, we have $\lambda(A) = \lambda(B)$.

Query answering with respect to relationally acyclic TBoxes is k-EXPSPACE-complete in combined complexity, where k is the number of different ranks among $\mathsf{N_C}(\mathcal{T})$. In data complexity, it is NC^1-complete.

Our study starts from the question whether every $\mathcal{TEL}^{\circ}$ TBox, without any syntactic restriction, is ultimately periodic. To answer it, we employ conjunctive grammars.

3.4 Future $\mathcal{TEL}^{\circ}$ and unary conjunctive grammars

In this section, we prove two theorems that establish our key result: $\{\{n \in \mathbb{N} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\} \mid \mathcal{T}\ \mathcal{TEL}^{\circ}_{future}$ TBox $, A, B \in \mathsf{N_C}\}$ is the set of Parikh images of unary conjunctive languages. This will allow us to apply Theorems 2.4 and 2.6 to analyse ultimate periodicity of $\mathcal{TEL}^{\circ}$ TBoxes.

Theorem 3.6 (TBoxes to Grammars)**.** *For every $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}$, one can construct in polynomial time a unary conjunctive grammar $G_{\mathcal{T}} = (N, \{c\}, R)$ such that for any $A, B \in \mathsf{N_C}(\mathcal{T})$, there is $\mathcal{N}_{AB} \in N$ such that $c^n \in L_{G_{\mathcal{T}}}(\mathcal{N}_{AB})$ iff $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$.*

Given a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}$, we sketch the construction of $G_{\mathcal{T}}$. The first step is to ensure that every role name in $\mathcal{T}$ can be treated as rigid. For each $C \in \mathsf{N_C}(\mathcal{T})$, $r \in \mathsf{N_R^{loc}}(\mathcal{T})$, introduce a pair of concept names C_r, C'_r not appearing in $\mathcal{T}$ and let $\mathcal{T}_{rig}$ be obtained from $\mathcal{T}$ as follows:

1. for each $r \in \mathsf{N_R^{loc}}(\mathcal{T})$, substitute every $A \sqsubseteq \exists r.B \in \mathcal{T}$ with

$$A \sqsubseteq \exists r.B_r \qquad\qquad B_r \sqsubseteq B \tag{3.12}$$

2. for each C_r, substitute every $\exists r.A \sqsubseteq B \in \mathcal{T}$ with

$$A \sqcap C_r \sqsubseteq A'_r \qquad\qquad \exists r.A'_r \sqsubseteq B \tag{3.13}$$

3. substitute each $r \in \mathsf{N_R^{loc}}(\mathcal{T})$ with a new $r' \in \mathsf{N_R^{rig}}$.

Intuitively, in a derivation using $\mathcal{T}_{rig}$, a fact $B_r(b, n)$ created from some $A(a, n)$ and $A \sqsubseteq \exists r.B_r$ guards the "locality" of $r(a, b, n)$ at time n. Any application of a derivation rule of form (3.10) using a fact of the form $r(a, b, k)$, and hence any effect of b on a, is only possible using some $\exists r.A'_r \sqsubseteq C$ and $A'_r(b, k)$, and thus is limited to $k = n$, since $A'_r(b, k)$ can only be derived using $A \sqcap B_r \sqsubseteq A'_r$ and $B_r(b, n)$. The translation from $\mathcal{T}$ to $\mathcal{T}_{rig}$ is polynomial, and it is not hard to prove the following lemma.

Lemma 3.7. *Let $\mathcal{T}$ be a $\mathcal{TEL}^{\circ}$ TBox. For any $A, B \in \mathsf{N_C}(\mathcal{T})$ and $n \in \mathbb{Z}$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ if and only if $\mathcal{T}_{rig} \models A \sqsubseteq \bigcirc^n B$.*

Definition 3.8. *Given a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}$, $G_{\mathcal{T}} = (N_{\mathcal{T}}, \{c\}, R_{\mathcal{T}})$, where $N_{\mathcal{T}} = \{\mathcal{N}_{AB} \mid A, B \in \mathsf{N_C}(\mathcal{T}_{rig})\}$ and $R_{\mathcal{T}}$ contains exactly:*

$$\mathcal{N}_{AB} \to \varepsilon, \qquad \text{for } A \sqsubseteq B \in \mathcal{T}_{rig} \text{ or } A = B \tag{3.14}$$

$$\mathcal{N}_{AB} \to c^n, \qquad \text{for } A \sqsubseteq \bigcirc^n B \in \mathcal{T}_{rig},\ n > 0 \tag{3.15}$$

$$\mathcal{N}_{AB} \to \mathcal{N}_{AC} \,\&\, \mathcal{N}_{AD}, \qquad \text{for } \begin{array}{l} A \in \mathsf{N_C}(\mathcal{T}_{rig}), \\ C \sqcap D \sqsubseteq B \in \mathcal{T}_{rig} \end{array} \tag{3.16}$$

$$\mathcal{N}_{AB} \to \mathcal{N}_{CD}, \qquad \text{for } \left\{ \begin{array}{l} A \sqsubseteq \exists r.C \\ \exists r.D \sqsubseteq B \end{array} \right\} \subseteq \mathcal{T}_{rig} \tag{3.17}$$

$$\mathcal{N}_{AB} \to \mathcal{N}_{AC}\,\mathcal{N}_{CB}, \qquad \text{for } A, B, C \in \mathsf{N_C}(\mathcal{T}_{rig}) \tag{3.18}$$

Intuitively, for every pair of concept names $A, B \in \mathsf{N_C}(\mathcal{T}_{rig})$, $G_{\mathcal{T}}$ encodes every possible way of deriving $B(a, n)$ from $\{A(a, 0)\} \cup \mathcal{T}_{rig}$: either directly (3.14, 3.15), or by obtaining $C(a, n)$ and $D(a, n)$ that together give $B(a, n)$ (3.16), or by going through a null (3.17), or through an intermediate point $C(a, k)$, $0 \leqslant k \leqslant n$ (3.18). One can show that a derivation witnessing $(\mathcal{T}, \{A(a, 0)\}) \vdash B(a, n)$ using rules of the form (3.7)–(3.11) corresponds to a derivation for $G_{\mathcal{T}} \vdash \mathcal{N}_{AB}(c^n)$, and vice versa. Theorem 3.6 follows by Lemma 3.7, Proposition 3.3 and definition of $L_{G_{\mathcal{T}}}(\mathcal{N}_{AB})$.

Example 3.9 (Ex. 3.4 cont'd)**.** *Recall that $\mathcal{T} \models \mathsf{Prof} \sqsubseteq \bigcirc^3 \mathsf{Happy}$. Figure 3.2 illustrates the derivations witnessing $G_{\mathcal{T}} \vdash \mathcal{N}_{\mathsf{ProfProf}}(c^3)$ and $G_{\mathcal{T}} \vdash \mathcal{N}_{\mathsf{ProfProud}}(c^3)$. We obtain $G_{\mathcal{T}} \vdash \mathcal{N}_{\mathsf{ProfHappy}}(c^3)$ using the rule $\mathcal{N}_{\mathsf{ProfHappy}} \to \mathcal{N}_{\mathsf{ProfProf}} \,\&\, \mathcal{N}_{\mathsf{ProfProud}}$. One can further check that $L_{G_{\mathcal{T}}}(\mathcal{N}_{\mathsf{ProfHappy}}) = \{c^{3+n} \mid n \in \mathbb{N}\}$, since $\mathcal{N}_{\mathsf{ProfHappy}} \to \mathcal{N}_{\mathsf{ProfProf}}\mathcal{N}_{\mathsf{ProfHappy}}$ is in $R_{\mathcal{T}}$ and $G_{\mathcal{T}} \vdash \mathcal{N}_{\mathsf{ProfProf}}(c^n)$ for every n.*

We now turn our attention to the converse translation.

Theorem 3.10 (Grammars to TBoxes)**.** *For every unary conjunctive grammar $G = (N, \{c\}, R)$, one can construct in polynomial time a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}_G$ and $A \in \mathsf{N_C}(\mathcal{T}_G)$, such that for every $\mathcal{B} \in N$ there is $B \in \mathsf{N_C}(\mathcal{T}_G)$ such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B$ iff $c^n \in L_G(\mathcal{B})$.*

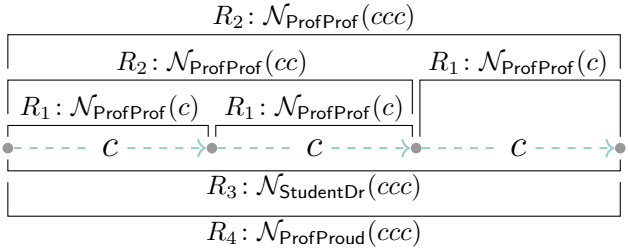


Figure 3.2: Derivations witnessing $G_{\mathcal{T}} \vdash \mathcal{N}_{\mathsf{ProfProf}}(c^3)$, in the upper part, and $G_{\mathcal{T}} \vdash \mathcal{N}_{\mathsf{ProfProud}}(c^3)$, in the lower part. Intuitively, each symbol c stands for a step forward in time (cf. Figure 3.1). The grammar rule of $G_{\mathcal{T}}$ used to obtain each proposition is given in the following denotation: $R_1 \colon \mathcal{N}_{\mathsf{ProfProf}} \to c$, $R_2 \colon \mathcal{N}_{\mathsf{ProfProf}} \to \mathcal{N}_{\mathsf{ProfProf}}\mathcal{N}_{\mathsf{ProfProf}}$, $R_3 \colon \mathcal{N}_{\mathsf{StudentDr}} \to c^3$, $R_4 \colon \mathcal{N}_{\mathsf{ProfProud}} \to \mathcal{N}_{\mathsf{StudentDr}}$.

Let G be a unary conjunctive grammar with nonterminals $N = \{\mathcal{B}_1, \dots, \mathcal{B}_m\}$. Without loss of generality, we assume that its rules are of the forms

$$\mathcal{B}_i \to \varepsilon \tag{3.19}$$
$$\mathcal{B}_i \to c^n, \quad n > 0 \tag{3.20}$$
$$\mathcal{B}_i \to \alpha_1 \tag{3.21}$$
$$\mathcal{B}_i \to \alpha_1 \,\&\, \alpha_2 \tag{3.22}$$

where α_1, α_2 are nonempty strings of nonterminals. Indeed, every unary grammar can be converted to this form in polynomial time.

Fix concept names $A, B_1, \dots, B_m$, and, for each $\alpha_l = \mathcal{B}_{i_1} \dots \mathcal{B}_{i_k}$ that occurs in the rules of G, introduce concept names $C_{i_j \dots i_k}$ and rigid role names $r_{i_j \dots i_k}$, for $1 \leqslant j < k$. Let $\mathcal{J}$ denote the set of number sequences $i_j \dots i_k$ appearing in the subscripts of these symbols. We use the symbol ι to denote the elements of $\mathcal{J}$, and write $i\iota$ to mean the sequence obtained from ι by appending i in the beginning.

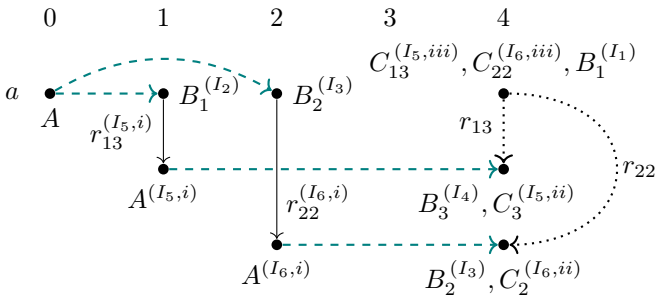


Figure 3.3: Derivation witnessing $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_1(a,4)$. The concept inclusions used to obtain facts are given in superscripts.

Moreover, let $\iota(\alpha_l)$ denote exactly the sequence $i_1 \dots i_k$.

Definition 3.11. *Given $G = (N, \{c\}, R)$, with $N = \{\mathcal{B}_1, \dots, \mathcal{B}_m\}$ and rules of the forms* (3.19)–(3.22)*, $\mathcal{T}_G$ contains exactly the following concept inclusions.*

$A \sqsubseteq B_i$,	*for each rule of the form* (3.19)	(3.19^*)
$A \sqsubseteq \bigcirc^n B_i$,	*for each rule of the form* (3.20)	(3.20^*)
$C_{\iota(\alpha_1)} \sqsubseteq B_i$,	*for each rule of the form* (3.21)	(3.21^*)
$C_{\iota(\alpha_1)} \sqcap C_{\iota(\alpha_2)} \sqsubseteq B_i$,	*for each rule of the form* (3.22)	(3.22^*)
$B_i \sqsubseteq \exists r_{i\iota} \,.\, A$,	*for* $i\iota \in \mathcal{J}$	(23^*)
$\exists r_{i\iota} \,.\, C_\iota \sqsubseteq C_{i\iota}$	*for* $\iota,\ i\iota \in \mathcal{J}$	(24^*)
$B_i \sqsubseteq C_i$	*for* $i \in \{1, \dots, m\}$ *s. t.* $i \in \mathcal{J}$	(25^*)

We show that $c^n \in L_G(\mathcal{B}_i)$ iff $\mathcal{T} \models A \sqsubseteq \bigcirc^n B_i$. We first illustrate this on an example.

Example 3.12. *Consider the grammar G defined by the following rules (subset of those presented in Example 2.5 such that $L_G(\mathcal{B}_1) =$*

$\{c, c^4\}$*) and the corresponding concept inclusions of* $\mathcal{T}_G$*:*

$$\begin{array}{llll} \mathcal{B}_1 & \to & \mathcal{B}_1\mathcal{B}_3 \;\&\; \mathcal{B}_2\mathcal{B}_2 & C_{13} \sqcap C_{22} \sqsubseteq B_1 \quad (I_1) \\ \mathcal{B}_1 & \to & c & A \sqsubseteq \bigcirc B_1 \quad (I_2) \\ \mathcal{B}_2 & \to & cc & A \sqsubseteq \bigcirc^2 B_2 \quad (I_3) \\ \mathcal{B}_3 & \to & ccc & A \sqsubseteq \bigcirc^3 B_3 \quad (I_4) \end{array}$$

Additionally, $\mathcal{T}_G$ *contains:*

$$B_1 \sqsubseteq \exists r_{13}.A \;\; (i) \qquad B_3 \sqsubseteq C_3 \;\; (ii) \qquad \exists r_{13}.C_3 \sqsubseteq C_{13} \;\; (iii) \qquad (I_5)$$
$$B_2 \sqsubseteq \exists r_{22}.A \;\; (i) \qquad B_2 \sqsubseteq C_2 \;\; (ii) \qquad \exists r_{22}.C_2 \sqsubseteq C_{22} \;\; (iii) \qquad (I_6)$$

Then, we derive $B_1(a, 4)$ *from* $\mathcal{T}_G \cup \{A(a, 0)\}$ *as shown in Figure 3.3.*

More generally, consider a rule of the form (3.21): $\mathcal{B}_i \to \mathcal{B}_{i_1} \dots \mathcal{B}_{i_k}$. It states that $c^n \in L(\mathcal{B}_i)$ if $n = n_1 + \dots + n_k$ and $c^{n_j} \in L(\mathcal{B}_{i_j})$, $1 \leqslant j \leqslant k$. The right-hand side of the rule is represented in $\mathcal{T}_G$ by $C_{i_1 \dots i_k}$, and whenever $C_{i_1 \dots i_k}(e, \ell)$ is derived, $B_i(e, \ell)$ follows by the corresponding concept inclusion (3.21*). The derivation is performed in steps. Starting from $A(a, 0)$, we derive $B_{i_1}(a, n_1)$, and then go, by $B_{i_1} \sqsubseteq \exists r_{i_1 \dots i_k}.A$ (23*), to a new null, b, where the process restarts, recursively, from $A(b, n_1)$ targeting $C_{i_2 \dots i_k}$. This fact is stored in the index of the role name $r_{i_1 \dots i_k}$, so when $C_{i_2 \dots i_k}(b, n_1 + n')$ is inferred, and only at that point, it is lifted up to $C_{i_1 \dots i_k}(a, n_1 + n')$ by $\exists r_{i_1 \dots i_k}.C_{i_2 \dots i_k} \sqsubseteq C_{i_1 \dots i_k}$ (24*). The inclusion $B_i \sqsubseteq C_i$ (25*) ends the recursion. Lemma 3.13 formalises this intuition and is proved by induction on k.

Lemma 3.13. *For* $\iota = i_1 \dots i_k \in \mathcal{J}$, $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_\iota$ *if and only if* $n = n_1 + \dots + n_k$*, such that* $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ *for* $1 \leqslant j \leqslant k$.

Concept inclusions of form (3.22*) just generalise this to the case of conjunction. We prove the next lemma by showing that the existence of a derivation witnessing $\mathcal{T}_G \vdash A \sqsubseteq \bigcirc^n B_i$ implies the existence of a derivation witnessing $G \vdash \mathcal{B}_i(c^n)$, and vice versa, by induction on the derivation length. This finalises the proof of Theorem 3.10.

Lemma 3.14. *$\mathcal{T}_G \models A \sqsubseteq \circ^n B_i$ if and only if $G \vdash \mathcal{B}_i(c^n)$, for $i \in \{1, \dots, m\}$.*

3.5 Linear $\mathcal{TEL}^{\circ}$ and context-free grammars

If a TBox $\mathcal{T}$ belongs to both $\mathcal{TEL}^{\circ}_{future}$ and $\mathcal{TEL}^{\circ}_{lin}$, the grammar $G_{\mathcal{T}}$ provided by Definition 3.8 in the proof of Theorem 3.6 does not contain rules of type (3.16), and is thus context-free. In this section, we prove a similar result about the linear fragment in general, which is used later in Section 3.6 for query answering relative to $\mathcal{TEL}^{\circ}_{lin}$ TBoxes.

Theorem 3.15. *For every $\mathcal{TEL}^{\circ}_{lin}$ TBox $\mathcal{T}$, there exists a context-free grammar $\Gamma_{\mathcal{T}} = (N, \{c, d\}, R)$, of size polynomial in $|\mathcal{T}|$, such that for any $A, B \in \mathsf{N_C}(\mathcal{T})$, there is $\mathcal{N}_{AB} \in N$ such that $\mathcal{T} \models A \sqsubseteq \circ^n B$ iff there exists $w \in L_{\Gamma_{\mathcal{T}}}(\mathcal{N}_{AB})$ with $\#c(w) - \#d(w) = n$.*

To reuse the ideas of the proof of Theorem 3.6, we would need to get rid of local role names that occur in $\mathcal{T}$. However, we cannot rely on the construction of $\mathcal{T}_{rig}$ as in Section 3.4, since it introduces concept inclusions of the form $A \sqcap C_r \sqsubseteq A'_r$, so that $\mathcal{T}_{rig}$ does not belong to $\mathcal{TEL}^{\circ}_{lin}$ even if $\mathcal{T}$ does. We treat separately the case where $\mathcal{T}$ does not feature any local role name and the case where it does.

3.5.1 The case of rigid role names only

Suppose $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}_{lin}$ TBox such that all role names in $\mathcal{T}$ are rigid. We construct a grammar $\Gamma_{\mathcal{T}} = (N, \{c, d\}, R)$ in a similar way as $G_{\mathcal{T}}$ in Section 3.4.

Definition 3.16. *Given a $\mathcal{TEL}^{\circ}_{lin}$ TBox $\mathcal{T}$ such that $\mathsf{N_R}(\mathcal{T}) \subseteq \mathsf{N_R^{rig}}$, $\Gamma_{\mathcal{T}} = (N_{\mathcal{T}}, \{c, d\}, R_{\mathcal{T}})$, where $N_{\mathcal{T}} = \{\mathcal{N}_{AB} \mid A, B \in \mathsf{N_C}(\mathcal{T})\}$ and $R_{\mathcal{T}}$ contains exactly the rules defined by* (3.14), (3.15), (3.17), (3.18)

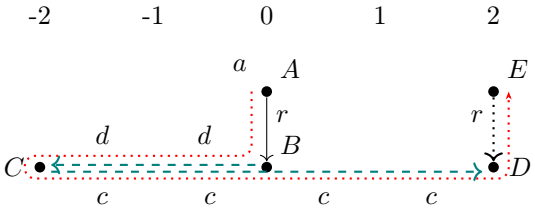


Figure 3.4: A derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash E(a,2)$ and the corresponding word *ddcccc* can be read along the dotted line.

with $\mathcal{T}_{rig} = \mathcal{T}$*, as well as the following rules.*

$$\mathcal{N}_{AB} \quad \to \quad d^{\,|n|}, \qquad \text{for } A \sqsubseteq \bigcirc^n B \in \mathcal{T},\ n < 0 \qquad (3.15^*)$$

In a word $w \in \{c,d\}^*$, a symbol c corresponds to a step forwards in time, and a symbol d to a step backwards. Otherwise, the intuition behind $\Gamma_{\mathcal{T}}$ is the same as that given for $G_{\mathcal{T}}$ in Section 3.4. For every derivation witnessing $\Gamma_{\mathcal{T}} \vdash \mathcal{N}_{AB}(w)$, we can construct a corresponding derivation of $B(a, \#c(w) - \#d(w))$ from $\mathcal{T} \cup \{A(a,0)\}$. Conversely, from a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$, we obtain a word w such that $\Gamma_{\mathcal{T}} \vdash \mathcal{N}_{AB}(w)$ as follows: whenever a derivation rule uses a concept inclusion of the form $A' \sqsubseteq \bigcirc^k B'$, we write c^k if $k \geqslant 0$, and $d^{\,|k|}$ when $k < 0$.

Lemma 3.17. *If* $\mathsf{N_R}(\mathcal{T}) \subseteq \mathsf{N_R^{rig}}$*, for any* $A, B \in \mathsf{N_C}(\mathcal{T})$*,* $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ *iff there exists* $w \in L_{\Gamma_{\mathcal{T}}}(\mathcal{N}_{AB})$ *with* $\#c(w) - \#d(w) = n$*.*

Example 3.18. *Consider a* $\mathcal{TEL}^{\circ}_{lin}$ *TBox* $\mathcal{T}$ *containing the following concept inclusions, with* $r \in \mathsf{N_R^{rig}}$*. Figure 3.4 shows a derivation witnessing* $(\mathcal{T}, \{A(a,0)\}) \vdash E(a,2)$ *and the corresponding word.*

$$A \sqsubseteq \exists r.B \qquad B \sqsubseteq \bigcirc^{-2} C \qquad C \sqsubseteq \bigcirc^4 D \qquad \exists r.D \sqsubseteq E$$

Lemma 3.17 cannot be extended beyond $\mathcal{TEL}^{\circ}_{lin}$ because different words, say *ddcccc* and *cdcccd*, correspond to the same "shift" in time

(here, 2), but the semantics of rules of type (3.16) treats them as different: if $\mathcal{N} \to ddcccc$ and $\mathcal{M} \to cdcccd$, then $\mathcal{N}\&\mathcal{M}$ generates an empty language, while we would need $\mathcal{N}\&\mathcal{M}$ to generate cc.

3.5.2 The case of both rigid and local role names

For this case, we provide a translation from $\mathcal{T}$ to $\Gamma_{\mathcal{T}}$ which is not constructive but nevertheless guarantees the existence of $\Gamma_{\mathcal{T}}$ and the bound on its size stated by Theorem 3.15. To actually build $\Gamma_{\mathcal{T}}$, one needs to compute the set $\{A \sqsubseteq B \mid \mathcal{T} \models A \sqsubseteq B\}$, and we do not provide any recipe for that when $\mathcal{T}$ features local role names. However, Theorem 3.15 is enough to show ultimate periodicity of $\mathcal{TEL}^{\circ}_{lin}$ TBoxes, as we do in the next section.

Consider a $\mathcal{TEL}^{\circ}_{lin}$ TBox $\mathcal{T}$ and let $\mathcal{T}_0$ be obtained from $\mathcal{T}$ by removing all concept inclusions that use local role names. Then, let

$$\mathcal{T}^{lin}_{rig} = \mathcal{T}_0 \cup \{A \sqsubseteq B \mid \mathcal{T} \models A \sqsubseteq B\}. \tag{3.26}$$

Intuitively, in a derivation using $\mathcal{T}$, if a rule of form (3.11) produces a fact $r(a, b, n)$ with some $r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{loc}}$, and later a rule of form (3.10) uses $r(a, b, n')$ to "propagate back" the consequences of facts derived about b on a, the locality of r implies that $n = n'$. A derivation that uses $\mathcal{T}^{lin}_{rig}$ "skips" the derivation between these two rule applications, immediately inferring a new fact for a at time n.

Example 3.19. *Let $r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{loc}}$ and $\mathcal{T}$ contain the concept inclusions:*

$$A \sqsubseteq \bigcirc B \qquad F \sqsubseteq \bigcirc G \qquad C \sqsubseteq \bigcirc^{-3} D \qquad D \sqsubseteq \bigcirc^{3} E$$
$$B \sqsubseteq \exists r.C \qquad \exists r.E \sqsubseteq F$$

Then $\mathcal{T} \models B \sqsubseteq F$, and $\mathcal{T}^{lin}_{rig}$ contains the concept inclusions in the first line and $B \sqsubseteq F$. Figure 3.5 illustrates two derivations of $G(a, 2)$: one from $\{A(a, 0)\} \cup \mathcal{T}$, and one from $\{A(a, 0)\} \cup \mathcal{T}^{lin}_{rig}$.

In general, the following lemma holds.

Lemma 3.20. *Let $\mathcal{T}$ be a $\mathcal{TEL}^{\circ}_{lin}$ TBox. For any $A, B \in \mathsf{N}_{\mathsf{C}}(\mathcal{T})$ and $n \in \mathbb{Z}$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ if and only if $\mathcal{T}^{lin}_{rig} \models A \sqsubseteq \bigcirc^n B$.*

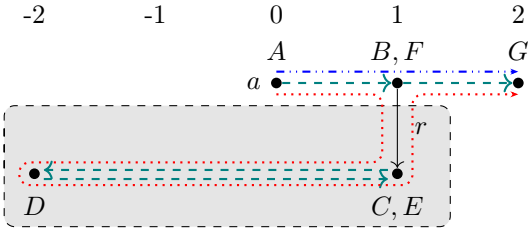


Figure 3.5: A derivation of $G(a, 2)$ from $\{A(a, 0)\}$ and the TBox $\mathcal{T}$ of Example 3.19 goes along the dotted line. A derivation with $\mathcal{T}_{rig}^{lin}$ uses $B \sqsubseteq F$ and skips the part in the grey box, passing along the dash-dotted line.

The proof of Theorem 3.15 uses Lemma 3.20 and the facts that $\mathsf{N}_{\mathsf{R}}(\mathcal{T}_{rig}^{lin}) \subseteq \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}}$ and that $|\mathcal{T}_{rig}^{lin}|$ is polynomial in $|\mathcal{T}|$ to assume that $\mathcal{T}$ uses only rigid roles, and concludes via Lemma 3.17 and the polynomial time construction of $\Gamma_{\mathcal{T}}$ in Definition 3.16.

3.6 Query answering relative to $\mathcal{TEL}^{\circ}$ TBoxes

We now draw the consequences of the correspondences between $\mathcal{TEL}^{\circ}_{future}$ or $\mathcal{TEL}^{\circ}_{lin}$ TBoxes and grammars for answering atomic queries mediated by $\mathcal{TEL}^{\circ}$ TBoxes.

3.6.1 Undecidability results

First, we close negatively the question of $\mathcal{TEL}^{\circ}$ ultimate periodicity.

Theorem 3.21. *The following statements hold.*

(i) There exists a $\mathcal{TEL}^{\circ}_{future}$ TBox which is not ultimately periodic.

(ii) It is undecidable to check if the set $\{n \in \mathbb{N} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$ *is semilinear for a* $\mathcal{TEL}^{\circ}_{future}$ *TBox* $\mathcal{T}$ *and* $A, B \in \mathsf{N_C}(\mathcal{T})$.

Proof. (i) Let G be the grammar of Example 2.5. By Theorem 3.10, there exists a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}_G$ and $A, B \in \mathsf{N_C}(\mathcal{T}_G)$ such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B$ iff $c^n \in L_G(\mathcal{N}_1)$. Hence, $\{n \in \mathbb{N} \mid \mathcal{T}_G \models A \sqsubseteq \bigcirc^n B\} = \{4^k \mid k \in \mathbb{N}\}$, which is not semilinear.

(ii) We proceed by a reduction from the problem of deciding whether $L(G)$ is regular for a unary conjunctive grammar G, which is undecidable by point *(iii)* of Theorem 2.6. Given a unary conjunctive grammar G, by Theorem 3.10, one can construct a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}_G$ and $A, B \in \mathsf{N_C}(\mathcal{T}_G)$ such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B$ iff $c^n \in L_G(\mathcal{S})$, where $\mathcal{S}$ is the start symbol of G, so that $L(G) = L_G(\mathcal{S})$. Hence, $\{n \in \mathbb{N} \mid \mathcal{T}_G \models A \sqsubseteq \bigcirc^n B\}$ is the Parikh image of $L(G)$. Thus, by Theorem 2.4, $L(G)$ is regular iff this set is semilinear. □

Second, we obtain some undecidability results for query answering relative to $\mathcal{TEL}^{\circ}$ TBoxes.

Theorem 3.22. *Query answering is undecidable, for combined complexity, with* $\mathcal{TEL}^{\circ}$ *TBoxes and with* $\mathcal{TEL}^{\circ}_{future}$ *TBoxes extended with rigid concept names.*

Proof. By Theorem 2.6 *(i)*, checking if $L(G) = \emptyset$ for a given unary conjunctive grammar G is undecidable. We reduce this problem to query answering with respect to TBoxes. As in the proof of Theorem 3.21 *(ii)*, given a unary conjunctive grammar G, one can construct a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}_G$ and $A, B \in \mathsf{N_C}(\mathcal{T}_G)$ such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B$ iff $c^n \in L(G)$. Let C be a concept name not appearing in $\mathcal{T}_G$. We construct $\mathcal{T}'_G$ such that $L(G) \neq \emptyset$ iff $(\mathcal{T}'_G, \{A(a,0)\}) \models C(a,0)$ as follows. For the $\mathcal{TEL}^{\circ}$ case, let $\mathcal{T}'_G = \mathcal{T}_G \cup \{B \sqsubseteq C, C \sqsubseteq \bigcirc^{-1} C\}$, and for $\mathcal{TEL}^{\circ}_{future}$ with rigid concept names, let C be rigid and $\mathcal{T}'_G = \mathcal{T}_G \cup \{B \sqsubseteq C\}$. □

It is known that rigid concept names can be simulated with rigid role names if the language allows for the concept $\top$ (which is such that $\top^{\mathfrak{I},i} = \Delta_{\mathfrak{I}}$ for every $\mathfrak{I} = (\Delta_{\mathfrak{I}}, \cdot^{\mathfrak{I}})$ and $i \in \mathbb{Z}$, and is not allowed in

the original definition of $\mathcal{TEL}^{\circ}$ by Gutiérrez-Basulto et al. [2016a]): adding to the TBox $C \equiv \exists r.\top$ for an $r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}} \setminus \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}}(\mathcal{T})$ makes C rigid. It thus follows from Theorem 3.22 that query answering is undecidable also if one extends $\mathcal{TEL}^{\circ}_{future}$ with $\top$.

Note that Theorem 3.22 does not imply undecidability for data complexity. However, it implies that even having fixed a TBox $\mathcal{T}$, there is no "computational" way to obtain an algorithm for query answering relative to that $\mathcal{T}$.

3.6.2 Decidability results

We first show that query answering is decidable with $\mathcal{TEL}^{\circ}_{future}$ TBoxes.

Theorem 3.23. *Query answering relative to $\mathcal{TEL}^{\circ}_{future}$ TBoxes is* P*-complete, both for combined and data complexity.*

Proof sketch. The lower bounds hold already for the description logic $\mathcal{EL}$ (without temporal operators) [Calvanese et al., 2013]. For the upper bounds, we provide a polynomial reduction from the problem of deciding whether $(\mathcal{T}, \mathcal{D}) \models A(a, n)$ to that of checking whether a word belongs to the language of a conjunctive grammar, which can be tested in polynomial time (Theorem 2.3). Our reduction builds a $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}' \cup \mathcal{T}_{\mathcal{D}}$, an assertion $C_a(a, l)$ and a concept name A_n such that $(\mathcal{T}, \mathcal{D}) \models A(a, n)$ iff $(\mathcal{T}' \cup \mathcal{T}_{\mathcal{D}}, \{C_a(a, l)\}) \models A_n(a, n)$, then uses Proposition 3.3 and Theorem 3.6 to conclude. The idea is to encode all information about a in $\mathcal{D}$ into $C_a(a, l)$ thanks to $\mathcal{T}_{\mathcal{D}}$.

Let $l, m \in \mathbb{Z}$ be the least and the greatest timestamps appearing in $\mathcal{D}$. We introduce concept names $\{C_a \mid a \in \mathrm{obj}(\mathcal{D})\}$ and $\{A_k \mid A \in \mathsf{N}_{\mathsf{C}}(\mathcal{T}), l \leqslant k \leqslant m+1\}$, and role names $\{\rho^r_{ab} \in \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}} \mid a, b \in \mathrm{obj}(\mathcal{D}), r \in \mathsf{N}_{\mathsf{R}}(\mathcal{T})\}$. For the convenience of notation, we write A_k for all $k \geqslant l$, assuming that $A_k = A_{m+1}$ when $k > m$. The TBox $\mathcal{T}'$ contains the following inclusions, for all $k \in \{l, \dots, m+1\}$.

$$A_k \sqsubseteq \bigcirc^s B_{k+s} \qquad \text{for } A \sqsubseteq \bigcirc^s B \in \mathcal{T} \tag{3.27}$$

$$A_k \sqcap A'_k \sqsubseteq B_k \qquad \text{for } A \sqcap A' \sqsubseteq B \in \mathcal{T} \tag{3.28}$$
$$A_k \sqsubseteq \exists r.B_k \qquad \text{for } A \sqsubseteq \exists r.B \in \mathcal{T} \tag{3.29}$$
$$\exists r.A_k \sqsubseteq B_k \qquad \text{for } \exists r.A \sqsubseteq B \in \mathcal{T} \tag{3.30}$$

Additionally, the TBox $\mathcal{T}_{\mathcal{D}}$ contains the following inclusions.

$$C_a \sqsubseteq \bigcirc^{k-l} A_k \qquad \text{for } A(a,k) \in \mathcal{D} \tag{3.31}$$
$$C_a \sqsubseteq \exists \rho^r_{ab} \,.\, C_b \qquad \text{for } r(a,b,\ell) \in \mathcal{D} \tag{3.32}$$
$$\exists \rho^r_{ab} \,.\, A_k \sqsubseteq B_k \qquad \text{for } \exists r.A \sqsubseteq B \in \mathcal{T}, r(a,b,\ell) \in \mathcal{D}, \text{ where } r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}} \text{ or } \ell = k. \tag{3.33}$$

Both $\mathcal{T}'$ and $\mathcal{T}_{\mathcal{D}}$ can be constructed in polynomial time w.r.t. $|\mathcal{T}| + |\mathcal{D}|$ and are expressed in $\mathcal{TEL}^{\circ}_{future}$ (since $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}_{future}$ TBox and $k - l \geqslant 0$ for every $A(a,k) \in \mathcal{D}$ by definition of l) and we show that $(\mathcal{T}, \mathcal{D}) \models A(a,n)$ iff $(\mathcal{T}' \cup \mathcal{T}_{\mathcal{D}}, \{C_a(a,l)\}) \models A_n(a,n)$. □

One could prove Theorem 3.23 without using grammars. Given $(\mathcal{T}, \mathcal{D})$, and $A(a,n)$, construct $(\mathcal{T}_n, \mathcal{D}')$ with $\mathcal{T}_n$ defined as $\mathcal{T}'$ above except that we use $k \in \{l, \ldots, n\}$ in the concept inclusions, and $\mathcal{D}' = \{A_k(b,k) \mid A(b,k) \in \mathcal{D}\} \cup \{r(b,c,k) \in \mathcal{D}\}$. Using the inability of $\mathcal{TEL}^{\circ}_{future}$ to reason backwards, one can show that $(\mathcal{T}, \mathcal{D}) \models A(a,n)$ iff $(\mathcal{T}_n, \mathcal{D}') \models A_n(a,n)$. Then, $\mathcal{T}_n$ is *temporally acyclic*, in the terminology of Gutiérrez-Basulto et al. [2016a], who prove that query answering relative to such TBoxes is in P, even with rigid concept names (this is not a contradiction with Theorem 3.22, since $\mathcal{T}_n$ is constructed for a fixed n, while in the proof of Theorem 3.22 rigid concept names are used to simulate an existential query of the form $\exists n.A(a,n)$). On the other hand, the technique we present here allows us to reuse existing algorithms and tools for conjunctive grammars: a parser generator *Whale Calf* [Okhotin, 2002] and an efficient parsing method tailored to *unary* conjunctive grammars [Okhotin and Reitwießner, 2012].

We also obtain positive results for the linear fragment. In the next theorem, we use the following measure for the "size" of semilinear sets. Let $\|u\| = |u_1| + \cdots + |u_n|$ for $\vec{u} \in \mathbb{Z}^n$. We define $\|\mathcal{L}\|$ as the

least number $\|\vec{b}\| + \|\vec{p}_1\| + \cdots + \|\vec{p}_l\|$ among all representations of a linear set $\mathcal{L}$ by an offset $\vec{b}$ and periods $\vec{p}_1, \ldots \vec{p}_l$, and $\|S\|$ as the least sum $\|\mathcal{L}_1\| + \cdots + \|\mathcal{L}_m\|$ among all representations of a semilinear set S as a union of linear sets. For an ultimately periodic TBox $\mathcal{T}$, we set $\|\mathcal{T}\| = \max_{A,B \in \mathsf{N_C}(\mathcal{T})}(\|\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}\|)$.

Theorem 3.24. *The following statements hold.*

(i) Every $\mathcal{TEL}^{\bigcirc}_{lin}$ TBox $\mathcal{T}$ is ultimately periodic, $\|\mathcal{T}\| \leqslant 2^{\mathrm{poly}(|\mathcal{T}|)}$.

(ii) Query answering with $\mathcal{TEL}^{\bigcirc}_{lin}$ TBoxes is NL*-complete, for data complexity.*

(iii) Query answering with $\mathcal{TEL}^{\bigcirc}_{lin}$ TBoxes without local role names is in ExpSpace*, for combined complexity.*

Proof sketch. For *(i)*, let $\mathcal{T}$ be a $\mathcal{TEL}^{\bigcirc}_{lin}$ TBox. By Theorem 3.15, there exists a context-free grammar $\Gamma_{\mathcal{T}} = (N, \{c, d\}, R)$ such that for any $A, B \in \mathsf{N_C}(\mathcal{T})$, there is $\mathcal{N}_{AB} \in N$ such that $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ iff there exists $w \in L_{\Gamma_{\mathcal{T}}}(\mathcal{N}_{AB})$ with $\#c(w) - \#d(w) = n$. Let $L = L_{\Gamma_{\mathcal{T}}}(\mathcal{N}_{AB})$. By Theorem 2.4, since $\Gamma_{\mathcal{T}}$ is context-free, the Parikh image $p(L) \subseteq \mathbb{N}^2$ of L (with alphabet ordered as c, d) is semilinear. Hence, for every $A, B \in \mathsf{N_C}(\mathcal{T})$, $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\} = \{n \in \mathbb{Z} \mid n = \#c(w) - \#d(w), w \in L\} = \{u_1 - u_2 \mid \vec{u} \in p(L)\}$ is semilinear, since it is the image of the semilinear set $p(L)$ under the linear mapping $\vec{u} \mapsto u_1 - u_2$. For the bound on $\|\mathcal{T}\|$, we use the methods introduced by Esparza et al. [2011] to establish that $\|p(L)\| = 2^{\mathrm{poly}(|\Gamma_{\mathcal{T}}|)}$, and further observe that $\|\{u_1 - u_2 \mid \vec{u} \in p(L)\}\| \leqslant 2\|p(L)\|$.

The lower bound in *(ii)* follows from the atemporal case Calvanese et al. [2013]. For the upper bounds, both in *(ii)* and *(iii)*, we provide a translation of $\mathcal{TEL}^{\bigcirc}_{lin}$ TBoxes to linear *datalog*$^{\bigcirc\Diamond}$ programs (cf. proof of [Gutiérrez-Basulto et al., 2016a, Theorem 5]).

Given a $\mathcal{TEL}^{\bigcirc}_{lin}$ TBox $\mathcal{T}$, let $\Phi_{\mathcal{T}}$ be a (linear) *datalog*$^{\bigcirc\Diamond}$ program defined as follows. For each $\exists r.A \sqsubseteq B \in \mathcal{T}$, it contains $B(x) \leftarrow r(x, y), A(y)$, and, if $r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{rig}}$, also $B(x) \leftarrow \Diamond r(x, y), A(y)$

and $B(x) \leftarrow \Diamond^- r(x,y), A(y)$. If $r(a,b,k) \in \mathcal{D}$ for $r \in \mathsf{N}_\mathsf{R}^\mathsf{rig}$, the facts $r(a,b,m)$, $m \in \mathbb{Z}$, are not derived explicitly, but are simulated by the latter two rules. Further, for every $A, B \in \mathsf{N}_\mathsf{C}(\mathcal{T})$, we take a representation of the semilinear set $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$ as a union of linear sets $\mathcal{L}_1 \cup \dots \cup \mathcal{L}_m$, $\mathcal{L}_i = \{b^i + k_1 p_1^i + \dots k_l p_l^i \mid k_1, \dots, k_l \in \mathbb{N}\}$. Then, $\Phi_\mathcal{T}$ contains the rules $F_i^{AB}(x) \leftarrow \bigcirc^{-b^i} A(x)$, $F_i^{AB}(x) \leftarrow \bigcirc^{-p_j^i} F_i^{AB}(x)$, $B(x) \leftarrow F_i^{AB}(x)$, for all $i \in \{1, \dots, m\}$. Thus, if $A(a,k) \in \mathcal{D}$, the fact $F_i^{AB}(a,m)$, and therefore $B(a,m)$ is derived for all m such that $m - k \in \mathcal{L}_i$. It can be shown that $(\mathcal{T}, \mathcal{D}) \models A(a,n)$ iff $(\Phi_\mathcal{T}, \mathcal{D}) \models A(a,n)$ and that $|\Phi_\mathcal{T}| \leqslant 2^{\mathrm{poly}(|\mathcal{T}|)}$.

Moreover, if $\mathsf{N}_\mathsf{R}(\mathcal{T}) \subseteq \mathsf{N}_\mathsf{R}^\mathsf{rig}$, $\Phi_\mathcal{T}$ can be built from $\mathcal{T}$ using the automata-theoretic construction of Esparza et al. [2011]. Points *(ii)* and *(iii)* then follow from Theorem 2.8. □

Remark 3.25. *Recall that we assume a unary encoding of numbers, as is customary in temporal reasoning and as it is done by Gutiérrez-Basulto et al. [2016a]. If sequences $\bigcirc \cdots \bigcirc$ were written as $\bigcirc^n$ with n encoded in binary, translating a TBox into a grammar would cause an exponential blow-up (since c^n is actually a word of length n), making our algorithms exponentially slower w.r.t. $|\mathcal{T}|$.*

3.7 Discussion

Connections between temporal logics, such as *LTL*, and formal languages (in particular, regular languages) are well-known [Vardi, 1996, Demri et al., 2016]. In this chapter, we established new ones, between the fragments $\mathcal{TEL}^\bigcirc_{lin}$ and $\mathcal{TEL}^\bigcirc_{future}$ of the temporal description logic $\mathcal{TEL}^\bigcirc$, on the one side, and context-free and unary conjunctive languages, on the other. Using these connections, we obtained several important results on $\mathcal{TEL}^\bigcirc$, both negative and positive, from the formal language theory.

In particular, $\mathcal{TEL}^\bigcirc_{future}$ TBoxes are in a one-to-one correspondence with unary conjunctive grammars (Theorems 3.6 and 3.10). Therefore, $\mathcal{TEL}^\bigcirc_{future}$ is not ultimately periodic (Theorem 3.21),

which is arguably unexpected, as its temporal component, *LTL*, is ultimately periodic [Sistla and Clarke, 1985], and its DL component, $\mathcal{EL}$, is such that every KB possesses a canonical model which has, informally speaking, a regular structure Kontchakov and Zakharyaschev [2014]. Moreover, atomic OMQ answering relative to $\mathcal{TEL}^{\circ}$ TBoxes is undecidable (Theorem 3.22). On the other hand, the same correspondence allows us to use parsing algorithms for conjunctive grammars as tools for OMQ answering relative to $\mathcal{TEL}^{\circ}_{future}$ TBoxes, leading to a drastic decrease of complexity, from undecidability to polynomial time, owed to a mere removal of the temporal operator $\bigcirc^{-}$ ("at the previous moment").[1] Despite the partial undecidability result of Theorem 3.21, it remains open if ultimate periodicity of $\mathcal{TEL}^{\circ}$ TBoxes is decidable, since for the corresponding problem—given a unary conjunctive grammar tell if all its nonterminals generate regular languages—no result is known.

The linear fragment, $\mathcal{TEL}^{\circ}_{lin}$, is connected to context-free grammars (Theorem 3.15). As a result, it is ultimately periodic, and enjoys considerably low data complexity of query answering (Theorem 3.24).

Language theorists may find interesting that every unary conjunctive grammar is transformable, in polynomial time, to the "normal form" of Definition 3.8, by applying first Theorem 3.10, then Theorem 3.6. Moreover, the grammars that correspond to "pure *LTL*" TBoxes (i.e., do not have any rule of form (3.17) in this normal form) are guaranteed to generate regular languages. This is, to the best of our knowledge, the first nontrivial sufficient condition for this property.

On the more theoretical side, it is possible that the TBox-grammar correspondence can be lifted to more expressive temporal description logics and more general classes of formal grammars (e.g. Boolean grammars [Okhotin, 2013]). On the applications side, this correspondence can be employed for developing a practical reasoner for $\mathcal{TEL}^{\circ}_{future}$.

[1] My supervisor still does not believe this is true.

Chapter 4

A temporal description logic with undecidable rewritability problem

4.1 Overview

We have seen in the previous chapter that a combination in $\mathcal{TEL}^{\circ}$ of rigid roles, existential restrictions, and both of the temporal operators $\bigcirc$ and $\bigcirc^{-}$ leads to undecidability of query answering relative to $\mathcal{TEL}^{\circ}$ TBoxes. It is a convincing example of how a temporal component may destroy nice computational properties of the base relational language—in this case, $\mathcal{EL}$.

Apart from $\mathcal{EL}$, the lightweight description logics include the *DL-Lite* family, designed to ensure that all ontology-mediated queries are FO-rewritable. Like $\mathcal{EL}$, standard *DL-Lite* logics only allow TBoxes. The distinguishing feature of the family is that existential restrictions are limited to be *unqualified*: one can say $A \sqsubseteq \exists r$, but not $A \sqsubseteq \exists r.B$. That is, $A(a)$ only implies that there exists an

outgoing role r, but nothing is known about its other end. Unlike $\mathcal{TEL}^{\circ}$, *DL-Lite* admits $\bot$ in the right-hand sides of concept inclusions, thus equalising the combined complexity of query answering with the complexity of consistency checking for knowledge bases.

Work has also been done on extending *DL-Lite* with non-Horn RBoxes [Kontchakov et al., 2020]. Although ontology-mediated queries are no longer guaranteed to be FO-rewritable in this case, answering them is still decidable, albeit NExpTime-complete, for combined complexity.[1] We start by showing, in Section 4.3, that FO-rewritability is decidable for queries with full-fledged non-Horn TBoxes and RBoxes.

We then turn our attention to the temporal setting. Temporal versions of *DL-Lite* were extensively studied [Artale et al., 2014], and query answering relative to temporal *DL-Lite* TBoxes was shown to be PSpace-complete (combined) and NC^1-complete (data) [Artale et al., 2022]. The data complexity comes from the fact that every such query is FO(RPR)-rewritable.

Unrestricted temporal RBoxes notoriously lead to high computational complexity even if coupled with a lightweight TBox [Lutz et al., 2008, Artale et al., 2014, 2022].[2] Thus, any type of temporalised concept inclusions in *DL-Lite* with unguarded non-Horn temporalised role inclusions (say, $r \sqsubseteq u \sqcup \bigcirc v$) result in undecidability of consistency checking.

For Horn RBoxes, consistency checking is decidable in ExpSpace or PSpace depending on the available temporal operators, and FO(RPR)-rewritability (of all ontology-mediated queries) is restored [Artale et al., 2022].

We are interested in the complexity of *checking* rewritability in presence of non-Horn role inclusions. Note that where consistency checking is undecidable, checking rewritability into, say, FO($<$), is also undecidable. Indeed, suppose $Q = (\mathcal{O}, P)$, where P is a propo-

[1] Perhaps *DL-Heavy* would be a better name for a logic of this complexity, but we stick to the conventional terminology.

[2] See also Hodkinson et al. [2001] for general effects of applying temporal operators to relations of higher arities.

sitional symbol, is not FO(<)-rewritable, and $\mathcal{O}'$ does not use P and the relation symbols of $\mathcal{O}$. Then $Q' = (\mathcal{O} \cup \mathcal{O}', P)$ is FO(<)-rewritable (to $\top$) whenever $(\mathcal{O}', \varnothing)$ is inconsistent.

Our goal is to understand if a temporal *DL-Lite* logic can feature an undecidable rewritability checking in the less trivial case where consistency checking and query answering are decidable. With this in mind, we design a version of temporal *DL-Lite* that admits non-Horn temporalised concept inclusions, and only guarded $\Diamond$-role inclusions that take the form

$$r_1 \sqcap \dots \sqcap r_n \sqcap \Diamond^* r_{n+1} \sqcap \dots \Diamond^* r_k \;\sqsubseteq\; r_{k+1} \sqcup \dots \sqcup r_m, \tag{4.1}$$

where the r_i are role names or their inverses, $\Diamond^*$ stand for arbitrary sequences of $\Diamond$ and $\Diamond^-$, and $n \geqslant 1$.

Example 4.1. Imagine that we are modelling a transport network. Some passengers may like to go by plane in one direction but return by train (say, to safely bring back a selection of wines, cheeses and fine teas). To highlight such routes, one could use the following role inclusion:

$$\mathsf{flight} \sqcap \Diamond\, \mathsf{train}^- \sqsubseteq \mathsf{safeFlightConnection}, \tag{4.2}$$

where train^- denotes the inverse of the role train (connection) from one city to another. (We do not regard the roles flight and train as rigid because they depend on the season, day of a week, etc.) We may further partition the connections into certain classes, e.g., national and international, and infer properties of connected cities based on this classification. This is achieved by means of the following role inclusions:

$$\begin{aligned} &\mathsf{safeFlightConnection} \sqsubseteq \mathsf{national} \;\sqcup \mathsf{international}, \\ &\exists \mathsf{international} \sqsubseteq \mathsf{InternationalAirport}. \end{aligned} \tag{4.3}$$

Axiom (4.2) is an example of a *guarded* $\Diamond$-role inclusion, where the temporalised role $\Diamond\, \mathsf{train}^-$ is "guarded" by the role name flight. Thus, $\mathsf{safeFlightConnection}$ may be inferred at a time instant only when there exists a flight from one city to another and, sometime later, there is a return train. ⊣

Following the standard naming schema for languages in the temporal *DL-Lite* family [Artale et al., 2014], ours should go under the name

$$\mathit{TDL\text{-}Lite}^{\bigcirc\Diamond\Box/g\text{-}\Diamond}_{bool/bool}$$

That is, it is a temporal *DL-Lite* language that allows Boolean concept and role inclusions, and all the three types of temporal operators applied to concepts, while only the guarded $\Diamond$ type for roles. For sanity reasons, we will just refer to this language as

$$\mathit{TDL\text{-}Lite}^{4}$$

the temporal *DL-Lite* of Chapter 4.

We prove in Section 4.4 that consistency checking for $\mathit{TDL\text{-}Lite}^{4}$ knowledge bases can be done in ExpSpace, and is ExpTime-hard, while the complexity drops to PSpace if we disallow existential restrictions in the right-hand sides of concept inclusions. Automatically, the same bounds hold for combined complexity of query answering relative to $\mathit{TDL\text{-}Lite}^{4}$ ontologies. However, we show in Section 4.5 that rewritability (over a restricted database schema) of ontology-mediated queries in $\mathit{TDL\text{-}Lite}^{4}$ into FO($<$) or FO($<, \equiv$) is undecidable. Furthermore, rewritability into FO($<$, MOD), FO(RPR) or (linear) *datalog*(S, $<$) is undecidable modulo standard complexity-theoretic assumptions.

4.2 The $\mathit{TDL\text{-}Lite}^{4}$ language

Syntax. As before, $\mathsf{N_I}$, $\mathsf{N_C}$, and $\mathsf{N_R}$ stand for the sets of *individual names*, *concept names*, and *role names*, respectively. A *role* r is either a role name p or its inverse p^-. A *concept* C is either a concept name A or an unqualified existential restriction $\exists r$. A *concept inclusion* in $\mathit{TDL\text{-}Lite}^{4}$ takes the general TDL form (2.28):

$$\mathcal{O}^* C_1 \sqcap \dots \sqcap \mathcal{O}^* C_k \sqsubseteq \mathcal{O}^* C_{k+1} \sqcup \dots \sqcup \mathcal{O}^* C_{k+m},$$

where, as usual, $\mathcal{O}^*$ are arbitrary sequences of temporal operators $\bigcirc$, $\Diamond$, $\Box$ and their inverses. A *guarded $\Diamond$-role inclusion* (guarded $\Diamond$-RI) takes the form (4.1).

We work with *TBoxes*, finite sets of concept inclusions, and *RBoxes*, finite sets of guarded $\Diamond$-role inclusions. Taken together, they form an *ontology* $\mathcal{O}$ in *TDL-Lite*$^{\mathcal{A}}$.

We also consider the *flat* fragment that disallows concepts $\exists r$ on the right-hand side of concept inclusions, and the non-temporal fragment *DL-Lite*$^{\mathcal{A}}$, also known as *DL-Lite*$_{bool/bool}$ [Kontchakov et al., 2020], which is obtained by dropping the temporal operators from both concept and role inclusions.

Semantics are inherited from the general temporal description logic of Section 2.4.

As in the case of $\mathcal{TEL}^{\bigcirc}$, we will be interested in ontology-mediated queries of the form $(\mathcal{O}, A)$, where A is a concept name.

4.3 Deciding first-order rewritability for *DL-Lite*$^{\mathcal{A}}$ queries

We first establish that FO-rewritability of queries with non-temporal *DL-Lite*$^{\mathcal{A}}$ ontologies can be checked in 3NExpTime, thus showing that non-Horn role inclusions on their own do not cause undecidability of this problem.

The decision procedure relies on a translation of an ontology-mediated query to an equivalent monadic disjunctive datalog query. For the latter language, FO-rewritability is decidable in 2NExpTime. Our translation is similar to that of Feier et al. [2019], who studied FO-rewritability problem for the description logic $\mathcal{ALCI}$. In that logic, role inclusions have the simple form $r \sqsubseteq s$. In our case, dealing with non-Horn role inclusions results in an exponential increase in the size of the target datalog query, lifting the complexity one exponent up. We remind the reader that in the

non-temporal case we work with standard databases D instead of temporal databases $\mathcal{D}$.

Theorem 4.2. FO*-rewritability of queries with DL-Lite*^{4} *ontologies is decidable in* 3NExpTime.

In the rest of this section, we sketch the proof of this theorem. Fix an ontology-mediated query $Q = (\mathcal{O}, G)$, with $G \in \mathsf{N_C}$. We would like to obtain a monadic disjunctive datalog query (Π_Q, G) that is equivalent to Q in the sense that $(\mathcal{O}, D) \models A(a)$ if and only if $(\Pi_Q, D) \models G(a)$, for any temporal database D and $a \in \mathrm{obj}(D)$.

For the moment, we will use *constants* in place of some variables in the rules with the expected meaning. We will show later how to get rid of them. Suppose $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$. We encode the TBox $\mathcal{T}$ in a program $\Pi_\mathcal{T}$, and the RBox $\mathcal{R}$ in a program $\Pi_\mathcal{R}$, and the query itself in a goal rule Π_G.

Symbols. For every role name $p \in \mathsf{N_R}(\mathcal{O})$, we will use new (i.e. not already appearing in the ontology) concept names E_p and E_{p^-}. Given an ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$, a set of roles $R = \{r_1, \dots, r_k\} \subseteq \mathrm{RoI}(\mathcal{O})$ is called a *role type*, if the following two conditions hold:

1. The structure $\mathcal{I}_R = \big(\{a, b\}, \cdot^\mathcal{I}\big)$, such that $r^\mathcal{I} = \{(a, b)\}$ for all $r \in R$ and $r^\mathcal{I} = \varnothing$ otherwise, is a model of $\mathcal{R}$.
2. The knowledge base

$$\mathcal{K}_R = \big(\mathcal{T} \cup \{E_r \sqsubseteq \exists r, E_{r^-} \sqsubseteq \exists r^-\}, \{E_r(a), E_{r^-}(b) \mid r \in R\}\big)$$

is consistent.

In Π_Q, we use the following relation symbols:

- For each concept name $A \in \mathsf{N_C}(\mathcal{O}) \cup \{G\}$, a unary relation symbol A.
- For each role $r \in \mathrm{RoI}(\mathcal{O})$, a binary relation symbol r and two unary relation symbols E_r, E_{r^-}. If $\bot$ is used in role inclusions of Q, we treat $E_\bot, E_{\bot^-}$ simply as $\bot$.

Additionally, for each role type $R \subseteq \mathrm{Rol}(\mathcal{O})$, we will use a dedicated constant d_{R^-}.

Rules. Let $\Pi_{\mathcal{T}}$ be a program encoding the TBox, $\mathcal{T}$, such that, for each concept inclusion, $C_1 \sqcap \cdots \sqcap C_k \sqsubseteq C_{k+1} \sqcup \cdots \sqcup C_{k+m}$, it contains a rule:

$$D_{k+1}(x) \vee \cdots \vee D_{k+m}(x) \leftarrow D_1(x) \wedge \cdots \wedge D_k(x), \tag{4.4}$$

where $D_i = A_i$, if $C_i = A_i$, and $D_i = E_r$, if $C_i = \exists r$.

To encode the RBox, we use a projection of roles, which are binary, to pairs of monadic IDBs. We first introduce some definitions. For $R = \{r_1, \ldots, r_k\}$, let $\mathcal{E}_R(x)$ denote the conjunction

$$E_{r_1}(x) \wedge \cdots \wedge E_{r_k}(x)$$

and $\mathcal{E}_{R^-}(x)$ denote the conjunction

$$E_{r_1^-}(x) \wedge \cdots \wedge E_{r_k^-}(x).$$

Furthermore, let $\mathrm{Tp}(W)$ denote the collection of all role types $R \subseteq \mathrm{Rol}(\mathcal{O})$ such that $W \subseteq R$. For each set of roles $W = \{w_1, \ldots, w_m\}$ (which is not necessarily a role type) consider the following expression:

$$\left(\bigvee_{R \in \mathrm{Tp}(W)} \mathcal{E}_R(x) \wedge \mathcal{E}_{R^-}(y) \right) \leftarrow w_1(x, y) \wedge \cdots \wedge w_m(x, y) \tag{4.5}$$

While the head of this "rule" is in disjunctive normal form, we obtain a set of proper rules, denoted as Π_W, by converting (4.5) to a conjunctive normal form and then adding a separate rule for each disjunctive clause. Furthermore, for each role $w \in \mathrm{Rol}(\mathcal{O})$ consider the following rules:

$$\left(\bigvee_{R \in \mathrm{Tp}(\{w\})} \mathcal{E}_R(x) \wedge \mathcal{E}_{R^-}(d_{R^-}) \right) \leftarrow E_w(x) \tag{4.6}$$

The corresponding rule set, denoted as $\Pi_{\exists w}$, is obtained from the above formulas. Intuitively, it is needed to capture the drama happening in the anonymous part of the model, represented by the d_{R^-} constants.

We can now define the program $\Pi_{\mathcal{R}}$ to be the union of the following rule sets:

- Π_W, for each set of roles $W \subseteq \mathrm{Rol}(\mathcal{O})$ as defined in (4.5);
- programs $\Pi_{\exists r}$, for each role $r \in \mathrm{Rol}(\mathcal{O})$ as defined in (4.6);
- Π_r, which, for each role $r \in \mathrm{Rol}(\mathcal{O})$, is the following rule:

$$E_r(x) \leftarrow r(x, y); \tag{4.7}$$

The goal program, Π_G, contains a single rule:

$$\mathsf{Goal}(x) \leftarrow G(x) \tag{4.8}$$

Recall that we defined $\Pi_Q = \Pi_{\mathcal{T}} \cup \Pi_{\mathcal{R}} \cup \Pi_G$. It can be seen from the construction that $|\Pi_Q| = \mathcal{O}\left(2^{|Q|}\right)$. The following lemma is easy to show by constructing a model of $(\mathcal{O}, D)$ that falsifies $G(a)$ from a model of (Π_Q, D) that does the same, and vice versa.

Lemma 4.3. *For any database D, $(\mathcal{O}, D) \models G(a)$ if and only if $(\Pi_Q, D) \models \mathsf{Goal}(a)$.*

Removing the constants. We note that d_{R^-} constants can be eliminated at a cost of a polynomial growth of the program. Indeed, let Π_Q^c comprise those rules of Π_Q that use constants, while Π_Q^{cf} consists of all constant-free rules. Let also Π_Q^g be the result of grounding *all* rules in Π_Q with constants $\{d_{R^-} \mid R \text{ is a role type}\}$. Since each rule uses at most two variables, the size of Π_Q^g is polynomial in the size of Π_Q. Finally, it is not hard to see that $\Pi_Q^g \cup \Pi_Q^{cf}$ is a constant-free program equivalent to Π_Q.

Deciding FO-rewritability. For monadic disjunctive datalog queries, FO-rewritability is decidable in 2NExpTime [Feier et al., 2019, Feder and Vardi, 1998]. Since our translation from Q to Π_Q incurs an exponential blow-up, this immediately yields the 3NExpTime upper bound stated in the theorem.

As noted above, Feier et al. [2019] develop a related translation that maps queries with ontologies in $\mathcal{ALCI}$ to monadic disjunctive datalog programs in polynomial time, and therefore obtain a 2NExpTime upper bound in that setting. They also establish a matching lower bound via a converse translation from monadic disjunctive datalog programs to $\mathcal{ALCI}$ ontologies. That argument does not carry over to *DL-Lite*^{4}, however, since it relies on qualified existential restrictions $\exists r.C$ which are not allowed in the latter. Establishing lower bounds for the complexity of FO-rewritability in the *DL-Lite*^{4} therefore remains an open technical problem.

4.4 Query answering relative to *TDL-Lite*^{4} ontologies

In this section, we prove the following theorem.

Theorem 4.4. *Query answering relative to TDL-Lite*^{4} *ontologies is in* ExpSpace *and* ExpTime*-hard for combined complexity. With flat TDL-Lite*^{4} *ontologies, it is* PSpace*-complete.*

The proof is by obtaining the same bounds on the consistency checking for *TDL-Lite*^{4} knowledge bases. The lower bounds are inherited from *DL-Lite*^{4}, in the general case, and from *LTL*, in the flat case.

For the upper bounds, fix a *TDL-Lite*^{4} KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$. The idea is to translate $\mathcal{K}$ to the universal fragment of $\mathcal{QTL}$, and then ground it to an *LTL* KB, with at most exponential growth in size, in the general case, and a polynomial growth, in the flat case. Then,

the theorem will follow from PSPACE-completeness of consistency checking for *LTL* KBs [Manna and Pnueli, 1991].

The flat case

For flat *TDL-Lite*$^{\mathcal{A}}$, the translation to $\mathcal{QTL}$ is somewhat similar to $\cdot^\dagger$ of Section 2.4.1. However, we shall avoid the existential quantifier that $\cdot^\dagger$ introduces for existential restrictions. Instead, for every role r, we will use symbols E_r and E_{r^-}, as in the previous section.

Recall that concept and role inclusions in temporal description logics have the form

$$\vartheta_1 \sqcap \ldots \sqcap \vartheta_k \sqsubseteq \vartheta_{k+1} \sqcup \ldots \sqcup \vartheta_{k+m} \tag{4.9}$$

where the ϑ_i are temporalised concepts or roles. In *TDL-Lite*$^{\mathcal{A}}$, the use of temporal operators is restricted in role inclusions, but this does not matter now.

We define a translation $\cdot^\ddagger$ as follows:

$$A^\ddagger = A(x) \qquad (\exists r)^\ddagger = E_r(x) \tag{4.10}$$

$$p^\ddagger = p(x,y) \qquad (p^-)^\ddagger = p(y,x) \tag{4.11}$$

Let $\mathcal{O}^\ddagger$ be a $\mathcal{QTL}$ ontology that for every concept inclusion $\alpha \in \mathcal{O}$ of the form (4.9) contains the formula $\alpha^\ddagger$:

$$\Box\Box^- \, [\forall x \, . \, \mathcal{O}^* \vartheta_1^\ddagger \vee \ldots \vee \mathcal{O}^* \vartheta_k^\ddagger \longrightarrow \mathcal{O}^* \vartheta_{k+1}^\ddagger \wedge \ldots \wedge \mathcal{O}^* \vartheta_{k+m}^\ddagger]$$

and for every role inclusion $\beta \in \mathcal{O}$, the formula $\beta^\ddagger$:

$$\Box\Box^- \, [\forall x \, \forall y \, . \, \mathcal{O}^* \vartheta_1^\ddagger \vee \ldots \vee \mathcal{O}^* \vartheta_k^\ddagger \longrightarrow \mathcal{O}^* \vartheta_{k+1}^\ddagger \wedge \ldots \wedge \mathcal{O}^* \vartheta_{k+m}^\ddagger]$$

Additionally, for every role r, we put in $\mathcal{O}^\ddagger$ the axiom:

$$\Box\Box^- \, [\forall x \forall y \, . \, r^\ddagger(x,y) \to E_r(x) \wedge E_{r^-}(y)] \tag{4.12}$$

When no existential restrictions are present in the right-hand sides of concept inclusions, this translation preserves consistency. Note also that $|\mathcal{K}^\ddagger| = \text{poly}\,(|\mathcal{K}|)$ is only polynomially larger than $\mathcal{K}$.

Lemma 4.5. *For a flat TDL-Lite*^{4} *KB* $\mathcal{K}$*,* $\mathcal{K}$ *is consistent whenever* $\mathcal{K}^\ddagger$ *is consistent.*

Now, $\mathcal{K}^\ddagger$ is in the universal fragment of $\mathcal{QTL}$. This allows us to *ground* it to *LTL* with constants from $\mathrm{obj}(\mathcal{D})$. It is not hard to see that the resulting *LTL* KB is consistent whenever so is the original, and, since the relation symbols in $\mathcal{K}^\ddagger$ are of arity at most 2, its size is polynomial in $|\mathcal{K}^\ddagger|$. This justifies the theorem for the case of flat *TDL-Lite*^{4}.

The general case

For the full *TDL-Lite*^{4}, we shall extend $\mathcal{K}^\ddagger$ with formulas that describe the behaviour of roles in the anonymous parts of models. To this end, we define a KB $\mathcal{K}^\star = (\mathcal{O}^\star, \mathcal{D})$, where $\mathcal{O}^\star$ contains every axiom of $\mathcal{O}^\ddagger$ and, additionally, for every role r in $\mathcal{O}$, $\mathcal{O}^\star$, the following sentence:

$$\Box\Box^- \quad \big[\forall x\,.\,E_r(x) \quad \to \bigvee_{R \in \mathrm{Tp}(\{r\})} \big(\mathcal{E}_R(x) \wedge \exists y\,.\,\mathcal{E}_{R^-}(y)\big)\big] \tag{4.13}$$

This causes an exponential growth in the size of the KB, since, in principle, there may be exponentially many types in $\mathrm{Tp}(\{r\})$. That is, $|\mathcal{K}^\star| = 2^{\mathrm{poly}(|\mathcal{K}|)}$.

Lemma 4.6. *For a TDL-Lite*^{4} *KB* $\mathcal{K}$*,* $\mathcal{K}$ *is consistent whenever* $\mathcal{K}^\star$ *is consistent.*

However, $\mathcal{K}^\star$ is not yet in the universal fragment, as the axiom (4.13) contains an existential quantifier. To get rid of it, we introduce a propositional symbol P_R and a constant d_{R^-} for every role type R. Thus, let $\mathcal{K}^\bullet$ be obtained from $\mathcal{K}^\star$ by substituting all formulas of type (4.13) with:

$$\Box\Box^- \,[\forall x\,.\,E_r(x) \quad \to \bigvee_{R \in \mathrm{Tp}(\{r\})} \mathcal{E}_R(x) \wedge P_R] \tag{4.14}$$

$$P_R \to \mathcal{E}_{R^-}(d_{R^-}) \tag{4.15}$$

Note that the formula (4.15) talks about the moment of time 0. Moreover, it is still the case that $|\mathcal{K}^\bullet| = 2^{\mathrm{poly}(|\mathcal{K}|)}$.

Lemma 4.7. *For a TDL-Lite^4 KB $\mathcal{K}$, $\mathcal{K}^\star$ is consistent whenever $\mathcal{K}^\bullet$ is consistent.*

Finally, $\mathcal{K}^\bullet$ is in the universal fragment of $\mathcal{QTL}$. In this case, we ground with constants $\mathrm{obj}(\mathcal{A}) \cup \{d_{R^-} \mid R \subseteq \mathrm{Rol}(\mathcal{O})$ a role type$\}$. Once again, the relation symbols in $\mathcal{K}^\bullet$ are of arity at most 2, which guarantees at most polynomial increase in size.

4.5 Rewritability of *TDL-Lite*^{4} queries is undecidable

We now establish the key result of this chapter. Namely, checking whether an ontology-mediated query with a *TDL-Lite*^{4} ontology is rewritable into any of the first-order query languages or (linear) datalog is undecidable even for flat ontologies.

More precisely, FO(<)- and FO(<, ≡)-rewritability is undecidable unconditionally, while for other languages undecidability is conditioned on the respective complexity class being strictly smaller than coNP. That is, $\mathcal{L}$-rewritability is undecidable for $\mathcal{L}$ being:

- FO(RPR), if $\mathrm{NC}^1 \neq$ coNP;
- linear *datalog*(S, <), if NL ≠ coNP;
- *datalog*(S, <), if P ≠ coNP.

Theorem 4.8. *Let $\mathcal{L}$ be one of* FO(<)*,* FO(<, ≡)*,* FO(<, MOD)*,* FO(RPR)*, or (linear)* datalog(S, <)*. Then $\mathcal{L}$-rewritability of ontology-mediated queries with TDL-Lite^4 ontologies is undecidable modulo the assumptions above.*

Let $M = (S, 2, s_0, \delta)$ be a 2-counter machine. We construct a query $Q_M = (\mathcal{O}_M, Rel_M, \mathsf{Yellow})$, which is FO(<)-rewritable (and thus rewritable into any of the languages from the theorem statement) if M does not halt. If M halts, answering Q_M becomes coNP-hard, and thus Q_M cannot be rewritable into any language that guarantees lower data complexity. Thus, modulo the respective assumptions, checking rewritability into any of the rewriting languages would allow to solve the halting problem for M.

The coNP-hardness of Q_M is achieved by a reduction from graph non-3-colourability. We encode the computation of M in a database; the query first performs a simple correctness check of the input encoding, and then, whenever the given computation is halting, triggers graph colouring. In case M does not halt, however, no halting computation can be found in any database. In this case, answering Q_M reduces to the initial check, which can be described in FO(<).

Overview of the construction

Recall that a *configuration* of a 2-counter machine M is a triple (s, c_1, c_2) with s the current state and c_1, c_2 the current counter values, and that a *computation* of M is a sequence of configurations

$$(s_0, c_1^0, c_2^0), (s_1, c_1^1, c_2^1), \dots, (s_{n-1}, c_1^{n-1}, c_2^{n-1}) \tag{4.16}$$

such that for each $i < n-1$ we have $\delta(s_i, \mathrm{sgn}(c_1^i), \mathrm{sgn}(c_2^i)) = (s_{i+1}, \varepsilon_1, \varepsilon_2)$ and $c_j^{i+1} = c_j^i + \varepsilon_j$, $0 \leqslant i < n-1$.

To simulate the computations of M we need a representation of its configurations, counters and transitions. The database schema Rel_M contains the following symbols:

- For each state $s_i \in S$ and every $\alpha, \beta \in \{0, 1\}$, representing the signs of the two counters, a role name $s_i^{\alpha\beta}$.
- Role names u_1, u_2, for the two counters.
- A role name e that will stand for edges of the graph.

- A role name r that we will use in the reduction.

Intuitively, a configuration of M is represented by a pair of database objects, and role triples of $s_i^{\alpha\beta}$, u_1 and u_2, arranged according to certain rules described below, indicate the current state and counter values. A sequence of objects connected by such triples will thus represent a computation of M. Our ontology $\mathcal{O}_M$ will, first, validate such a sequence, second, ensure that the machine halts in the end of it by "walking" along, and third, enforce a colouring of a graph attached to the end of that sequence. The colours provided will be 4. The final step of answering Q_M will be to detect the presence (in every model) of one distinguished colour, Yellow, which happens only if the graph is *not* 3-colourable. Thus, we will define $\mathcal{O}_M$ as a union $\mathcal{O}_v \cup \mathcal{O}_w \cup \mathcal{O}_p \cup \mathcal{O}_d$, for *validate*, *walk*, *paint* and *detect*. We proceed to the details.

Computation paths

A configuration (s_i, c_1, c_2) is represented by a pair of objects a, a' and three timestamps ℓ_0, ℓ_1, ℓ_2 such that $\ell_1 = \ell_0 + c_1$ and $\ell_2 = \ell_0 + c_2$. The state of the machine and the signs of the counters are asserted by existence of a connection $s_i^{\alpha\beta}(a, a')$. The actual values of the counters c_1 and c_2 are indicated, respectively, by existence of connections $u_1(a, a', \ell_1)$ and $u_2(a, a', \ell_2)$. The crucial property—that all three roles point at the same object a'—will be ensured later by means of guarded $\Diamond$-role inclusions. When we represent a computation, the next configuration is supposed to appear between a' and some other object a''. The correctness of the transition depends on how incoming and outgoing role triples are arranged on the timeline of a'.

Given a temporal database $\mathcal{D}$ over the schema Rel_M we say that $\mathcal{D}$ has a *computation path* if there is a pair (σ, ℓ_0) with $\ell_0 \in \mathrm{tem}(\mathcal{D})$ and σ being a sequence of objects $\sigma_0, \sigma_1, \dots, \sigma_n \in \mathrm{obj}(\mathcal{D})$ such that the following properties hold for $0 \leqslant j < n$:

- $s_j^{\alpha_j \beta_j}(\sigma_j, \sigma_{j+1}, \ell_0) \in \mathcal{D}$

- there is $\ell_1^j \geqslant \ell_0$ such that $u_1(\sigma_j, \sigma_{j+1}, \ell_1^j) \in \mathcal{D}$
- there is $\ell_2^j \geqslant \ell_0$ such that $u_2(\sigma_j, \sigma_{j+1}, \ell_2^j) \in \mathcal{D}$

We call such a path *almost valid* if the following conditions are satisfied for $0 \leqslant j < n$:

(stt) there is no m and no $(s_k, \alpha', \beta') \neq (s_j, \alpha_j, \beta_j)$ such that

$$\mathcal{D} \models \exists x \,.\, s_k^{\alpha'\beta'}(x, \sigma_j, m)$$

(cnt) there is no $m \neq \ell_i^j$, $i = 1, 2$, such that

$$\mathcal{D} \models \exists x \,.\, u_i(x, \sigma_j, m)$$

(trn) $\delta(s_j, \alpha_j, \beta_j) = (s_{j+1}, \ell_1^{j+1} - \ell_1^j, \ell_2^{j+1} - \ell_2^j)$

and simply *valid* if it additionally complies with:

(sgn) $\alpha_j = \mathrm{sgn}(\ell_1^j)$, $\beta_j = \mathrm{sgn}(\ell_2^j)$

Essentially, **(stt)** and **(cnt)** mean that at a transition from the pair (σ_{j-1}, σ_j) to the pair (σ_j, σ_{j+1}), the state and the counter values *before* the transition are defined uniquely. Then, **(trn)** ties them together with the state and counter values *after* the transition, at least on the syntactic level of role superscripts. The condition **(sgn)** confirms that these superscripts actually correspond to the positioning of role triples.

A valid computation path is *initial* if $s_0^{00}(\sigma_0, \sigma_1, \ell_0) \in \mathcal{D}$, which corresponds to the initial state s_0 and both counters at zero. It is *halting* if $s_{n-1}^{\alpha\beta}(\sigma_{n-1}, \sigma_n, \ell_0)$ is such that $\delta(s_{n-1}, \alpha, \beta)$ is undefined. Finally, it is called *proper* when it is valid, initial, and halting.

Lemma 4.9. *There exists a database $\mathcal{D}$ containing a proper computation path if and only if M halts.*

Proof. By construction, there is a one to one correspondence between computations of the form (4.16) and computation paths. Initial paths correspond to computations where $c_1^0 = c_2^0 = 0$ and halting paths to those where $\delta(s_{n-1}, \mathrm{sgn}(c_1^{n-1}), \mathrm{sgn}(c_2^{n-1}))$ is undefined. □

If M halts, we denote by $\mathcal{D}_M$ a database that contains exactly a proper computation path $(\sigma, 0)$ and nothing else.

Axiomatising valid computation paths

We now define the validation ontology, $\mathcal{O}_v$, which ensures that every path in a given database is valid. To ensure **(stt)**, for all $(i, \alpha, \beta) \neq (j, \gamma, \delta)$, we say:

$$\exists (s_i^{\alpha\beta})^- \sqcap \Diamond \exists (s_j^{\gamma\delta})^- \sqsubseteq \bot \tag{4.17}$$

Then, **(cnt)** is formalised as follows, for $i = 1, 2$:

$$\exists u_i^- \sqcap \Diamond \exists u_i^- \sqsubseteq \bot \tag{4.18}$$

For **(trn)**, we shall say that the change of the state and of the counter values complies with the transition function δ.

We first deal with the states. For each pair $s_i^{\alpha\beta}, s_j^{\gamma\delta}$ such that $\delta(s_i, \alpha, \beta) \neq (s_j, \varepsilon_1, \varepsilon_2)$ for all $\varepsilon_1, \varepsilon_2 \in \{-1, 0, 1\}$, we say:

$$\exists (s_i^{\alpha\beta})^- \sqcap \exists s_j^{\gamma\delta} \sqsubseteq \bot \tag{4.19}$$

We now turn to the counter values. Suppose $\delta(s_j, \alpha_j, \beta_j) = (s_{j+1}, \varepsilon_1, \varepsilon_2)$, with $\varepsilon_1 = 1$. Then, in case we had $u_1(\sigma_j, \sigma_{j+1}, \ell_1^j)$, we want to see the next outgoing u_1, from σ_{j+1}, at time $\ell_1^{j+1} = \ell_1^j + 1$, not earlier, not later. We catch the possible violations to this with the following axioms:

$$\Diamond\Diamond^- \exists (s_i^{\alpha\beta})^- \sqcap \exists u_1^- \sqcap \Diamond^- \exists u_1 \sqsubseteq \bot \tag{4.20}$$

$$\Diamond\Diamond^- \exists (s_i^{\alpha\beta})^- \sqcap \exists u_1^- \sqcap \exists u_1 \sqsubseteq \bot \tag{4.21}$$

$$\Diamond\Diamond^- \exists (s_i^{\alpha\beta})^- \sqcap \exists u_1^- \sqcap \Diamond\Diamond \exists u_1 \sqsubseteq \bot \tag{4.22}$$

The three axioms correspond to the cases when $\ell_1^{j+1} < \ell_1^j$, $\ell_1^{j+1} = \ell_1^j$, and $\ell_1^{j+1} > \ell_1^j + 1$, respectively.

The case when $\varepsilon_1 \in \{0, -1\}$, as well as the case of the second counter, is treated in the same fashion. Let $\mathcal{O}'_v$ comprise the axioms from (4.17) to (4.22).

Lemma 4.10. *If $(\mathcal{O}'_v, \mathcal{D})$ is consistent then every computation path in $\mathcal{D}$ is almost valid.*

Proof. Suppose $\mathcal{D}$ contains a computation path (σ, ℓ_0) that is not almost valid. This means that at least one of **(stt)**, **(cnt)** or **trn** is violated for (σ, ℓ_0). By axioms (4.17)-(4.22), any such violation entails $\bot$. □

To complete $\mathcal{O}_v$, it remains to axiomatise **(sgn)**. We will use a role name v, not in Rel_M, and the following axioms:

$$s_i^{\alpha\beta} \sqcap \Diamond^\alpha u_1 \sqcap \Diamond^\beta u_2 \sqsubseteq v \tag{4.23}$$

for all $s_i \in S$ and $\alpha, \beta \in \{0, 1\}$, where $\Diamond^1 = \Diamond$ and $\Diamond^0$ is an empty sequence. Note that temporalised roles $\Diamond u_i$ are guarded by the roles $s_i^{\alpha\beta}$.

Let $\mathcal{O}_v$ extend $\mathcal{O}'_v$ with the axioms (4.23). Intuitively, $v(\sigma_j, \sigma_{j+1}, \ell_0)$ being inferred over $s_j^{\alpha,\beta}(\sigma_j, \sigma_{j+1}, \ell_0)$ *validates* the configuration. Thus, a valid path becomes a v-path from σ_0 to σ_n at time ℓ_0. Formally, a v-path from a to a' at ℓ in a structure $\mathfrak{I}$ is a sequence

$$v(d_0, d_1, \ell), \dots v(d_{n-1}, d_n, \ell)$$

where $d_j \in \Delta_\mathfrak{I}$, $(d_j, d_{j+1}) \in v^{\mathfrak{I},\ell}$, $a = d_0^\mathfrak{I}$, $a' = d_n^\mathfrak{I}$.

We observe the following nice properties of $\mathcal{O}_v$.

Lemma 4.11. *If $(\mathcal{O}_v, \mathcal{D})$ is consistent, it has a canonical model $\mathfrak{M}_{(\mathcal{O}_v,\mathcal{D})}$, i.e. such that for any $\mathfrak{M} \models (\mathcal{O}_v, \mathcal{D})$ there is a homomorphism from $\mathfrak{M}_{(\mathcal{O}_v,\mathcal{D})}$ to $\mathfrak{M}$. Moreover, one can assume that $\mathfrak{M}$ is flat.*

Proof. We observe that $\mathcal{O}_v$ is a *DL-Lite*$_{horn}$ ontology and has a canonical model $\mathfrak{M}_{(\mathcal{O}_v,\mathcal{D})}$ by the results of [Artale et al., 2013b]. Since $\mathcal{O}_v$ is also flat, discarding the anonymous elements from the domain does not break the property of $\mathfrak{M}_{(\mathcal{O}_v,\mathcal{D})}$ being a model. □

Lemma 4.12. *If $(\mathcal{O}_v, \mathcal{D})$ is consistent, then $\mathcal{D}$ contains a valid computation path (σ, ℓ_0) if and only if in every model of $(\mathcal{O}_v, \mathcal{D})$ there is a v-path from σ_0 to σ_n at ℓ_0.*

Proof. If $\mathcal{D}$ contains a valid computation path (σ, ℓ_0), $(\mathcal{O}_v, \mathcal{D}) \models v(\sigma_j, \sigma_{j+1}, \ell_0)$, $0 \leqslant j < n$, by axiom (4.23). For the other direction, take the canonical model $\mathfrak{M}_{(\mathcal{O}_v,\mathcal{D})}$ of Lemma 4.11 and observe that $\mathfrak{M} \models v(a, a', \ell)$ if and only if $(\mathcal{O}_v, \mathcal{D}) \models v(a, a', \ell)$. It is easy to see that every v-path in $\mathfrak{M}$ goes along a computation path that satisfies **(sgn)**. By Lemma 4.10, this computation path is almost valid, so it is valid. □

Axiomatising proper computation paths

We shall now distinguish proper paths from the rest of valid paths by means of the walking ontology, $\mathcal{O}_w$. The idea is to find an object marked with $\exists s_0^{00}$, walk down a v-path, and check that at the end the machine halts.

We introduce role names v^{00}, v^{01}, v^{11} and concept names $A, \overline{A}, B, \overline{B}$ together with new axioms. The concept A will appear in the beginning of an initial path, and then spread along the role v. Therefore:

$$\exists s_0^{00} \sqsubseteq A \qquad A \sqcap \overline{A} \sqsubseteq \bot \qquad v \sqsubseteq v^{00} \sqcup v^{01} \sqcup v^{11} \tag{4.24}$$

$$\exists v^{\alpha\beta} \sqsubseteq A^{\alpha} \qquad \exists (v^{\alpha\beta})^{-} \sqsubseteq A^{\beta} \tag{4.25}$$

with $A^0 = \overline{A}$ and $A^1 = A$. Intuitively, $\overline{A}$ corresponds to "not A", while $v^{\alpha\beta}$ says "A^{α} at the source of v implies A^{β} at the target of v". The absence of v^{10} among the possible role names reflects the truth

table of the implication. Since A is forced to appear at the beginning of an initial path, all valid paths that are initial are labelled by v^{11} in all models, while non-initial ones may get labelled by v^{00} and v^{01}. Therefore, A is propagated along valid initial paths.

If A holds at the end of a halting path, then it is also an initial one, and we signal this fact by landing the concept B:

$$A \sqcap \exists (s_i^{\alpha\beta})^- \sqsubseteq B \tag{4.26}$$

for every $s_i^{\alpha\beta}$ such that $\delta(s_i, \alpha, \beta)$ is undefined.

Let $\mathcal{O}_w$ comprise axioms from (4.24) to (4.26). We observe the following simple property.

Lemma 4.13. *If $(\mathcal{O}_v, \mathcal{D})$ is consistent, $(\mathcal{O}_v \cup \mathcal{O}_w, \mathcal{D}) \models B(a, \ell_0)$ if and only if $\mathcal{D}$ contains a proper computation path (σ, ℓ_0) with $a = \sigma_n$.*

Proof. Note that if $(\mathcal{O}_v, \mathcal{D})$ is consistent, $(\mathcal{O}_v \cup \mathcal{O}_w, \mathcal{D})$ is also consistent. If $(\mathcal{O}_v \cup \mathcal{O}_w, \mathcal{D}) \models B(a, \ell_0)$, it must be the case that:

1. $(\mathcal{O}_v \cup \mathcal{O}_w, \mathcal{D}) \models A(a, \ell_0)$;
2. $(\mathcal{O}_v \cup \mathcal{O}_w, \mathcal{D}) \models \exists (s_i^{\alpha\beta})^-(a, \ell_0)$, where $\delta(s_i, \alpha, \beta)$ is undefined

Condition 1 is only possible if there is $s_0^{00}(a', a'', \ell_0) \in \mathcal{D}$ and the canonical model $\mathfrak{M}_{(\mathcal{O}_v, \mathcal{D})}$ contains a v-path from some a' to a. By Lemma 4.12, $\mathcal{D}$ contains a valid computation path from a' to a. From $s_0^{00}(a', a'') \in \mathcal{D}$ it follows that this path is initial. From Condition 2 we conclude that it is halting. The other direction is straightforward. □

The ontology $\mathcal{O}_w$ links the computations of M, axiomatised by $\mathcal{O}_v$, to our original plan of reducing graph (non) 3-colourability to answering Q_M.

Proper computation paths and 3-colourability

We denote by $G_{\mathcal{D}}(a, \ell)$ the connected undirected graph the nodes of which are those objects of $\mathcal{D}$ that are reachable from a by connections e or e^- at time ℓ.

Whenever $\mathcal{D}$ contains a proper computation path (σ, ℓ_0), we would like to trigger the colouring of the graph $G_{\mathcal{D}}(\sigma_n, \ell_0)$. The starting point is $B(\sigma_n, \ell_0)$ provided by Lemma 4.13. First, we make B spread over the edges e of the graph, in the same fashion as for A and v before (we do not use e^{01} since we identify e^- and e):

$$B \sqcap \overline{B} \sqsubseteq \bot \qquad e \sqsubseteq e^{11} \sqcup e^{00} \tag{4.27}$$

$$\exists e^{00} \sqsubseteq \overline{B} \qquad \exists \left(e^{00}\right)^- \sqsubseteq \overline{B} \tag{4.28}$$

$$\exists e^{11} \sqsubseteq B \qquad \exists \left(e^{11}\right)^- \sqsubseteq B \tag{4.29}$$

Then we require every B-labelled object to be coloured in one of four colours, $\mathsf{Red}, \mathsf{Green}, \mathsf{Blue}, \mathsf{Yellow}$:

$$B \sqsubseteq \mathsf{Red} \sqcup \mathsf{Green} \sqcup \mathsf{Blue} \sqcup \mathsf{Yellow}. \tag{4.30}$$

The colours should be pairwise disjoint, so we let C_1, C_2 range over the colours in the following axioms:

$$C_1 \sqcap C_2 \sqsubseteq \bot, \text{ for all } C_1 \neq C_2. \tag{4.31}$$

Further, the colouring should be correct, that is, the ends of every edge should have distinct colours. This is obtained introducing role names $e_{(C_1,C_2)}$, for all C_1, C_2, and the axioms:

$$e \sqsubseteq \bigsqcup_{C_1 \neq C_2} e_{(C_1,C_2)} \qquad \exists e_{(C_1,C_2)} \sqsubseteq C_1 \qquad \exists e^-_{(C_1,C_2)} \sqsubseteq C_2 \tag{4.32}$$

We set the painting ontology, $\mathcal{O}_p$, to contain the axioms from (4.27) to (4.32).

Now, $G_{\mathcal{D}}(a, \ell_0)$ is not 3-colourable if and only if *in every model* of $(\mathcal{O}_v \cup \mathcal{O}_w \cup \mathcal{O}_p, \mathcal{D})$ there is a node of that graph coloured in Yellow.

Lemma 4.14. *If $(\mathcal{O}_v, \mathcal{D})$ is consistent and $\mathcal{D}$ contains a proper path (σ, ℓ_0), then $G_{\mathcal{D}}(\sigma_n, \ell_0)$ is not 3-colourable if and only if $\mathfrak{M} \models (\mathcal{O}_v \cup \mathcal{O}_w \cup \mathcal{O}_p, \mathcal{D})$ implies that $a^{\mathfrak{M}} \in \mathsf{Yellow}^{\mathfrak{M}}$ for some node a of $G_{\mathcal{D}}(\sigma_n, \ell_0)$.*

Proof. By Lemma 4.13, $(\mathcal{O}_v \cup \mathcal{O}_w, \mathcal{D}) \models B(\sigma_n, \ell_0)$. Then the axioms (4.27)-(4.32) ensure a correct colouring of objects in $G_{\mathcal{D}}(\sigma_n, \ell_0)$. Therefore, G is 3-colourable if and only if there exists $\mathfrak{M} \models (\mathcal{O}_v \cup \mathcal{O}_w \cup \mathcal{O}_p, \mathcal{D})$ that uses only $\mathsf{Red}, \mathsf{Green}, \mathsf{Blue}$. □

Encoding 3-colourability

Now almost everything is ready for the reduction. Suppose M halts. Take any connected undirected graph $G = (V, E)$ and select some node $v^* \in V$. Recall that there is a database $\mathcal{D}_M$ that contains exactly the proper path $(\sigma, 0)$ for M and nothing else. Let $\mathcal{D}_{G,\ell,a}$ be a database containing exactly the facts $e(a_u, a_v, \ell)$ for $\{u, v\} \in E$, with $a_{v^*} = a$. Then $\mathcal{D}_M \cup \mathcal{D}_{G,0,\sigma_n}$ is clearly consistent with $\mathcal{O}_v$, and by lemma 4.14, G is not 3-colourable whenever every model of

$$(\mathcal{O}_v \cup \mathcal{O}_w \cup \mathcal{O}_p, \mathcal{D}_M \cup \mathcal{D}_{G,\ell_0,\sigma_n})$$

contains a Yellow object of $\mathcal{D}_{G,\ell_0,\sigma_n}$. However, we can only query for the colour of a *concrete* object. To resolve this issue, we take $b \notin \Delta_{\mathcal{D}}$ and let $\mathcal{D}^b_{G,\ell}$ contain exactly the facts $r(a_u, b, \ell)$ for $u \in V$. Then we set $\mathcal{D}_G = \mathcal{D}_M \cup \mathcal{D}_{G,0,\sigma_n} \cup \mathcal{D}^b_{G,0}$. The idea of the axioms that follow is to ensure that G is *not* 3-colourable whenever $(\mathcal{O}_M, \mathcal{D}_G) \models \mathsf{Yellow}(b)$.

We employ $\mathcal{O}_d$, the detecting ontology, that makes Yellow spread along r in the same fashion as it was done in (4.27) for B and e:

$$\mathsf{Yellow} \sqcap \overline{\mathsf{Yellow}} \sqsubseteq \bot \qquad r \sqsubseteq r^{11} \sqcup r^{00} \tag{4.33}$$

$$\exists r^{00} \sqsubseteq \overline{\mathsf{Yellow}} \qquad \exists \left(r^{00}\right)^- \sqsubseteq \overline{\mathsf{Yellow}} \tag{4.34}$$

$$\exists r^{11} \sqsubseteq \mathsf{Yellow} \qquad \exists \left(r^{11}\right)^- \sqsubseteq \mathsf{Yellow} \tag{4.35}$$

We finally set $\mathcal{O}_M = \mathcal{O}_v \cup \mathcal{O}_w \cup \mathcal{O}_p \cup \mathcal{O}_d$ and $Q_M = (\mathcal{O}_M, \mathit{Rel}_M, \mathsf{Yellow})$.

Lemma 4.15. *If M halts, answering $Q_M = (\mathcal{O}_M, \mathit{Rel}_M, \mathsf{Yellow})$ is* coNP*-hard. Otherwise, Q_M is* FO(<)*-rewritable.*

Proof. If M halts, there exists the database $\mathcal{D}_M$. The translation from any connected undirected graph G to $\mathcal{D}_G$ can be performed by a constant-depth circuit. Then, $(\mathcal{O}_M, \mathcal{D}_G) \models \mathsf{Yellow}(b, 0)$ if and only if G is not 3-colourable.

Now suppose that M does not halt. Then no database $\mathcal{D}$ may contain a proper computation path. Therefore, if $(\mathcal{O}_v, \mathcal{D})$ is consistent, by Lemma 4.13 there exists $\mathfrak{M} \models (\mathcal{O}_v, \mathcal{D})$ that does not contain any B-labelled element. It follows that there is $\mathfrak{M} \models (\mathcal{O}_M, \mathcal{D})$ that does not contain any Yellow-labelled element. Therefore, $(\mathcal{O}_M, \mathcal{D}) \models \mathsf{Yellow}(a, \ell) \iff \mathcal{D} \models \bot$. Now suppose $(\mathcal{O}_v, \mathcal{D})$ is not consistent — then $(\mathcal{O}_M, \mathcal{D}) \models \mathsf{Yellow}(a, \ell)$ for any a. Thus, $(\mathcal{O}_M, \mathcal{D}) \models \mathsf{Yellow}(a, \ell)$ is equivalent to the inconsistency of $(\mathcal{O}_v, \mathcal{D})$. By Lemma 4.10, the latter amounts to a violation of **(stt)**, **(cnt)**, or **(trn)**, and can be easily expressed by a FO(<)-query to $\mathcal{D}$. □

Theorem 4.8 follows from the above lemma as explained in the beginning of the section.

4.6 Discussion

We took a first step towards understanding (temporal) *DL-Lite* with guarded non-Horn role inclusions by establishing two results: a decidability procedure for FO-rewritability of ontology-mediated queries in *DL-Lite*^{4} (= *DL-Lite*$_{bool/bool}$) (Theorem 4.2), and an undecidability result for rewritability into various query languages in its temporal version *TDL-Lite*^{4} (Theorem 4.8). An open question is whether rewritability is decidable for *TDL-Lite*^{4} without temporalised roles.

The undecidability proof relies only on the operators $\Diamond/\Diamond^-$ used solely in Horn concept and role inclusions. In fact, by introducing additional concept and role names, one can manage with just $\Diamond$ or just $\Diamond^-$,[3] and a single Horn guarded $\Diamond$-role inclusion. The non-Horn inclusions are purely non-temporal. Hence, the undecidability result

[3] Cf. the proof of Theorem 5.3.

already holds for unions of temporal Horn *DL-Lite* TBoxes with non-temporal $\mathit{DL\text{-}Lite}_{bool/bool}$ ontologies and a single guarded $\Diamond$-role inclusion. Ontology-mediated queries with temporal Horn *DL-Lite* TBoxes are all FO(RPR)-rewritable, but adding certain non-Horn concept inclusions destroys this property [Artale et al., 2022]. The natural question is whether one can algorithmically predict the effect of adding a given non-Horn but non-temporal ontology.

First-order rewritability (also known as *predicate boundedness*) has been studied for datalog queries themselves. Predicate boundedness is undecidable for binary programs [Hillebrand et al., 1995], and even for linear programs with a single binary IDB relation [Vardi, 1988]. Temporalised role inclusions in $\mathit{TDL\text{-}Lite}^4$ can be viewed, from the datalog perspective, as employing binary or even ternary IDBs. However, the undecidability proofs of Hillebrand et al. [1995] and Vardi [1988] rely on *chains*, i.e., rules whose body contains patterns such as $r(x_1, x_2) \land r(x_2, x_3)$, which are inexpressible in logics of the *DL-Lite* family. We return to predicate boundedness in more detail in the next chapter.

Chapter 5

On rewritability of *datalog*$^{\bigcirc\diamond}$ queries

5.1 Overview

We now turn our attention to temporal datalog. The use of variables in this language allows it to capture more complex patterns in the data than is possible with the restricted formulas of description logics. On the other hand, datalog is a Horn language, lacking the disjunction of *TDL-Lite*^{4}, and restricted from inferring *existence* of new objects outside of the database's domain, as opposed to $\mathcal{TEL}^\bigcirc$. This makes *datalog*$^{\bigcirc\diamond}$ queries easier to analyse, although they remain non-trivial.

Example 5.1. Imagine a PhD student working on a paper while hostel hopping. Finishing the paper requires staying at the same hostel for at least two consecutive nights. Bus services between hostels, which vary from one day to the next, and hostel vacancies are given by a temporal database with atoms of the form $\mathsf{busService}(a, b, \ell)$ and $\mathsf{Vacant}(a, \ell)$ (see Fig. 5.1 for an illustration). The following *datalog*$_m^{\bigcirc\diamond}$ query $(\Pi_1, \mathsf{Success})$ finds pairs (a, ℓ) such that having

started hopping at hostel a on day ℓ, the student will eventually submit the paper:

$$\Pi_1\colon \quad \begin{array}{l} \mathsf{Success}(x) \leftarrow \mathsf{Vacant}(x) \land \bigcirc\mathsf{busService}(x,y) \land \bigcirc\mathsf{Success}(y) \\ \mathsf{Success}(x) \leftarrow \mathsf{Vacant}(x) \land \bigcirc\mathsf{Vacant}(x). \end{array}$$

It is readily seen that answering this query is NL-complete for data complexity. If, however, we drop the next-time operator $\bigcirc$ from Π_1, it will become equivalent to $\mathsf{Vacant}(x)$. The next query $(\Pi_2, \mathsf{Promising})$ simply looks for pairs (a, ℓ) with a having vacancies for two consecutive nights some time later than ℓ:

$$\Pi_2\colon \quad \begin{array}{l} \mathsf{Promising}(x) \leftarrow \Diamond\mathsf{VacantFor2Nights}(x), \\ \mathsf{VacantFor2Nights}(x) \leftarrow \mathsf{Vacant}(x) \land \bigcirc\mathsf{Vacant}(x). \end{array}$$

This *datalog*$_m^{\bigcirc\Diamond}$ query has a FO(<)-rewriting:

$$\exists t' \big((t < t') \land \mathsf{Vacant}(x, t') \land \mathsf{Vacant}(x, t' + 1)\big),$$

Therefore, the respective data complexity is just AC^0. ⊣

For non-temporal datalog, FO-rewritability, known in database theory as *predicate boundedness*, has been investigated since the mid 1980s with the aim of optimising and parallelising datalog programs [Kanellakis, 1990, Dantsin et al., 2001]. Thus, predicate boundedness was shown to be undecidable for binary datalog queries [Hillebrand et al., 1995] and 2ExpTime-complete for monadic ones (even with a single recursive rule) [Cosmadakis et al., 1988, Benedikt et al., 2015, Kikot et al., 2021]. We thus distinguish the *monadic fragment* *datalog*$_m^{\bigcirc\Diamond}$ of *datalog*$^{\bigcirc\Diamond}$, to keep the hope for decidability.

Ontology-mediated queries given in propositional linear temporal logic *LTL* are always FO(RPR)-rewritable, but some may be also FO(<, MOD)- or even FO(<, ≡)-rewritable, and it is ExpSpace-complete to decide which is the case [Kurucz et al., 2023], even if we restrict the language to temporal Horn formulas. In fact, these

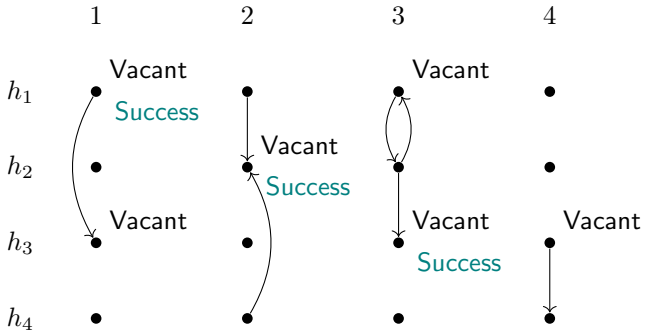


Figure 5.1: Illustrations for the query $(\Pi_1, \mathsf{Success})$. Hostels h_1, h_2, h_3, h_4 are shown on days $1, 2, 3, 4$; bus connections represented by arrows. The predicate Success is given as entailed by the program Π_1.

cases correspond (in the reverse order) to the trichotomy of data complexity for *LTL* OMAQs: answering such a query is either in AC^0, or in $\mathrm{ACC}^0 \setminus \mathrm{AC}^0$, or NC^1-complete [Straubing, 1994]. The data complexity of answering non-temporal monadic datalog queries comes from four complexity classes $\mathrm{AC}^0 \subsetneq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P}$ [Dantsin et al., 2001, Cosmadakis et al., 1988].

Our two-dimensional query language $\mathit{datalog}_m^{\bigcirc\Diamond}$ is a combination of monadic datalog and *LTL*. It is also close in spirit to $\mathit{datalog}_{1\mathrm{S}}$ [Chomicki and Imieliński, 1988, Chomicki, 1990], extending its monadic fragment with the eventuality operator $\Diamond$. The main question we would like to answer in this chapter is whether deciding rewritability of two-dimensional queries into, say, $\mathrm{FO}(<, \equiv)$, can be substantially harder than that of the corresponding one-dimensional datalog and *LTL* queries.

With this in mind, we focus on the linear fragment $\mathit{datalog}_{lm}^{\bigcirc\Diamond}$ of $\mathit{datalog}_m^{\bigcirc\Diamond}$. While the full language inherits from $\mathit{datalog}_{1\mathrm{S}}$ a PSpace-complete query answering problem (for data complexity

[Chomicki and Imieliński, 1988]), we have shown in Theorem 2.8 that linear queries can be answered in NL. That makes $datalog^{\bigcirc\Diamond}_{lm}$ queries not harder than non-temporal linear datalog queries—and inspires the idea to "temporalise" the existing rewritability checking technique developed for the latter.

By $datalog^{\bigcirc}_{lm}$ and $datalog^{\Diamond}_{lm}$ we denote the fragments of $datalog^{\bigcirc\Diamond}_{lm}$ that only admit the $\bigcirc/\bigcirc^-$ and $\Diamond/\Diamond^-$ operators, respectively. All of our queries are assumed to be connected in the sense that the graph induced by the body of each rule is connected. These fragments retain practical interest: as argued by Cosmadakis et al. [1988], non-temporal datalog programs used in practice tend to be linear and connected. For example, SQL:1999 explicitly supports linear recursion [ISO/IEC, 1999], which together with connectedness is a common constraint in the context of querying graph databases and knowledge graphs [Reutter et al., 2021, Urzua and Gutierrez, 2019], where the focus is on path queries [Francis et al., 2018, Harris and Seaborne, 2013].

It is known that deciding whether a linear monadic datalog query can be answered in AC^0 is PSPACE-complete [Cosmadakis et al., 1988, van der Meyden, 2000] (without the monadicity restriction, the problem is undecidable [Hillebrand et al., 1995]). The same problem for the propositional *LTL* fragments of $datalog^{\bigcirc}_{lm}$ and $datalog^{\bigcirc\Diamond}_{lm}$ is also PSPACE-complete [Kurucz et al., 2023].

Our main results in this chapter are as follows:

- It is undecidable whether a given $datalog^{\Diamond}_{lm}$-query is rewritable into $\mathrm{FO}(<)$ or $\mathrm{FO}(<,\equiv)$ (unconditionally), into $\mathrm{FO}(<,\mathrm{MOD})$ (if $\mathrm{ACC}^0 \neq \mathrm{NL}$) and into FO(RPR) (if $\mathrm{NC}^1 \neq \mathrm{NL}$).
- Answering any connected $datalog^{\bigcirc}_{lm}$-query is either in AC^0 or in $\mathrm{ACC}^0 \setminus \mathrm{AC}^0$, or NC^1-complete, or L-hard—and anyway it is in NL.

Dropping the past-time operators $\bigcirc^-$ and $\Diamond^-$ from the languages has no impact on these complexity results. Moreover, we obtain as

a corollary of the second point that it is decidable whether a connected $datalog^{\circ}_{lm}$-query is FO(RPR)-rewritable is PSPACE-complete (if $\mathrm{NC}^1 \neq \mathrm{L}$). Checking whether it is rewritable into FO($<$), FO($<, \equiv$) or FO($<, \mathrm{MOD}$) is in EXPSPACE.

Thus, the temporal operators $\bigcirc/\bigcirc^-$ and $\Diamond/\Diamond^-$ exhibit drastically different types of interaction between the object and temporal dimensions. To illustrate the reason for this phenomenon, consider first the following $datalog^{\circ}_{lm}$-program:

$$G(x) \leftarrow A(x) \wedge \bigcirc R(x, y) \wedge \bigcirc D(y), \tag{5.1}$$

$$D(x) \leftarrow \bigcirc D(x), \tag{5.2}$$

$$D(x) \leftarrow B(x). \tag{5.3}$$

Suppose a temporal database consists of timestamped facts $A(a, 0)$, $R(a, b, 1)$, $B(b, 5)$. We obtain $G(a, 0)$ by first applying rule (5.3) to infer $D(b, 5)$, then rule (5.2) to infer $D(b, 4), D(b, 3), D(b, 2)$, and $D(b, 1)$, and finally rule (5.1) to obtain $G(a, 0)$. Rules (5.2) and (5.3) are applied along the timeline of a single object, b, while the final application passes from one object, b, to another, a. To do so, we check whether a certain condition holds for the joint timeline of a and b, namely, that they are connected by R at time 1. However, if we are limited to the operators $\bigcirc/\bigcirc^-$, the number of steps that we can investigate along such a joint timeline is bounded by the maximum number of nested $\bigcirc/\bigcirc^-$ in the program. Therefore, there is little interaction between the two phases of inference that explore the relational and the temporal domains. In contrast, rules with $\Diamond/\Diamond^-$ can inspect both dimensions simultaneously as, for example, the rule

$$G(x) \leftarrow \Diamond R(x, y) \wedge D(y). \tag{5.4}$$

In this case, inferring $G(a, 0)$ requires checking the existence of an object b satisfying $D(b, 0)$ and $R(a, b, \ell)$ at some *arbitrarily distant* moment ℓ in the future. Given that our programs are monadic, the predicate $\Diamond R(x, y)$ cannot be expressed using operators $\bigcirc/\bigcirc^-$ only.

Our positive results are proved by generalising the automata-theoretic approach of Cosmadakis et al. [1988]. As a by-product, we obtain a method to decompose every connected $datalog^{\bigcirc}_{lm}$-query (Π, G) into a plain datalog part (Π_d, G) and a plain *LTL* part (Π_t, G), which are, however, substantially larger than (Π, G), so that the data complexity of answering (Π, G) equals the maximum of the respective data complexities of (Π_d, G) and (Π_t, G). This reinforces the "weakness of interaction" between the relational and the temporal parts of the query when the latter is limited to operators $\bigcirc/\bigcirc^-$. We also provide some evidence that, in contrast to the non-temporal case, the automata-theoretic approach cannot be generalised to the case of disconnected queries.

The undecidability of the decision problem for $datalog^{\Diamond}_{lm}$-queries is proved by a reduction of the halting problem for Minsky machines with two counters [Minsky, 1967]. The construction is similar to that for Theorem 4.8 of Chapter 4.

The chapter is organised as follows. In Section 5.2, we give some additional technical definitions. In Section 5.3, we show that checking whether a given query with the operator $\Diamond$ or $\Diamond^-$ has data complexity lower than L is undecidable. Section 5.4 considers $datalog^{\bigcirc}_{lm}$-queries by presenting a generalisation of the automata-theoretic approach of Cosmadakis et al. [1988], which is then used in Section 5.5 to provide the decidability results. We conclude with a discussion of future work and our final remarks in Section 5.6.

5.2 Additional preliminaries

Connected queries. For a $\mathcal{QTL}$ formula φ, its *Gaifman graph* is the graph whose nodes are the variables of φ and where two variables are connected by an edge if they occur in the same atom of φ. Then, φ is connected if its Gaifman graph is connected and if it does not contain propositional symbols. A $datalog^{\bigcirc\Diamond}$ query is connected if so are all its rules. We note that by our definition, all IDB relations of a connected monadic query are of arity 1.

CQs and databases. In the non-temporal setting, a body of a CQ (a conjunction of atoms), can be seen as a database (a set of facts). We have a similar correspondence in the temporal setting, for queries without operators $\Diamond$ and $\Diamond^-$. Indeed, observe that $\bigcirc^a(\varphi_1 \wedge \varphi_2) \equiv \bigcirc^a\varphi_1 \wedge \bigcirc^a\varphi_2$ and $\bigcirc\bigcirc^-\varphi \equiv \bigcirc^-\bigcirc\varphi \equiv \varphi$. Hence we can assume that any temporal CQ body is a conjunction of temporalised atoms of the form $\bigcirc^k R(\overline{z})$. Given a temporal CQ $q = \exists u\,.\,\varphi(\overline{x}, \overline{u})$ of this form, let $\mathcal{D}_q$ be a temporal database whose objects a_z correspond to the variables z of φ, and such that $R(a_{z_1}, \dots, a_{z_k}, \ell) \in \mathcal{D}$ whenever $\bigcirc^\ell R(z_1, \dots, z_k)$ is in φ. Then we can, just as in the non-temporal case, characterise the relation $\models$ in terms of homomorphisms:

Lemma 5.2. *For any temporal CQ $q(x_1, \dots, x_k)$ without $\Diamond$ and $\Diamond^-$, $\mathcal{D}, 0 \models q(a_1, \dots, a_k)$ if and only if there is a homomorphism h from $\mathcal{D}_q$ to $\mathcal{D}$ such that $h(a_{x_i}) = a_i$, for $i = 1, \dots, k$.*

Expansions for linear monadic queries

It is well-known that a non-temporal datalog query can be characterised via an infinite set of conjunctive queries called its expansions [Naughton, 1989]. We define expansions for $\mathit{datalog}^{\bigcirc\Diamond}_{lm}$ and use them as the main tool in our (un)decidability proofs.

CQ compositions. Let Π be a $\mathit{datalog}^{\bigcirc\Diamond}_{lm}$ program and $q(x)$ be a unary temporal CQ with a single answer variable and containing a unique IDB atom of Π, say $D(y)$. Let $p(x)$ be another temporal CQ with a single answer variable, and let $p'(y)$ be obtained from $p(x)$ by substituting x by y and all other variables with new ones not appearing in $q(x)$ and $p(x)$. A *composition* of q and p, denoted $q \circ p$, has the form of q with $D(y)$ substituted with $p'(y)$. We note that the variables of q remain present in $q \circ p$ and x is an answer variable of $q \circ p$. If p contains an IDB atom and $g(x)$ is another temporal conjunctive query, the composition can be extended in the same fashion to $(q \circ p) \circ g$, and so on. Note that, up to renaming of variables, $(q \circ p) \circ g$ and $q \circ (p \circ g)$ are the same queries, so we will

omit the brackets and write $q \circ p \circ g$.

Expansions are compositions of rule bodies of a program Π. Let $B_1, B_2, \dots, B_{n-1}$ be such that B_i is the body of the recursive rule:

$$C_i(x) \leftarrow A_i(x, y, u_1, \dots, u_{m_i}) \ \wedge \ \bigcirc^{k_i} C_{i+1}(y), \tag{5.5}$$

and B_n is the body of an initialization rule

$$C_n(x) \leftarrow B(x, v_1, \dots, v_{m_n}). \tag{5.6}$$

The composition $B_1 \circ \cdots \circ B_n$ is called an *expansion* of (Π, C_1), and n is its *length*. The set of all expansions of (Π, C_1) is denoted $\mathit{expand}(\Pi, C_1)$. Moreover, let Γ^r_Π be the set of all recursive rule bodies of Π and Γ^i_Π be the set of all initialization rule bodies. Then each expansion can be regarded (by omitting the symbol $\circ$) as a word in $(\Gamma^r_\Pi)^* \, \Gamma^i_\Pi$, and $\mathit{expand}(\Pi, C_1)$ as a sublanguage of $(\Gamma^r_\Pi)^* \, \Gamma^i_\Pi$. Adopting the language-theoretic notation, we will use small Latin letters w, u, v, etc., to denote expansions. To highlight the fact that each expansion is a temporal CQ with the answer variable x, we sometimes write $w(x) \in \mathit{expand}(\Pi, C_1)$.

It is a direct generalization from the case of plain datalog [Naughton, 1989] that $\mathcal{D}, \Pi \models C_1(a, \ell)$ if and only if there exists $w(x) \in \mathit{expand}(\Pi, C_1)$ such that $\mathcal{D}, \ell \models w(a)$.

5.3 Rewritability is undecidable for *datalog*$^{\diamond}_{lm}$

Our first result about checking rewritability of a given query is negative.

Theorem 5.3. *It is undecidable whether a given* datalog$^{\diamond}_{lm}$*-query can be answered in* AC^0, ACC^0 (*if* ACC$^0 \neq$ NL), *or* NC^1 (*if* NC$^1 \neq$ NL). *It is undecidable whether the query is* NL*-complete.*

Let $M = (S, 2, s_0, \delta)$ be a 2-counter machine. We construct a connected linear query (Π_M, G) using the operator $\Diamond$ only, such that it is FO(<)-rewritable if M halts, but otherwise becomes NL-complete (and thus not even FO(RPR)-rewritable, provided that $\mathrm{NC}^1 \neq \mathrm{NL}$). The case of the operator $\Diamond^-$ instead of $\Diamond$ is symmetric.

Overview of the construction

We repeat again that a *configuration* of a 2-counter machine M has the form (s, c_1, c_2) with s the current state and c_1, c_2 the current counter values, and that a *computation* of M is a sequence of configurations

$$(s_0, c_1^0, c_2^0), (s_1, c_1^1, c_2^1), \ldots, (s_{n-1}, c_1^{n-1}, c_2^{n-1}) \tag{5.7}$$

such that for each $i < n-1$ we have $\delta(s_i, \mathrm{sgn}(c_1^i), \mathrm{sgn}(c_2^i)) = (s_{i+1}, \varepsilon_1, \varepsilon_2)$ and $c_j^{i+1} = c_j^i + \varepsilon_j$, $0 \leqslant i < n-1$.

Compared with the proof of Theorem 4.8, the construction is dual: we want the query to be "easy" when M halts and "difficult" when it does not. By relying on the datalog syntax, we are also able to use a simpler database schema. In fact, we need just three EDB relations T, U_1, U_2, which stand for "transition", "first counter update", and "second counter update", respectively. States of M, instead, will be represented by IDB relations, and therefore we have to keep them monadic. Thus, configurations of M will correspond to objects of a temporal database and relation triples of T, U_1 and U_2 will play the role of transitions. As before, a computation of M will be represented by a sequence of nodes connected by such triples. Our program Π_M will generate an expansion along such a sequence, trying to assign to each configuration an IDB that represents a state of M, which will be possible while the placement of the connecting roles on the temporal line follows the rules of the transition function δ. If the machine halts, there is a maximum number of steps we can make, so the check can be done in AC^0. If M does not halt, however, it has arbitrarily long computations, and the query evaluation becomes NL-complete.

Computation paths and violations

A configuration (s, c_1, c_2) is represented by an object a and three timestamps ℓ_0, ℓ_1, ℓ_2 such that $\ell_1 = \ell_0 + c_1$ and $\ell_2 = \ell_0 + c_2$. The values of the counters c_1 and c_2 are indicated, respectively, by existence of connections $U_1(a, a', \ell_1)$ and $U_2(a, a', \ell_2)$ to some object a' that is supposed to represent the next configuration in the computation. For the transition to happen, we also require $T(a, a', \ell_0)$. A database $\mathcal{D}$ contains a *computation path* corresponding to the computation 5.7, if there is a pair (σ, ℓ_0), with $\sigma = (a_0, \dots, a_n) \in \mathrm{obj}(\mathcal{D})^{n+1}$, such that the following properties hold for $0 \leqslant j < n$:

- $T(\sigma_j, \sigma_{j+1}, \ell_0)$
- $U_1(\sigma_j, \sigma_{j+1}, \ell_0 + c_1^j) \in \mathcal{D}$
- $U_2(\sigma_j, \sigma_{j+1}, \ell_0 + c_2^j) \in \mathcal{D}$

In this section, we will only consider computations, and therefore computation paths, where $c_1^0 = c_2^0 = 0$.

Two types of problems may be encountered on such a path. A *representation violation* occurs when an object has more than one outgoing edge of type T, U_1, or U_2. A *transition violation* of type (s_i, α, β) is detected when there are two consecutive nodes σ_j, σ_{j+1}, $\alpha = \mathrm{sgn}(c_1^j), \beta = \mathrm{sgn}(c_2^j)$, and $\delta(s_i, \alpha, \beta) \neq (s_{j+1}, c_1^{j+1} - c_1^j, c_2^{j+1} - c_2^j)$.

The program Π_M will look for such violations. It will have an IDB relation symbol S_i per each state s_i of M. The initialisation rules allow to infer any state IDB from a representation violation, and the IDB S_i from a transition violation of type (s_i, α, β). The recursive rules then push the state IDB up a computation path following the rules of δ and tracing 'backwards' a computation of M. Once Π_M infers the initial state IDB at a position with zero counter values, we are done. It remains to give the rules explicitly.

The program Π_M

We use a shortcut $\Diamond^*$ to mean a 'reflexive' version of $\Diamond$, i.e. $\Diamond^*\varphi \equiv \Diamond\varphi \vee \varphi$. Clearly, every rule with $\Diamond^*$ can be rewritten to an equivalent set of rules without it. We need the rules:

$$S_i(x) \leftarrow \Diamond^* RV(x), \text{ for all } s_i \in S \tag{5.8}$$
$$RV(x) \leftarrow T(x,y) \wedge \Diamond T(x,z) \tag{5.9}$$
$$RV(x) \leftarrow U_1(x,y) \wedge \Diamond U_1(x,z) \tag{5.10}$$
$$RV(x) \leftarrow U_2(x,y) \wedge \Diamond U_2(x,z) \tag{5.11}$$

to detect representation violations. For transition violations, we first define IDBs NE^{ε}_c, where $c \in 1,2$ stand for the respective counter and $\epsilon \in \{-1,0,1\}$ stands for the change of that counter value in a transition. Each NE^{ε}_c detects situations when a correct transition was not executed, e.g. having $\epsilon = -1$, the timestamps ℓ and ℓ' that are marked by an outgoing U_c in consecutive configurations satisfy $\ell - 1 > \ell'$, $\ell = \ell'$, or $\ell < \ell'$, which they should not. The rules are the following:

$$NE^{-1}_c(y) \leftarrow U_c(y,z) \wedge U_c(x,y) \tag{5.12}$$
$$NE^{-1}_c(y) \leftarrow U_c(y,z) \wedge \Diamond\Diamond U_c(x,y) \tag{5.13}$$
$$NE^{-1}_c(y) \leftarrow U_c(x,y) \wedge \Diamond U_c(y,z) \tag{5.14}$$
$$NE^{1}_c(y) \leftarrow U_c(y,z) \wedge U_c(x,y) \tag{5.15}$$
$$NE^{1}_c(y) \leftarrow U_c(y,z) \wedge \Diamond U_c(x,y) \tag{5.16}$$
$$NE^{1}_c(y) \leftarrow U_c(x,y) \wedge \Diamond\Diamond U_c(y,z) \tag{5.17}$$
$$NE^{0}_c(y) \leftarrow U_c(y,z) \wedge \Diamond U_c(x,y) \tag{5.18}$$
$$NE^{0}_c(y) \leftarrow U_c(x,y) \wedge \Diamond U_c(y,z) \tag{5.19}$$

for $c \in \{1,2\}$. Then, again, any state can be inferred from a violation:

$$S_i(x) \leftarrow \Diamond^* NE^{\varepsilon_1}_1(y) \wedge \Diamond^{\alpha} U_1(x,y) \tag{5.20}$$
$$S_i(x) \leftarrow \Diamond^* NE^{\varepsilon_2}_2(y) \wedge \Diamond^{\beta} U_2(x,y) \tag{5.21}$$

for each transition $\delta(s_i, \alpha, \beta) = (s_j, \varepsilon_1, \varepsilon_2)$, and for all $\varepsilon_1, \varepsilon_2$ when $\delta(s_i, \alpha, \beta)$ is not defined. Finally, we require

$$S_i(x) \leftarrow T(x, y) \wedge \Diamond^{\alpha} U_1(x, y) \wedge \Diamond^{\beta} U_2(x, y) \wedge S_j(y) \tag{5.22}$$

$$G(x) \leftarrow S_0(x) \wedge U_1(x, y) \wedge U_2(x, y) \tag{5.23}$$

This finalises the construction of (Π_M, G).

Any expansion of (Π_M, G) starts with the body of the rule (5.23) and then continues by a sequence of bodies of the rules of the form (5.22) and ends with a detection either of a representation violation, defined by (5.9)—(5.11), or of a transition violation, defined by (5.20)-(5.21). If M halts in m steps, it is enough to consider expansions containing no more than m bodies of (5.22). Each expansion is a temporal CQ and can be rewritten into a FO(<) formula. The disjunction over all expansions with less than m bodies of (5.22) is then a FO(<)-rewriting of (Π_M, G).

If, on the contrary, M does not halt, we can use expansions representing arbitrarily long prefixes of its computation for a reduction from directed reachability problem, rendering answering of (Π_M, G) hard for NL.

Lemma 5.4. *If M halts, (Π_M, G) is* FO(<)*-rewritable. Otherwise, answering it is* NL*-hard.*

5.4 Automata-theoretic tools for connected *datalog*$^{\bigcirc}_{lm}$

From now on, we focus on connected queries that use operators $\bigcirc$ and $\bigcirc^{-}$ only. In this section, we develop a generalisation of the automata-theoretic approach to analysing query expansions proposed in Cosmadakis et al. [1988]. In Section 5.5 we use this approach to study rewritability of this kind of queries.

Recall from Section 5.2 the definitions of composition and expansion of rule bodies, alphabets Γ^r_Π and Γ^i_Π, and the language

$expand(\Pi, G) \subseteq (\Gamma^r_\Pi)^*(\Gamma^i_\Pi)$. We observe that for any sequence of rule bodies $B_1, \ldots, B_{n-1} \in \Gamma^r_\Pi$ and $B_n \in \Gamma^i_\Pi$ the compositions $B_1 \circ \cdots \circ B_{n-1}$ and $B_1 \circ \cdots \circ B_n$ are well-defined. They are words in the languages $(\Gamma^r_\Pi)^*$ and $(\Gamma^r_\Pi)^*(\Gamma^i_\Pi)$, respectively. Note that $B_1 \circ \cdots \circ B_n$ is a temporal CQ over schema $EDB(\Pi)$, while $B_1 \circ \cdots \circ B_{n-1}$ is that over $EDB(\Pi) \cup IDB(\Pi)$, since it contains the IDB atom of B_{n-1}. For $w \in (\Gamma^r_\Pi)^*(\Gamma^i_\Pi)$, $\mathcal{D}_w$ is defined as the (temporal) database corresponding to (temporal) CQ w, while for $w \in (\Gamma^r_\Pi)^*$ we define $\mathcal{D}_w$ as the database corresponding to the CQ obtained from w by omitting the IDB atom. For either w, $\mathcal{D}_w$ is over the schema $EDB(\Pi)$. Having that, we define the language $accept(\Pi, G) \subseteq (\Gamma^r_\Pi)^* \cup (\Gamma^r_\Pi)^*(\Gamma^i_\Pi)$ of all words $w \in (\Gamma^r_\Pi)^* \cup (\Gamma^r_\Pi)^*(\Gamma^i_\Pi)$ such that $(\Pi, \mathcal{D}_w) \models G(a_x, 0)$.

Plain datalog queries are either FO-rewritable (called *bounded*), or L-hard (*unbounded*), and a criterion of unboundedness can be formulated in language-theoretic terms [Cosmadakis et al., 1988]: a connected linear monadic plain datalog query (Π, G) is *unbounded* if and only if for every k there is $w \in expand(\Pi, G)$, $|w| > k$, such that its prefix of length k is *not* in $accept(\Pi, G)$.

Example 5.5. The query (Π, G), where Π is given by the following plain datalog rules, is unbounded.

$$G(x) \leftarrow R(x, y) \wedge S(y, x) \wedge G(y) \tag{5.24}$$

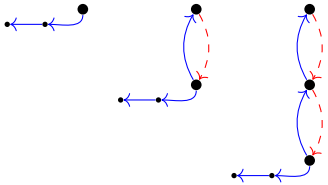
$$G(x) \leftarrow A(x) \tag{5.25}$$

However, if we substitute the rule (5.25) with another initialisation rule

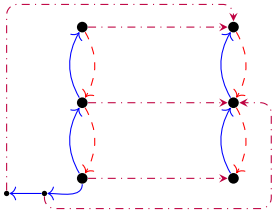
$$G(x) \leftarrow S(x, y) \wedge S(y, z) \tag{5.26}$$

it becomes bounded because for every $w \in expand(\Pi, G)$, $|w| > 2$ its prefix of length 2 is in $accept(\Pi, G)$. To see this, consider the expansions of the query, i.e., $expand(\{(5.24), (5.26)\}, G)$ given in Figure 5.2. ⊣

Cosmadakis et al. [1988] construct finite state automata for languages $expand(\Pi, G)$ and $notaccept(\Pi, G)$, the complement of



(a) Expansions: dashed arrows represent the relation R, solid arrows—the relation S.



(b) A homomorphism (dash-dotted arrows) from an expansion of length 3 to its own prefix of length 2.

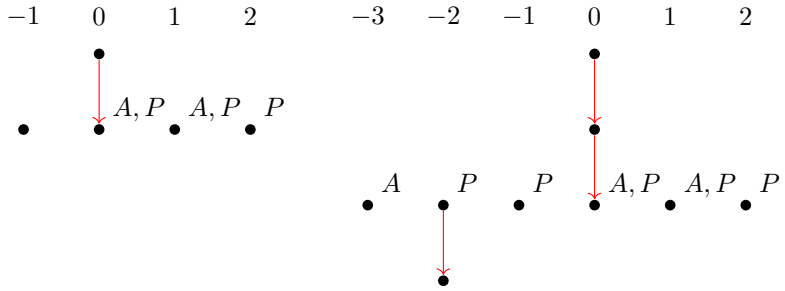
Figure 5.2: Illustrations for the query $\{(5.24), (5.26)\}, G)$ of Example 5.5.

$\mathit{accept}(\Pi, G)$, and use them to check their criterion in polynomial space. Our goal is to generalise this technique to the temporal case. However, we cannot use finite automata to work directly with sequences of rule bodies, since the language $\mathit{accept}(\Pi, G)$, in the presence of time, may be non-regular, as demonstrated by the following example.

Example 5.6. Consider a program

$$G(x) \leftarrow R(x, y) \wedge G(y) \qquad G(x) \leftarrow A(x) \wedge \bigcirc G(x)$$
$$G(x) \leftarrow P(x) \wedge \bigcirc^{-} G(x) \qquad G(x) \leftarrow P(x) \wedge R(y, x).$$

Denote its rule bodies $B_1(x, y) = R(x, y) \wedge G(y)$, $B_2(x) = A(x) \wedge \bigcirc G(x)$, and $B_3(x) = P(x) \wedge \bigcirc^{-} G(x)$, and the composition $w = B_1 B_2 B_2 B_3 B_3 B_3$. The composition w is in $\mathit{accept}(\Pi, G)$, and the corresponding database $\mathcal{D}_w$ is given in Figure 5.3a. In general, a composition of the form $B_1 B_2^n B_3^k$ is in $\mathit{accept}(\Pi, G)$ if and only if $n < k$, which, by a simple application of the pumping lemma, is a non-regular language. ⊣



(a) The temporal database $\mathcal{D}_w$ for $w = B_1B_2B_2B_3B_3B_3$ of Example 5.6.

(b) The temporal database $\mathcal{D}_w$ for $w = B_1^2B_2^2B_3^5B_2B_1$ of Example 5.7.

Figure 5.3: Expansions of $datalog^{\circ}_{lm}$ queries.

A compressed encoding for expansions

We introduce a larger alphabet and define more general versions of the languages $expand(\Pi, G)$ and $accept(\Pi, G)$ to regain their regularity. Recall that every composition w of rule bodies gives rise to a temporal database $\mathcal{D}_w$. Instead of working with w, we use an exponentially larger alphabet $\Omega = 2^{\Gamma^r_\Pi \cup \Gamma^i_\Pi \cup \{\bot, \top\}}$ to describe $\mathcal{D}_w$ as a word and define analogues of $expand(\Pi, G)$ and $accept(\Pi, G)$ over that alphabet. Consider a recursive rule

$$D(x) \leftarrow \bigcirc^{k_1} R_1(\overline{U}_1) \wedge \cdots \wedge \bigcirc^{k_s} R_s(\overline{U}_s) \wedge \bigcirc^{k} E(y), \tag{5.27}$$

where $E(y)$ is the unique IDB atom in the rule body and $k_i, k \in \mathbb{Z}$. We call such a rule *horizontal* if $x = y$ and *vertical* otherwise. In a composition $w \in (\Gamma^r_\Pi)^* \cup (\Gamma^r_\Pi)^*\Gamma^i_\Pi$, a *vertical (horizontal) segment* is a maximal subword that consists of vertical (respectively, horizontal) rule bodies. For every composition w we define the *description* $[w] \in \Omega^*$ of the respective database $\mathcal{D}_w$ as follows. Let $w = u_1v_1u_2v_2 \dots u_nv_n$, where u_i are vertical segments and v_i are horizontal segments, with u_1 and v_n possibly empty. For each

$u_i = B_1 \dots B_n$ we set $[u_i]$ to be the sequence $\{B_1\} \dots \{B_n\}$ of singleton sets, each of which contains a vertical rule body. For each $v_i = B_1 \dots B_n$, we construct $[v_i]$ in n steps, as follows. Recall that B_j, $1 \leqslant j < n$, has the form $A_j(x, \overline{u}) \wedge \bigcirc^{m_j} D(x)$, where $D(x)$ is the unique IDB atom in B_j. Let $\ell_1 = 0$ and $\ell_j = \sum_{i=1}^{j-1} m_i$ for $j \leqslant n$. Intuitively, ℓ_j is the moment of time where the body B_j lands in the composition $B_1 \circ \dots \circ B_n$. Let $\ell'_1, \dots, \ell'_s$ be the ordering of the numbers in the set $\{\ell_j\}_{j=1}^n$ in the increasing order. We set $\alpha_{\ell'_k}$ to be $\{B_j \mid \ell_j = \ell'_k\}$ for $\ell'_k \notin \{\ell_1, \ell_n\}$; $\{B_j \mid \ell_j = \ell_1\} \cup \{\bot\}$ for $\ell'_k = \ell_1 \neq \ell_n$; $\{B_j \mid \ell_j = \ell_n\} \cup \{\top\}$ for $\ell'_k = \ell_n \neq \ell_1$; and $\{B_j \mid \ell_j = \ell_1\} \cup \{\top, \bot\}$ for $\ell'_k = \ell_1 = \ell_n$. Additionally, we set $\alpha_k = \varnothing$ for $k \in [\ell'_1, \ell'_s] \setminus \{\ell'_1, \dots, \ell'_s\}$. Now we take $[v_i] = \alpha$.

Finally, $[w] = [u_1][v_1] \dots [u_n][v_n]$. We use the symbol Λ to refer to letters of $[w]$. We call letters of the form $\{B\}$, where B is a vertical rule body, *vertical letters*, and the rest—*horizontal letters*. Consequently, we can speak of vertical and horizontal segments of $[w]$, meaning maximal segments composed of vertical (respectively, horizontal) letters only.

Intuitively, $[w]$ describes $\mathcal{D}_w$, which can be seen as composed of $\mathcal{D}_{u_1}, \mathcal{D}_{v_1}, \dots, \mathcal{D}_{u_n}, \mathcal{D}_{v_n}$, described by $[u_1], [v_1], \dots, [u_n], [v_n]$, respectively. The symbol $\bot$, representing a vertical line meeting a horizontal one, marks the point in time where $\mathcal{D}_{u_i}$ is connected to $\mathcal{D}_{v_i}$, while the symbol $\top$, analogously, shows where $\mathcal{D}_{v_i}$ is connected to $\mathcal{D}_{u_{i+1}}$.

Example 5.7. Recall the rule bodies of Example 5.6 and consider the composition $w = B_1^2 B_2^2 B_3^5 B_2 B_1$. The corresponding $\mathcal{D}_w$ is depicted in Figure 5.3b. Then $u_1 = B_1^2$, $v_1 = B_2^2 B_3^5 B_2$, $u_2 = B_1$ and v_2 is empty. Thus, $[w]$ is equal to

$$\{B_1\}\{B_1\}\{B_2\}\{\top, B_3\}\{B_3\}\{\bot, B_2, B_3\}\{B_2, B_3\}\{B_3\}\{B_1\} \quad (5.28)$$

⊣

Not every word over Ω correctly describes a temporal database. This motivates the following definition: $\alpha \in \Omega^*$ is *correct* if (i) every

symbol Λ is either a singleton (standing for a vertical rule body) or a set of horizontal rule bodies, with a possible addition of $\bot$, $\top$, and (ii) every horizontal segment of α preceded by a vertical segment contains exactly one $\bot$, and every horizontal segment followed by a vertical one—exactly one $\top$. A correct word $\alpha \in \Omega^*$ describes a temporal database $\mathcal{D}_\alpha$ similarly to how $[w]$ describes $\mathcal{D}_w$. Formally, break α into vertical/horizontal segments $\mu_1 \upsilon_1 \dots \mu_n \upsilon_n$. For a vertical segment $\mu_i = \{B_1\} \dots \{B_n\}$, set $Q_{\mu_i} = B_1 \circ \dots \circ B_n$. For a horizontal segment $\upsilon_i = \Lambda_1 \dots \Lambda_s$, let $\Lambda_{j_\bot}$ be the one containing $\bot$ and $\Lambda_{j_\top}$—containing $\top$. Then Q_{υ_i} is the conjunction of rule bodies from υ_i, where each $B \in \Lambda_j$ is prefixed by $\bigcirc^{j-j_\bot}$, plus, if $i \neq n$, an IDB atom $\bigcirc^{j_\top - j_\bot} D(x)$. Finally, set $\mathcal{D}_\alpha = \mathcal{D}_{Q_\alpha}$ for $Q_\alpha = Q_{\mu_1} \circ Q_{\upsilon_1} \circ \dots \circ Q_{\mu_n} \circ Q_{\upsilon_n}$. This Q_α will also be useful further.

Expansion languages

We are now ready to define languages over Ω that will be useful to study the data complexity of our queries. Let $\mathit{Accept}(\Pi, G)$ be the language of correct words α such that $(\Pi, \mathcal{D}_\alpha) \models G(a_x, 0)$, with $a_x \in \mathrm{obj}(\mathcal{D}_\alpha)$, and $\mathit{NotAccept}(\Pi, G)$ be its complement.

We also need to define the language of expansions over the alphabet Ω that we will use together with the language $\mathit{NotAccept}(\Pi, G)$ to formulate a criterion for L-hardness similar to the one of Cosmadakis et al. [1988]. It would be natural to take the language of expansions as $\{[w] \mid w \in \mathit{expand}(\Pi, G)\}$. However, it is harder to define an automaton recognising such a language than it is for the language of $[w]$s as above where each horizontal letter $[w]_i$ may be extended (as a set) with arbitrary "redundant" rule bodies, and each horizontal segment may be extended, from the left and right, by "redundant" horizontal letters. For example, we will include into the language of expansions the word

$$\{B_1\}\{B_1\}\{B_3\}\{B_2, B_3\}\{\top, B_3\}\{B_2, B_3\}\{\bot, B_2, B_3\}\{B_2, B_3\}\{B_3\}\{B_1\}$$

alongside (5.28). It turns out that the latter language works for our

required criteria as well as the former one.

Formally, if $\alpha, \beta \in \Omega^*$, we write $\alpha \preccurlyeq \beta$ if $\alpha = \mu_1 \upsilon_1 \dots \mu_n \upsilon_n$ and $\beta = \mu_1 \upsilon'_1 \dots \mu_n \upsilon'_n$, where μ_i are vertical segments and υ_i, υ'_i are horizontal segments, and if $\upsilon_i = \Lambda_1 \dots \Lambda_m$, $\upsilon'_i = \Lambda'_1 \dots \Lambda'_s$, then there is $k \geqslant 0$ such that $m + k \leqslant s$ and $\Lambda_j \subseteq \Lambda'_{j+k}$, $1 \leqslant j \leqslant m$. We define $\mathit{Expand}(\Pi, G)$ to be the set of correct words $\alpha \in \Omega^*$ such that $[w] \preccurlyeq \alpha$ for some $w \in \mathit{expand}(\Pi, G)$. It is important for our purposes that these languages are regular. For $\mathit{Expand}(\Pi, G)$, the rules of the program Π may be naturally seen as transition rules of a two-way automaton whose states are the IDBs of Π (plus a final state). The initial state is the one associated with G, and the final state is reached by an application of an initialisation rule.

Lemma 5.8. *For any connected datalog$^{\bigcirc}_{lm}$-query (Π, G), the language $\mathit{Expand}(\Pi, G)$ is regular.*

The case of $\mathit{NotAccept}(\Pi, G)$ is more involved. Generalising from Cosmadakis et al. [1988], for an automaton to recognise if $\alpha \in \mathit{NotAccept}(\Pi, G)$ it suffices to find a *flat* model $\mathfrak{M}$ of $\mathcal{D}_\alpha$, and check that $\mathfrak{M}$ is also a model of Π and that it does not satisfy $G(a_x, 0)$. Moreover, since Π is connected, for $\mathfrak{M}$ to be a model it is enough that every "piece" of $\mathfrak{M}$ of radius $|\Pi|$, in terms of Gaifman graphs, satisfies all the rules. The idea is to precompute the answer for each such piece and encode them in the state-space of the automaton. The problem, specific to the temporal case, is that even flat models are still infinite in the temporal dimension. To resolve this, we observe that there are finitely many EDB atoms in any flat model $\mathfrak{M}$. Since, once again, Π is connected, to check that rules with EDBs are satisfied it is enough to consider only those pieces of $\mathfrak{M}$ that contain an EDB atom. For the rest of the rules, it suffices to check if such a finite piece *can* be extended infinitely in time to give a model of Π. The rules without EDBs can be seen as *LTL* rules, so we can employ satisfiability checking for *LTL* to perform this check.

Lemma 5.9. *For any connected datalog$^{\bigcirc}_{lm}$-query (Π, G), the languages $\mathit{NotAccept}(\Pi, G)$ and $\mathit{Accept}(\Pi, G)$ are regular.*

5.5 Decidability for connected $datalog^{\circ}_{lm}$

We use the automata introduced in the previous section to establish a decidable tetrachotomy of data complexity for $datalog^{\circ}_{lm}$ queries, from which we will derive results on rewritability checking.

Theorem 5.10. (*i*) *Answering a connected* $datalog^{\circ}_{lm}$ *query is either in* AC^0*, or in* $\mathrm{ACC}^0 \setminus \mathrm{AC}^0$*, or* NC^1*-complete, or* L*-hard, for data complexity.* (*ii*) *It is* PSpace*-complete to check whether answering a given query is* L*-hard (if* $\mathrm{NC}^1 \neq \mathrm{L}$*); whether it belongs to* AC^0*,* ACC^0*, or is* NC^1*-complete can be decided in* ExpSpace*.*

We first deal with (*i*). Intuitively, L-hardness is a consequence of the growth of query expansions in the relational domain. If this growth is limited, the query essentially defines a certain temporal property, which can be checked in NC^1.

Vertical boundedness

Given a word $\alpha \in \Omega^*$, we define height(α) as the number of vertical letters in α. Then, we call a query *vertically unbounded* if for every k there is a word $\alpha \in \mathit{Accept}(\Pi, G)$, height($\alpha$) $> k$, such that every prefix of α of height k is in $\mathit{NotAccept}(\Pi, G)$. Otherwise, the query is called *vertically bounded*.

Vertically unbounded queries can be shown to be L-hard by a direct reduction from the undirected reachability problem. Namely, take the deterministic automata for $\mathit{NotAccept}(\Pi, G)$ and $\mathit{Accept}(\Pi, G)$, supplied by Lemma 5.9, and apply the pumping lemma to obtain words $\xi, \upsilon, \zeta, \gamma$ such that height(υ) > 0, $\xi\upsilon^i\zeta \in \mathit{NotAccept}(\Pi, G)$ and $\xi\upsilon^i\zeta\gamma \in \mathit{Accept}(\Pi, G)$, for all $i \geqslant 0$. Then, given a graph $\mathcal{G}$ and two nodes s, t, use copies of $\mathcal{D}_\upsilon$ to simulate the edges of $\mathcal{G}$, and attach $\mathcal{D}_\xi$ to s and $\mathcal{D}_{\zeta\gamma}$ to t. Thus, you will obtain a temporal database $\mathcal{D}_\mathcal{G}$, where $\mathcal{D}_\mathcal{G}, \Pi, 0 \models G(s)$ if and only if there is a path from s to t.

Lemma 5.11. *If a query is vertically unbounded, then it is* L*-hard.*

Observe that $(\Pi, \mathcal{D}) \models G(a, \ell)$ whenever $\mathcal{D}, \ell \models Q_\alpha(a)$ for some $\alpha \in \mathit{Accept}(\Pi, G)$. If $\mathcal{D}, \ell \models Q_\alpha(a)$, then $\mu_1, \upsilon_1, \dots, \upsilon_n, \mu_n$, the vertical and horizontal segments of α, appear in $\mathcal{D}$, starting from a at time ℓ. Expand υ_i to υ_i', the $\preccurlyeq$-maximal horizontal segment fitting in $\mathcal{D}$, and let $\beta = \mu_1 \upsilon_1' \dots \mu_n \upsilon_n'$. Then $\beta \in \mathit{Accept}(\Pi, G)$ and $\mathcal{D}, \ell \models Q_\beta(a)$. Thus, to check $(\Pi, \mathcal{D}) \models G(a, \ell)$, it suffices to find vertical segments of some $\alpha \in \mathit{Accept}(\Pi, G)$, taking $\preccurlyeq$-maximal horizontal segments, and check that $\beta \in \mathit{Accept}(\Pi, G)$. If (Π, G) is vertically bounded, there are finitely many vertical segments of interest. Finding vertical segments, as well as extracting $\preccurlyeq$-maximal horizontal ones, can be done by a constant-depth circuit.

Lemma 5.12. *If (Π, G) is vertically bounded, then the data complexity of answering (Π, G) coincides with that of checking membership in $\mathit{Accept}(\Pi, G)$, modulo reductions computable by constant-depth circuits.*

Recall that $\mathit{Accept}(\Pi, G)$ is a regular language, so checking membership in it is either in AC^0, or $\mathrm{ACC}^0 \setminus \mathrm{AC}^0$, or NC^1-complete [Straubing, 1994]. This settles the part (*i*) of Theorem 5.10: a query is either vertically unbounded and thus L-hard, or vertically bounded and thus belongs to one of the three classes mentioned above.

Deciding data complexity and rewritability

It remains to address the part (*ii*) of Theorem 5.10. By a careful analysis of the proof of Lemma 5.9 one can show that $\mathit{NotAccept}(\Pi, G)$ is recognised by a nondeterministic automaton of size $2^{\mathrm{poly}(|\Pi|)}$. Further, checking whether its language belongs to AC^0, $\mathrm{ACC}^0 \setminus \mathrm{AC}^0$, or is NC^1-complete, can be done via the polynomial-space procedure developed by Kurucz et al. [2023]. Thus, given a vertically bounded query, its data complexity can be established in exponential space.

For the vertical boundedness itself, note that substituting $\mathit{Accept}(\Pi, G)$ with $\mathit{Expand}(\Pi, G)$ in the respective definition pre-

serves all the results proven so far. For $\mathit{Expand}(\Pi, G)$, we can get from Lemma 5.8 an exponential-size one-way automaton. Checking that the query is vertically unbounded can be done in nondeterministic space logarithmic in the size of the automata for $\mathit{Expand}(\Pi, G)$ and $\mathit{NotAccept}(\Pi, G)$, as it boils down to checking reachability in their Cartesian product.

Lemma 5.13. *Checking if a connected* $datalog^{\circ}_{lm}$ *query is vertically bounded is in* PSPACE*.*

For the lower bound, we show that checking boundedness is already PSPACE-hard for connected linear monadic queries in plain datalog. In fact, PSPACE-hardness was proved by Cosmadakis et al. [1988] for *program boundedness* of disconnected programs. A program Π is called bounded if (Π, G) is bounded for every IDB G in Π. We were able to regain the connectedness of Π by focusing on query boundedness (also called *predicate boundedness*) instead. The idea combines that of Cosmadakis et al. [1988] with that of Section 5.3: define an IDB F that slides along a computation, this time of a space-bounded Turing machine, looking for an erroneous transition.

Lemma 5.14. *Deciding boundedness of connected linear monadic queries in plain datalog is* PSPACE*-hard.*

Deciding rewritability. It follows from the above that a vertically unbounded query is not FO(RPR)-rewritable. By contrast, answering a vertically bounded query (Π, G) reduces to membership checking for the language $\mathit{NotAccept}(\Pi, G)$. Recall that reductions computable by constant-depth circuits can be expressed in first-order logic, and that the property $\alpha \in L$ for a regular language L is always definable in FO(RPR) [Immerman, 1999]. Moreover, it is definable in $\mathrm{FO}(<, \equiv)$ and in $\mathrm{FO}(<, \mathrm{MOD})$ precisely when membership problem for L belongs to AC^0 and ACC^0, respectively [Immerman, 1999, Kurucz et al., 2023]. Consequently, the complexity bounds for checking rewritability into these languages follow directly from part (*ii*) of Theorem 5.10.

A decomposition to one-dimensional queries

Lemmas 5.11 and 5.12 bring us to an interesting consideration: to understand the data complexity of a given query one should analyse its behaviour in the relational domain separately from that in the temporal domain. This can be given a precise sense as follows.

Proposition 5.15. *For every connected datalog*$^{\bigcirc}_{lm}$ *query* (Π, G) *there exist a datalog query* (Π_d, G) *and LTL query* (Π_t, G)*, such that:*

1. (Π, G) *is vertically bounded if and only if* (Π_d, G) *is bounded;*
2. *if* (Π, G) *is vertically bounded then the data complexity of answering it coincides with that for* (Π_t, G)*.*

This highlights the 'weak' kind of interaction between the two domains provided by the operators $\bigcirc$ and $\bigcirc^-$. Given the undecidability results of Section 5.3, no such or similar decomposition is possible for queries with $\Diamond$.

Technically, both Π_d and Π_t are obtained by simulating the deterministic automaton $\mathcal{A}_{\Pi,G}$ for the language $\mathit{Accept}(\Pi, G)$ provided by Lemma 5.9. In both programs, the IDBs correspond to the states of $\mathcal{A}_{\Pi,G}$ and EDBs to the letters of Ω. For Π_t these EDBs are unary and all the rules are horizontal, so that the expansions unwind fully in the temporal domain. In the case of Π_d, the EDBs are binary and every vertical transition of $\mathcal{A}_{\Pi,G}$ is a step by a binary relation, while the horizontal transitions are skipped (thus, vertical boundedness becomes just boundedness). In both programs, initialisation rules correspond to $\mathcal{A}_{\Pi,G}$ reaching an accepting state.

A note on disconnected queries

We conclude this section with a consideration on disconnected queries. In Cosmadakis et al. [1988], deciding boundedness is based on the fact that $\mathit{accept}(\Pi, G)$ remains regular even when Π has disconnected rules. This is not the case for $\mathit{datalog}^{\bigcirc}_{lm}$, as can be seen from the following example.

Example 5.16. Consider the program Π of four rules:

$$G(x) \leftarrow A(x) \land \bigcirc D(x) \qquad D(x) \leftarrow \bigcirc D(x)$$
$$D(x) \leftarrow R(x,y) \land \bigcirc^- B(y) \land \bigcirc^- D(y) \qquad D(x) \leftarrow A(y) \land B(x)$$

Let $B_1 = A(x) \land \bigcirc D(x)$, $B_2 = \bigcirc D(x)$, and $B_3 = R(x,y) \land \bigcirc^- B(y) \land \bigcirc^- D(y)$. Then $B_1 B_2^n B_3^m \in \mathit{accept}(\Pi, G)$, and, consequently, $\{\bot, B_1\}\{B_2\}^n\{\top\}\{B_3\}^m \in \mathit{Accept}(\Pi, G)$, whenever $m > n \geqslant 1$. This property is not recognisable by any finite state automaton. For more general classes of automata suitable to recognise $\mathit{accept}(\Pi, G)$ or $\mathit{Accept}(\Pi, G)$ in the disconnected setting, the properties that are to be checked along the lines of Cosmadakis et al. [1988], such as language emptiness or finiteness, become undecidable. Therefore, disconnected queries possibly require a different approach to analysis. ⊣

5.6 Discussion

We have investigated the complexity of the rewritability problem for some natural classes of monadic datalog queries with temporal operators.

For linear connected queries with operators $\bigcirc/\bigcirc^-$, we have generalised the automata-theoretic technique of Cosmadakis et al. [1988], developed originally for non-temporal datalog, to establish an $\mathrm{AC}^0/\mathrm{ACC}^0/\mathrm{NC}^1/\mathrm{L}/\mathrm{NL}$ classification of temporal query answering, and proved that deciding L-hardness of a given query is PSpace-complete, while checking its membership in AC^0 or ACC^0 can be done in ExpSpace. As a minor side product, we have established PSpace-hardness of deciding boundedness of atemporal connected linear monadic datalog queries.

In sharp contrast to the $\bigcirc/\bigcirc^-$ case, it turns out that checking (non-trivial) membership of queries with operators $\Diamond/\Diamond^-$ in the above complexity classes is undecidable.

The results of this chapter lead to a plethora of natural open questions. Some of them are briefly discussed below.

1. What happens if we disallow applications of $\Diamond/\Diamond^-$ to binary EDB predicates in $datalog^{\Diamond}_{lm}$-queries? We conjecture that this restriction makes checking membership in the above complexity classes decidable. In fact, this conjecture follows from a positive answer to the next question.

2. Can our decidability results for $datalog^{\bigcirc}_{lm}$ be lifted to $datalog^{\bigcirc}_{m}$-queries? Dropping the linearity restriction in the atemporal case results in the extra data complexity class, P, and the higher complexity, 2ExpTime-completeness, of deciding boundedness. The upper bound was obtained using tree automata in Cosmadakis et al. [1988], and we believe that this approach can be generalised to connected $datalog^{\bigcirc}_{m}$-queries in a way similar to what we have done above.

3. On the other hand, dropping the connectedness restriction might turn out to be trickier, if at all possible, as shown by Example 5.16. Finding a new automata-theoretic characterisation for disconnected $datalog^{\bigcirc}_{lm}$-queries remains a challenging open problem.

4. A decisive step in understanding the data complexity of answering queries mediated by a description logic ontology and monadic disjunctive datalog queries was made by Bienvenu et al. [2014], Feier et al. [2019], who established a close connection with constraint satisfaction problems (CSPs). In our case, quantified CSPs (see, e.g., Zhuk [2024]) seem to be more appropriate. Connecting the two areas might be beneficial to both of them.

Chapter 6

Conclusions

In this thesis, we studied several ways of extending relational ontology languages—the description logics $\mathcal{EL}$ and *DL-Lite*, and the recursive query language datalog—with temporal operators. Our central question was how these extensions affect the expressive power of the underlying languages and, consequently, the complexity of the main reasoning tasks: answering ontology-mediated queries and checking rewritability into standard database query languages.

We observe that all of our undecidability proofs that require substantial expressive power—such as simulating conjunctive grammars in temporal $\mathcal{EL}$ or counter machines in temporal *DL-Lite* and datalog—depend on applying temporal operators to binary relation symbols, a feature known in the literature as *non-monodicity* [Hodkinson et al., 2001]. In expressive languages, non-monodicity is well known to cause undecidability of consistency checking. Our results show that even over lightweight base languages, where adding very limited temporalised non-monodic relations still preserves decidability of consistency checking, undecidability may nevertheless emerge for more delicate reasoning tasks.

For $\mathcal{EL}$ extended with the operator $\bigcirc$ ("at the next moment"),

we established a correspondence between ontologies and conjunctive grammars. In the context of ontology-mediated queries, this correspondence is important in two respects. From a theoretical perspective, it provided a powerful tool for analysing the expressive power of the logic and the complexity of query answering, allowing us to settle open questions that remained resistant to standard methods from the model theory of temporal description logics. From a practical perspective, the same connection is promising because it relates query answering to the well-developed area of string parsing and processing, with its highly optimised algorithms and implementations, which could potentially be used off the shelf for query answering.

We believe, however, that this correspondence is also of independent interest in the broader setting of temporal ontology languages. It extends the classical connection between linear temporal logic and regular languages and suggests a more refined language-theoretic perspective on temporal logics. This points to an intriguing research direction: what other language-theoretic phenomena can be captured by temporal ontologies? An even bolder question concerns the number-theoretic properties of sets of the form

$$\{n \in \mathbb{Z} \mid \mathcal{O} \models \forall \overline{x} \, . \, R(\overline{x}) \to \bigcirc^n S(\overline{x})\}.$$

Answering such questions may yield deeper insight into the sources of complexity in temporal reasoning.

We also showed that applying the operator $\Diamond$ ("eventually in the future") to binary relation symbols, even in a highly restricted guarded form, suffices to axiomatise computations of 2-counter machines. In temporal *DL-Lite*, this increase in expressive power can be traced to the ability to derive new binary facts via role inclusions. In *monadic* linear datalog, the jump in expressiveness relative to the non-temporal case is even more striking. In particular, the undecidability of first-order rewritability for linear monadic datalog queries with $\Diamond$ indicates that expansions of such queries cannot be analysed independently of the data. This raises a broader question: since

the expressiveness of fragments of datalog can be characterised via their correspondence with classes of constraint satisfaction problems [Feder and Vardi, 1998], is there a natural analogue of this correspondence for datalog extended with $\Diamond$?

By contrast, the operator $\bigcirc$ behaves in a more "relational" way, at least for connected queries: under a suitable encoding, their expansions can be analysed using automata-theoretic techniques, much like in the non-temporal case. However, these results are currently limited to connected queries. In the non-temporal setting, disconnected queries typically have the same complexity as connected ones, although proving this requires more intricate constructions. In the temporal setting, by contrast, our intuition is that disconnected queries require an essentially different approach. It would be rather surprising[1] if the complexity of the general case turned out to coincide with that of connected queries.

Still within the connected setting, it should be possible to extend our results to arbitrary monadic connected queries with $\bigcirc$ (that is, not necessarily linear ones) by temporalising the tree automata of Cosmadakis et al. [1988]. Although disconnected queries present an interesting theoretical challenge, connectedness remains a natural and practically relevant restriction. The methods developed in this thesis may therefore support the search for rewritings that enable more efficient use of datalog over temporal databases.

An important limitation of both the language-theoretic techniques used for temporal $\mathcal{EL}$ and the automata-theoretic tools applied to datalog is that they assume unary encoding of timestamps in databases and of the iteration counts of nested temporal operators in ontologies. From a theoretical point of view, this does not undermine the comparison of ontology languages, since all of them are studied under the same encoding assumptions. For practical query answering, however, algorithms that work directly with bi-

[1] A result is *surprising* if it surprises at least one person. The definition of an *interesting* result is analogous and left to the reader.

nary notation may be more efficient. For temporal $\mathcal{EL}$, we note a development obtained after the original submission of this thesis, extending the results reported here to binary encoding of numbers in both databases and ontologies [Gnatenko and Kontchakov, 2026]. For the future fragment of temporal $\mathcal{EL}$, however, this result essentially confirms that binary encoding offers no more efficient alternative than expanding to unary form, at an exponential cost. By contrast, for the linear fragment, a more efficient method was found. In the case of temporal datalog, our main concern was rewritability checking rather than query answering, so the only numbers involved are those that count the nesting depth of temporal operators. In this setting, unary encoding remains a natural choice, since it directly reflects the length of the formulas themselves. Even so, understanding the effect of binary encoding remains an interesting and worthwhile challenge.

Appendix A

Proofs for Chapter 2

A.1 Proofs for Section 2.5

Proposition 2.7. *For a temporal CQ $q(\overline{x})$, checking whether $\mathcal{D}, \ell \models q(\overline{a})$ is in AC^0 for data complexity.*

Proof. We show that there exists an FO(<)-formula $\psi_q(\overline{x}, t)$ such that for any $\mathcal{D}$, any $\overline{a} \in \mathrm{obj}(\mathcal{D})^{|\overline{x}|}$, and any $\ell \in \mathrm{tem}(\mathcal{D})$, $\mathcal{D}, \ell \models q(\overline{a})$ if and only if $\mathcal{D} \models \psi_q(\overline{a}, \ell)$. We first prove two auxiliary lemmas.

Lemma A.1. *If $\varphi(\overline{z})$ is defined by (2.35) and has N temporal operators, then for any $\mathcal{D}$ with $\mathrm{tem}(\mathcal{D}) = [l, r]$ and any $\overline{a} \in \mathrm{obj}(\mathcal{D})^{|\overline{z}|}$:*

$$\mathcal{D}, r + N + 1 \models \varphi(\overline{a}) \iff \mathcal{D}, \ell \models \varphi(\overline{a}), \text{ for all } \ell > r + N \quad \text{(A.1)}$$

$$\mathcal{D}, l - N - 1 \models \varphi(\overline{a}) \iff \mathcal{D}, \ell \models \varphi(\overline{a}), \text{ for all } \ell < l - N \quad \text{(A.2)}$$

Proof. The proof is by induction. For $N = 0$, φ is a conjunction of atoms. Clearly, $\mathcal{D}, \ell \not\models \varphi(\overline{a})$ for any $\ell \notin [l, r]$. We justify the induction step for (A.1), while the case of (A.2) is analogous. We do another induction on the structure of the formula. The case $\varphi = R(z_1, \ldots, z_m)$ is trivial. In the case $\varphi = \varphi_1 \wedge \varphi_2$ by the induction hypothesis on N everything works. For the temporal operators, we

consider the most interesting case of $\varphi = \Diamond^- \varphi_1$, while the other cases can be proved by a simple application of the induction hypothesis and are left to the reader. Let $\varphi = \Diamond^- \varphi_1$. If $\mathcal{D}, \ell \not\models \Diamond^- \varphi_1(\overline{a})$ for all $\ell > r + N$, we are done. Otherwise, let $\ell_0 > r + N$ be the least timestamp such that $\mathcal{D}, \ell_0 \models \Diamond^- \varphi_1(\overline{a})$. By the choice of ℓ_0, $\mathcal{D}, \ell_0 - 1 \models \varphi_1(\overline{a})$, where $\ell_0 - 1 \geqslant r + N$. By the induction hypothesis, $\mathcal{D}, r + N \models \varphi_1(\overline{a})$, and therefore $\mathcal{D}, \ell \models \Diamond^- \varphi_1(\overline{a})$ for $\ell > r + N$. $\square$

Lemma A.2. *Let $\varphi(\overline{z})$ be defined by* (2.35)*. For any subformula $\varkappa(\overline{z})$ of φ, there exist formulas $\psi_\varkappa(\overline{z}, t)$ and $\psi^i_\varkappa(\overline{z})$ for $i \in [-N - 1, \dots, -1] \cup [1, \dots, N + 1]$, such that for any $\mathcal{D}$ with $\operatorname{tem}(\mathcal{D}) = [l, r]$ and any objects $\overline{a} \in \operatorname{obj}(\mathcal{D})^{|\overline{z}|}$ we have:*

$$\mathcal{D}, \ell \models \varkappa(\overline{a}) \iff \mathcal{D} \models \psi_\varkappa(\overline{a}, \ell), \text{ for } l \leqslant \ell \leqslant r \tag{A.3}$$

$$\mathcal{D}, (r + i) \models \varkappa(\overline{a}) \iff \mathcal{D} \models \psi^i_\varkappa(\overline{a}), \text{ for } 1 \leqslant i \leqslant N + 1 \tag{A.4}$$

$$\mathcal{D}, (l + i) \models \varkappa(\overline{a}) \iff \mathcal{D} \models \psi^i_\varkappa(\overline{a}), \text{ for } -N - 1 \leqslant i \leqslant -1 \tag{A.5}$$

Proof. The proof is, once again, by induction. For the base case, we set $\psi_R(\overline{z}, t) = R(\overline{z}, t)$ and $\psi^i_R(\overline{z}) = \bot$. For an induction step, we define:

$$\psi_{\varkappa_1 \wedge \varkappa_2} = \psi_{\varkappa_1} \wedge \psi_{\varkappa_2} \tag{A.6}$$

$$\psi^i_{\varkappa_1 \wedge \varkappa_2} = \psi^i_{\varkappa_1} \wedge \psi^i_{\varkappa_2}, \text{ for all } i \tag{A.7}$$

$$\psi^{-1}_{\bigcirc \varkappa}(\overline{z}) = \psi_\varkappa(\overline{z}, \min \mathcal{D}) \tag{A.8}$$

$$\psi^{N+1}_{\bigcirc \varkappa}(\overline{z}) = \psi^{N+1}_\varkappa(\overline{z}) \tag{A.9}$$

$$\psi^i_{\bigcirc \varkappa}(\overline{z}) = \psi^{i+1}_\varkappa(\overline{z}), \text{ for all other } i \tag{A.10}$$

$$\psi_{\bigcirc \varkappa}(\overline{z}, t) = \exists t' [(t' = t + 1) \wedge \psi_\varkappa(\overline{z}, t')] \vee ((t = \max \mathcal{D}) \wedge \psi^1_\varkappa(\overline{z})) \tag{A.11}$$

$$\psi^i_{\Diamond \varkappa}(\overline{z}) = \bigvee_{i < j \leqslant N+1} \psi^j_\varkappa(\overline{z}), \text{ for } 0 < i < N + 1 \tag{A.12}$$

$$\psi^i_{\Diamond\varkappa}(\overline{z}) = \bigvee_{i<j\leqslant N+1} \psi^j_{\varkappa}(\overline{z}) \wedge \exists t[(\min \leqslant t \leqslant \max) \wedge \psi_{\varkappa}(\overline{z}, t)], \quad \text{for } -N-1 \leqslant i < 0 \tag{A.13}$$

$$\psi^{N+1}_{\Diamond\varkappa}(\overline{z}) = \psi^{N+1}_{\varkappa}(\overline{z}) \tag{A.14}$$

$$\psi_{\Diamond\varkappa}(\overline{z}, t) = \exists t'[(t < t') \wedge \psi_{\varkappa}(\overline{z}, t')] \vee \bigvee_{1\leqslant i\leqslant N+1} \psi^i_{\varkappa}(\overline{z}) \tag{A.15}$$

And analogously for $\bigcirc^-$ and $\Diamond^-$. The correctness of the induction step follows from the semantics of temporal operators and the statement of Lemma A.1. □

Now, for $q = \exists \overline{u} \,.\, \varphi(\overline{x}, \overline{u})$, we apply Lemma A.2 to $\varphi(\overline{x}, \overline{u})$ and obtain $\psi_\varphi(\overline{x}, \overline{u}, t)$. The required rewriting $\psi_q(\overline{x}, t)$ is then the formula $\exists \overline{u} \,.\, \psi_\varphi(\overline{x}, \overline{u}, t)$. □

A.2 Proofs for Section 2.6

Theorem 2.8. *Answering linear datalog$^{\bigcirc\Diamond}$ queries is* PSPACE*-complete and* NL*-complete, respectively, for combined and data complexity.*

Proof. The lower bound follows from NL-hardness of query answering for non-temporal linear datalog. For a program Π, a rule

$$C(\overline{x}) \leftarrow \varphi(\overline{x}, \overline{y}, \overline{u}) \ \wedge \ \bigcirc^k D(\overline{y}), \tag{A.16}$$

with $k \in \{-1, 0, 1\}$ and $C, D \in \mathit{IDB}(\Pi)$, database instance $\mathcal{D}$, $\overline{c}, \overline{d} \in \mathrm{obj}(\mathcal{D})$ and $\ell \in \mathbb{Z}$, we write $C(\overline{c}) \leftarrow^{\varphi}_{\ell,k} D(\overline{d})$ if $\mathcal{D}, \ell \models \exists \overline{u} \,.\, \varphi(\overline{c}, \overline{d}, \overline{u})$. We write $C(\overline{c}) \leftarrow_{\ell,k} D(\overline{d})$ if there exists a rule (A.16) in Π such that $C(\overline{c}) \leftarrow^{\varphi}_{\ell,k} D(\overline{d})$. Similarly, for an initialization rule

$$C(\overline{x}) \leftarrow \varphi(\overline{x}, \overline{u}), \tag{A.17}$$

we write $C(\overline{c}) \leftarrow_\ell^\varphi$ if $\mathcal{D}, \ell \models \exists \overline{u} \,.\, \varphi(\overline{c}, \overline{u})$,
and we define $C(\overline{c}) \leftarrow_\ell$ analogously. Firstly, we observe the following important *monotonicity* property:

$$\begin{aligned} C(\overline{c}) \leftarrow_{\ell,k} D(\overline{d}) \text{ iff } C(\overline{c}) \leftarrow_{\ell',k} D(\overline{d}) \\ C(\overline{c}) \leftarrow_\ell \text{ iff } C(\overline{c}) \leftarrow_{\ell'} \end{aligned} \tag{A.18}$$

for any $\overline{c}, \overline{d}, C, D, k$ as above and any $\ell, \ell' \in \mathbb{Z}$ such that either $\ell, \ell' > \max \mathcal{D} + K_\Pi$ or $\ell, \ell' < \min \mathcal{D} - K_\Pi$, where K_Π is the maximal number of nested temporal operators in a rule body of Π.

We call

$$C_0(\overline{c}^0) \leftarrow_{\ell_0,k_0} C_1(\overline{c}^1) \leftarrow_{\ell_1,k_1} \cdots \leftarrow_{\ell_{n-1},k_{n-1}} C_n(\overline{c}^n) \tag{A.19}$$

an *entailment sequence from* $C_0(\overline{c}^0)$ *at* ℓ_0 *to* $C_n(\overline{c}^n)$ *at* ℓ_n *in* $\mathcal{D}$ if $C_i \in \mathit{IDB}(\Pi)$, $\overline{c}^i \subseteq \mathrm{obj}(\mathcal{D})$, $\ell_i \in \mathbb{Z}$, $\ell_{i+1} = \ell_i + k_i$. An entailment sequence is called *initialized* if $C_n(\overline{c}^n) \leftarrow_{\ell_n}$. It should be clear that for all $\ell \in \mathbb{Z}$ and $\overline{g} \subseteq \mathrm{obj}(\mathcal{D})$, $\mathcal{D}, \Pi \models G(\overline{g}, \ell)$ if and only if there exists some initialized entailment sequence from $G(\overline{g})$ at ℓ in $\mathcal{D}$. Our aim now is to show:

Lemma A.3. *If there exists an initialized entailment sequence from* $C_0(c_0)$ *at* $\ell_0 \in [\min \mathcal{D}, \max \mathcal{D}]$ *in* $\mathcal{D}$*, then there exists an initialized entailment sequence* $C_0(c_0) \leftarrow_{\ell_0,k_0} C_1(c_1) \leftarrow_{\ell_1,k_1} \cdots \leftarrow_{\ell_{n-1},k_{n-1}} C_n(c_n)$*, with* $\ell_i \in [\min \mathcal{D} - K_\Pi - M_{\Pi,\mathcal{D}}, \max \mathcal{D} + K_\Pi + M_{\Pi,\mathcal{D}}]$*, where* $M_{\Pi,\mathcal{D}} = 2(|\mathit{IDB}(\Pi)|(|\mathrm{obj}(\mathcal{D})|)^m)^2$*, where* m *is the maximal arity among* $\mathit{IDB}(\Pi)$*.*

Proof. To simplify the notation, we will prove the lemma for the case $m = 1$, i.e. for monadic queries. The generalisation for the non-monadic case is straightforward.

Consider a sequence (A.19) in the assumption of the lemma and take the minimal i such that:

- either $\ell_i = \max \mathcal{D} + K_\Pi + 1$ and there exists $j > i$ with $\ell_j = \max \mathcal{D} + K_\Pi + 2(|\mathit{IDB}(\Pi)||\mathrm{obj}(\mathcal{D})|)^2 + 1$ and $\ell_t > \ell_i$ for all $i < t < j$,

- or $\ell_i = \min \mathcal{D} - K_\Pi - 1$ and there exists $j > i$ with $\ell_j = \min \mathcal{D} - K_\Pi - 2(|IDB(\Pi)||\mathrm{obj}(\mathcal{D})|)^2 - 1$ and $\ell_t < \ell_i$ for all $i < t < j$.

If such i does not exist, then the sequence (A.19) is already as required by the lemma and we are done. We show how to construct the required entailment sequence for the first case above; the second is analogous and left to the reader. There are two options:

(a) either for all $j > i$ we have $\ell_i > \max \mathcal{D} + K_\Pi + 1$,

(b) or there is $j > i$ with $\ell_j = \max \mathcal{D} + K_\Pi + 1$.

First, we consider (b). Take the minimal such j. We will show how to adjust (A.19) by replacing the entailment sequence from $C_i(c_i)$ at ℓ_i to $C_j(c_j)$ at ℓ_j by another entailment sequence from $C_i(c_i)$ at ℓ_i to $C_j(c_j)$ at ℓ_j satisfying the condition of the lemma. To this end, it will be convenient to consider *derivation sequences* $C_0(c_0) \leftarrow_{k_0} C_1(c_1) \leftarrow_{k_1} \cdots \leftarrow_{k_{n-1}} C_n(c_n)$ with $k_i \in \{-1, 0, 1\}$. We denote such a sequence by $\boldsymbol{der}$ and let $sh_{\boldsymbol{der}}(0) = 0$ and $sh_{\boldsymbol{der}}(i) = \sum_{j=0}^{i-1} k_j$ for $0 < i \leqslant n$. We denote by $\boldsymbol{der}(\ell_0)$ the entailment sequence $C_0(c_0) \leftarrow_{\ell_0,k_0} C_1(c_1) \leftarrow_{\ell_0+k_0,k_1} \cdots \leftarrow_{\ell_0+k_0+\cdots+k_{n-1},k_n} C_n(c_n)$. Let $\boldsymbol{der}$ be $C_i(c_i) \leftarrow_{k_i} \cdots \leftarrow_{k_{j-1}} C_j(c_j)$ such that $\boldsymbol{der}(\boldsymbol{\ell}_i) = C_i(c_i) \leftarrow_{\ell_i,k_i} \cdots \leftarrow_{\ell_{j-1},k_{j-1}} C_j(c_j)$. It follows from the minimality of j that $\min\{sh_{\boldsymbol{der}}(i)\} = 0$ and $sh_{\boldsymbol{der}}(j) = 0$. It should be clear that if we are able to convert $\boldsymbol{der}$ to

$$\boldsymbol{der}' = C'_0(c'_0) \leftarrow_{k'_0} C'_1(c'_1) \leftarrow_{k'_1} \cdots \leftarrow_{k_{m-1}} C'_m(c'_m) \tag{A.20}$$

such that

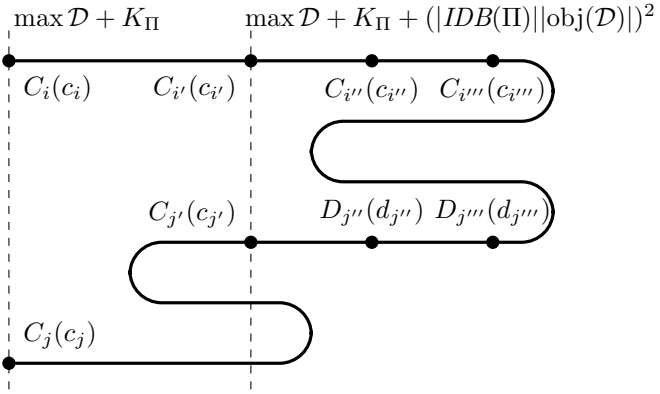
$$\begin{gathered} C'_0(c'_0) = C_i(c_i),\ C'_m(c'_m) = C_j(c_j),\ \min\{sh_{\boldsymbol{der}'}(i)\} = 0, \\ sh_{\boldsymbol{der}'}(m) = 0,\ \text{and}\ \max\{sh_{\boldsymbol{der}'}(i)\} \leqslant 2(|IDB(\Pi)||\mathrm{obj}(\mathcal{D})|)^2, \end{gathered} \tag{A.21}$$

then we are done. Indeed, due to (A.18), if $\boldsymbol{der}$ and $\boldsymbol{der}'$ are two derivation sequences with the same initial $C(c)$, the same final

$D(d)$, and satisfying $\min\{sh(i)\} = 0$, then $\boldsymbol{der}'(\max\mathcal{D} + K_\Pi + 1)$ is an entailment sequence iff $\boldsymbol{der}(\max\mathcal{D} + K_\Pi + 1)$ is an entailment sequence. In $\boldsymbol{der}$, consider the smallest i' such that $sh_{\boldsymbol{der}}(i') = (|IDB(\Pi)||\text{obj}(\mathcal{D})|)^2$ and then the smallest $j' > i'$ such that $sh_{\boldsymbol{der}}(j') = (|IDB(\Pi)||\text{obj}(\mathcal{D})|)^2$ (both i' and j' must exist by our assumptions). If $\boldsymbol{der}' = C_{i'}(c_{i'}) \leftarrow_{k_{i'}} C_{i'+1}(c_{i'+1}) \leftarrow_{k_{i'+1}} \ldots C_{j'}(c_{j'})$ is such that $\max\{sh_{\boldsymbol{der}'}(i)\} \leqslant (|IDB(\Pi)||\text{obj}(\mathcal{D})|)^2$, then we proceed to find the new $i' > j'$ and the new j' such that $sh_{\boldsymbol{der}}(i') = sh_{\boldsymbol{der}}(j') = (|IDB(\Pi)||\text{obj}(\mathcal{D})|)^2$ and treat the new i', j' as the initial ones. Suppose, on the other hand, $\max\{sh_{\boldsymbol{der}'}(i)\} > (|IDB(\Pi)||\text{obj}(\mathcal{D})|)^2$. It follows that there exists a pair $C(c), D(d)$, i'', j'' with $i' < i'' < j'' < j'$, and i''', j''' with $i' < i''' < j''' < j'$, such that

- $C(c) = C_{i''}(c_{i''}) = C_{i'''}(c_{i'''})$,
- $D(d) = C_{j''}(c_{j''}) = C_{j'''}(c_{j'''})$,
- $sh_{\boldsymbol{der}'}(i'') = sh_{\boldsymbol{der}'}(j'') < sh_{\boldsymbol{der}'}(i''') = sh_{\boldsymbol{der}'}(j''') = t$

The $\boldsymbol{der}$ is illustrated below:



It should be clear that $C(c), D(d)$, i'', j'' can be chosen so that $t \leqslant |IDB(\Pi)||\text{obj}(\mathcal{D})|$. We modify $\boldsymbol{der}$ by replacing it with $\boldsymbol{der}' =$

$C_i(c_i) \leftarrow_{k_i} \ldots C_{i''}(c_{i''}) \leftarrow_{k_{i'''}} C_{k_{i'''}+1}(c_{k_{i'''}+1}) \ldots C_{j'''}(c_{j'''}) \leftarrow_{k_{j''}} C_{k_{j''}+1}(c_{k_{j''}+1}) \cdots \leftarrow_{k_{j-1}} C_j(c_j)$. It should be clear that:

- the length of $\boldsymbol{der'}$ is smaller than the length of $\boldsymbol{der}$ (which is equal to $j - i + 1$)
- $\min\{sh_{\boldsymbol{der'}}(i)\} = 0$

It is easy to see that $\boldsymbol{der'}$ will satisfy (b) above again. We repeat the process described in the proof above treating $\boldsymbol{der'}$ as $\boldsymbol{der}$. By repeating this process finitely many times, we are able to achieve $\boldsymbol{der'}$ satisfying (A.21).

Now, consider the first option (a) above. Let $\boldsymbol{der} = C_i(c_i) \leftarrow_{k_i} C_{i+1}(c_{i+1}) \leftarrow_{k_{i+1}} \ldots C_n(c_n)$ such that $\boldsymbol{der}(\boldsymbol{\ell}_i) = C_i(c_i) \leftarrow_{\ell_i,k_i} \cdots \leftarrow_{\ell_{n-1},k_{n-1}} C_n(c_n)$. It follows $\min\{sh_{\boldsymbol{der}}(i)\} = 0$. As in the proof above, we need to convert $\boldsymbol{der}$ to $\boldsymbol{der'}$ (A.20) satisfying

$$C'_0(c'_0) = C_i(c_i),\ C'_m(c'_m) = C_n(c_n),$$
$$\min\{sh_{\boldsymbol{der'}}(i)\} = 0, \text{ and } \max\{sh_{\boldsymbol{der'}}(i)\} \leqslant 2(|IDB(\Pi)||\mathrm{obj}(\mathcal{D})|)^2. \tag{A.22}$$

By our assumptions, (A.22) does not hold for $\boldsymbol{der}$, so consider the smallest i' in $\boldsymbol{der}$ such that $sh_{\boldsymbol{der}}(i') = (|IDB(\Pi)||\mathrm{obj}(\mathcal{D})|)^2$. First, we assume that there exists no $j' > i'$ such that $sh_{\boldsymbol{der}}(i') = sh_{\boldsymbol{der}}(j')$. In this case, from $\boldsymbol{der''} = C_{i'}(c_{i'}) \leftarrow_{k_{i'}} \cdots \leftarrow_{k_n} C_n(c_n)$, we remove all the loops of the form $C(c) \leftarrow \cdots \leftarrow C(c)$. It should be clear that $\max\{sh_{\boldsymbol{der''}}(i)\} \leqslant |IDB(\Pi)||\mathrm{obj}(\mathcal{D})|$ and $|\min\{sh_{\boldsymbol{der''}}(i)\}| \leqslant |IDB(\Pi)||\mathrm{obj}(\mathcal{D})|$. Therefore, $\boldsymbol{der'}$ obtained by removing the loops from $\boldsymbol{der}$ after i' will satisfy (A.22) and we are done. If, on the other hand, there exists $j' > i'$ such that $sh_{\boldsymbol{der}}(i') = sh_{\boldsymbol{der}}(j')$, we adjust the sequence $C_{i'}(c_{i'}) \leftarrow_{\ell_{i'},k_{i'}} \cdots \leftarrow_{\ell_{j'-1},k_{j'-1}} C_{j'}(c_{j'})$ in $\boldsymbol{der}$ as explained in the proof for (b). The resulting $\boldsymbol{der'}$ will be shorter than $\boldsymbol{der}$ and, unless it satisfies (A.22), (a) will apply to it. It should be clear that after finitely many repetitions of the process described in the proof above (for (a)), we obtain $\boldsymbol{der'}$ satisfying (A.22). □

We now return to the proof of the theorem. The above lemma proposes the following way to answer (Π, G) over a database $\mathcal{D}$. Rewrite (Π, G) into a *datalog*$(\mathrm{S}, <)$ query (Π', G) and evaluate the latter over a temporal database that is equal to $\mathcal{D}$ except that $\mathrm{tem}(\mathcal{D}')$ spans $[\min \mathcal{D} - K_\Pi - M_{\Pi,\mathcal{D}}, \max \mathcal{D} + K_\Pi + M_{\Pi,\mathcal{D}}]$. The following bounds hold:

$$|\Pi'| = \mathrm{poly}\,(|\Pi|) \qquad |\mathcal{D}'| = \mathrm{poly}\left(|\mathcal{D}|^{|\Pi|}\right) \tag{A.23}$$

The NL upper bound for data complexity follows from that for *datalog*$(\mathrm{S}, <)$. For combined complexity, the PSpace upper bound follows from the fact that $|\mathcal{D}'|$ is exponential in $|\Pi|$, but only polynomial in $|\mathcal{D}|$, and the proof of Gottlob and Papadimitriou [2003, Theorem 4.4]. □

Appendix B

Proofs for Chapter 3

B.1 Two definitions of ultimate periodicity

Gutiérrez-Basulto et al. [2016a] define ultimate periodicity using quasimodels. We show here that every $\mathcal{TEL}^{\circ}$-TBox $\mathcal{T}$ ultimately periodic under this definition is also ultimately periodic as defined in Section 3.2, and vice versa. First, we recall the notion of quasimodels. The following few paragraphs quote almost verbatim Gutiérrez-Basulto et al. [2016a] (the difference is that they use concept inclusions of the form $\bigcirc^k A \sqsubseteq B$, $k \in \{-1, 0, 1\}$, while we use $A \sqsubseteq \bigcirc^k B$, $k \in \mathbb{Z}$; it is straightforward that these forms are equivalent in terms of expressive power).

Fix a KB $(\mathcal{T}, \mathcal{D})$ with a $\mathcal{TEL}^{\circ}$-TBox $\mathcal{T}$, and let $\mathsf{N_C}(\mathcal{T}, \mathcal{D})$ be the set of concept names used in $(\mathcal{T}, \mathcal{D})$. A map $\pi \colon \mathbb{Z} \to 2^{\mathsf{N_C}(\mathcal{T}, \mathcal{D})}$ is a trace for $\mathcal{T}$ if it satisfies the following:

(t1) if $A \sqcap A' \sqsubseteq B \in \mathcal{T}$ and $A, A' \in \pi(n)$, then $B \in \pi(n)$;

(t2) if $A \sqcap \bigcirc^k B \in \mathcal{T}$ and $A \in \pi(n)$, then $B \in \pi(n + k)$.

Let π be a trace for $\mathcal{T}$. For a rigid role name $r \in \mathsf{N}_\mathsf{R}^\mathsf{rig}$ the r-projection of π is a map $\mathrm{proj}_r(\pi)\colon \mathbb{Z} \to 2^{\mathsf{N}_\mathsf{C}(\mathcal{T},\mathcal{D})}$ that sends each $i \in \mathbb{Z}$ to $\{A \mid \exists r.B \sqsubseteq A \in \mathcal{T}, B \in \pi(i)\}$. For a local role name $r \in \mathsf{N}_\mathsf{R}^\mathsf{loc}$, $\mathrm{proj}_r(\pi)$ is defined in the same way on 0 but is $\emptyset$ for all other $i \in \mathbb{Z}$. Given a map $\rho\colon \mathbb{Z} \to 2^{\mathsf{N}_\mathsf{C}(\mathcal{T},\mathcal{D})}$ and $n \in \mathbb{Z}$, we say that π contains the n-shift of ρ and write $\rho \subseteq^n \pi$ if $\rho(i-n) \subseteq \pi(i)$, for all $i \in \mathbb{Z}$.

We now define quasimodels. Let $D = \mathrm{obj}(\mathcal{D}) \cup \mathsf{N}_\mathsf{C}(\mathcal{T},\mathcal{D})$. A *quasimodel* $\mathfrak{Q}$ for $(\mathcal{T},\mathcal{D})$ is a set $\{\pi_d \mid d \in D\}$ of traces for $\mathcal{T}$ such that

(q1) $A \in \pi_a(n)$, for all $A(a,n) \in \mathcal{D}$;

(q2) $B \in \pi_B(0)$, for all $B \in \mathsf{N}_\mathsf{C}(\mathcal{T},\mathcal{D})$;

(q3) $A \in \pi_a(n)$, for all $B \in \pi_b(n)$, $r(a,b,n) \in \mathcal{D}$, and $\exists r.B \sqsubseteq A \in \mathcal{T}$;

(q3') $\mathrm{proj}_r(\pi_b) \subseteq^0 \pi_a$, for all $r(a,b,n) \in \mathcal{D}$, $r \in \mathsf{N}_\mathsf{R}^\mathsf{rig}$;

(q4) if $A \in \pi_d(n)$, then $\mathrm{proj}_r(\pi_B) \subseteq^n \pi_d$, for all $d \in D$, $n \in \mathbb{Z}$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$.

Compared to the original [Gutiérrez-Basulto et al., 2016a], we split the point **(q3)** into two versions, treating local and rigid roles, fixing a small glitch in the original definition after a discussion with the authors. This does not affect further results.

Intuitively, π_a represents $a \in \mathrm{obj}(\mathcal{D})$; and π_B represents all elements that witness B for $A \sqsubseteq \exists r.B \in \mathcal{T}$. For the purposes of query answering, canonical quasimodels are defined. The *canonical quasimodel* is the limit of the following saturation procedure. Start with initially empty maps π_d, for $d \in D$, and apply **(t1)**-**(t2)**, **(q1)**-**(q4)** as rules: e.g., **(q3')** says "if $r(a,b,n) \in \mathcal{D}$, for $r \in \mathsf{N}_\mathsf{R}^\mathsf{rig}$, and $A \in \mathrm{proj}_r(\pi_b)(i)$, then add A to $\pi_a(i)$".

Theorem B.1 (Gutiérrez-Basulto et al. [2016a])**.** *Let $\mathfrak{Q} = \{\pi_d \mid d \in D\}$ be the canonical quasimodel of $(\mathcal{T},\mathcal{D})$ where $\mathcal{T}$ is a $\mathcal{TEL}^\circ$-TBox. Then, for any $A \in \mathsf{N}_\mathsf{C}$, $(\mathcal{T},\mathcal{D}) \models A(a,n)$ iff $A \in \pi_a(n)$, for $a \in \mathrm{obj}(\mathcal{D}), n \in \mathbb{Z}$.*

Finally, let $\mathcal{T}$ be a $\mathcal{TEL}^{\circ}$-TBox and $\mathfrak{Q}$ the canonical quasimodel for $(\mathcal{T}, \emptyset)$. We say that $\mathcal{T}$ is *ultimately periodic w.r.t. quasimodels*, if for each $A \in \mathsf{N_C}(\mathcal{T})$ there are positive integers $m_P, p_P, m_F, p_F \in \mathbb{N}$, such that the following conditions hold for π_d, with $d = A$.

$$\pi_d(n - p_P) = \pi_d(n) \qquad \text{for all } n \leqslant -m_P \tag{B.1}$$

$$\pi_d(n + p_F) = \pi_d(n) \qquad \text{for all } n \geqslant m_F \tag{B.2}$$

Note that by definition, the traces π_A, $A \in \mathsf{N_C}(\mathcal{T})$, are the same in all canonical quasimodels of $(\mathcal{T}, \mathcal{D})$, for any $\mathcal{D}$. We observe the following property.

Lemma B.2. *For any $\mathcal{TEL}^{\circ}$-TBox $\mathcal{T}$, $A \in \mathsf{N_C}(\mathcal{T})$, and $n \in \mathbb{Z}$, $\pi_A(n) = \{B \in \mathsf{N_C}(\mathcal{T}) \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$.*

Proof. Suppose $\mathfrak{Q} = \{\pi_d \mid d \in \mathsf{N_C}(\mathcal{T}) \cup \{a\}\}$ is the canonical quasimodel of $(\mathcal{T}, \{A(a, 0)\})$, and observe further that in this case $\pi_A = \pi_a$. The lemma follows by Theorem B.1 and Proposition 3.3. □

Now, recall from Section 3.2, that we call a $\mathcal{TEL}^{\circ}$-TBox $\mathcal{T}$ *ultimately periodic (w.r.t. concept inclusions)*, if for every pair $A, B \in \mathsf{N_C}(\mathcal{T})$ the set $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$ is semilinear.

Our goal in this section is to prove that the two definitions are equivalent (Proposition B.5 below). First, we recall a useful fact from arithmetic (see Niven et al. [1991] for details).

Lemma B.3 (Linear Diophantine equations)**.** *An equation $px+qy = c$, where $p, q, c \in \mathbb{Z}$, has a solution (where x and y are integers) if and only if c is a multiple of the greatest common divisor of p and q. Moreover, if (x, y) is a solution, then the other solutions have the form $(x + tp', y - tq')$, $t \in \mathbb{Z}$, and p' and q' are the quotients of p and q (respectively) by the greatest common divisor of p and q.*

In this section, a *simple* set is a set of the form $\{b + kp \mid k \in \mathbb{N}\}$, where $b, p \in \mathbb{Z}$. One can show that every semilinear set $S \subseteq \mathbb{Z}$ is representable as a union of simple sets. For completeness of the presentation, we give a direct proof here.

Lemma B.4. *The following statements hold.*

(i) Every set of the form $\{kp + mq \mid k, m \in \mathbb{N}\}$*, where* $p, q \in \mathbb{Z}$*, is a finite union of simple sets.*

(ii) Every semilinear $S \subseteq \mathbb{Z}$ *is a finite union of simple sets.*

Proof. (i) Let $\mathcal{L} = \{kp + mq \mid k, m \in \mathbb{N}\}$. If $p = 0$ or $q = 0$, $\mathcal{L}$ is itself a simple set. So suppose $p \neq 0$ and $q \neq 0$. We consider three cases:

- If $p, q > 0$, then $\mathcal{L} \subseteq \mathbb{N}$, and by Theorem 2.4 $\mathcal{L} = p(L)$ for some regular language $L \subseteq \{c\}^*$. Let $M = (Q, \{c\}, q_0, \delta, F)$ be the minimal deterministic automaton recognising L, with the set of states Q and final states $F \subseteq Q$, initial state $q_0 \in Q$ and $\delta \colon Q \times \{c\} \to Q$. It is easy to see that for the sequence $q_0, q_1, q_2, \ldots$, where $q_{i+1} = \delta(q_i, c)$, there exist unique $m, p \in \mathbb{N}$ such that $q_j = q_{j+p}$ for all $j \geqslant m$. Then

$$\mathcal{L} = \{j < m \mid q_j \in F\} \quad \cup \bigcup_{\substack{m \leqslant j < m+p \\ q_j \in F}} \{j + kp \mid k \in \mathbb{N}\}$$

- If $p, q < 0$, we take $\mathcal{L}' = \{-1 \cdot n \mid n \in \mathcal{L}\}$, and obtain the representation as in the previous case, then multiply everything by -1.

- If p and q are of different signs. Let $d \geqslant 1$ be their greatest common divisor, and $p' = p/d$, and $q' = q/d$. We have $\mathcal{L} = f(\mathcal{L}')$, where $f(x) = dx$, $x \in \mathbb{Z}$, and $\mathcal{L}' = \{kp' - mq' \mid k, m \in \mathbb{N}\}$. Observe that if a set is simple, its image under f is also simple — hence it is enough to represent $\mathcal{L}'$ as a union of simple sets. By Lemma B.3, there are $k, m \in \mathbb{Z}$ such that $kp' + mq' = 1$. Since p and q are of different signs, we can safely assume that $k, m \in \mathbb{N}$ (otherwise, we find $t \in \mathbb{Z}$ such that $k + tp', m - tq' > 0$, and take that solution). It follows

that $\mathbb{N} \subseteq \mathcal{L}'$. Similarly, we find a positive integer solution for $kp' + mq' = -1$, and conclude that $\{-n \mid n \in \mathbb{N}\} \subseteq \mathcal{L}'$. Thus $\mathcal{L}' = \mathbb{Z} = \{k \cdot 1 \mid k \in \mathbb{N}\} \cup \{k \cdot (-1) \mid k \in \mathbb{N}\}$.

(ii) Since a semilinear set is a finite union of linear sets, it is enough to prove that every linear set $\mathcal{L}$ is a finite union of simple sets. Suppose $\mathcal{L} = \{b + k_1p_1 + \cdots + k_lp_l \mid k_1, \ldots, k_l \in \mathbb{N}\}$. We do induction on l. The cases $l = 0$ or $l = 1$ are trivial. Now suppose that any linear set representable using $l - 1$ periods is a union of simple sets.

Then we have:

$$S = \{b + k_1p_1 + \cdots + k_lp_l \mid k_i \in \mathbb{N}\} =$$
$$\{b + k_1p_1 + \cdots + k_{l-1}p_{l-1} \mid k_i \in \mathbb{N}\} + \{k_lp_l \mid k_l \in \mathbb{N}\} =$$
$$\left(\bigcup_{i=1}^{m} \mathcal{L}_i\right) + \{k_lp_l \mid k_l \in \mathbb{N}\} = \bigcup_{i=1}^{m} (\mathcal{L}_i + \{k_lp_l \mid k_l \in \mathbb{N}\})$$

where $S_1 + S_2 = \{x + y \mid x \in S_1, y \in S_2\}$, and $\mathcal{L}_i$ are simple.

If $\mathcal{L}_i = \{b\}$, then $\mathcal{L}_i + \{k_lp_l \mid k_l \in \mathbb{N}\} = \{b + k_lp_l \mid k \in \mathbb{N}\}$, and we are done. If $\mathcal{L}_i = \{b + kp \mid k \in N\}$, then $\mathcal{L}_i + \{k_lp_l \mid k_l \in \mathbb{N}\} = \{b\} + \{kp + k_lp_l \mid k, k_l \in \mathbb{N}\}$. By point *(i)*, the latter is a finite union of simple sets $\bigcup_{j=1}^{r}\{b_r + k_rp_r \mid k_r \in \mathbb{N}\}$ so $\mathcal{L}_i + \{k_lp_l \mid k_l \in \mathbb{N}\} = \{b\} + \bigcup_{j=1}^{r}\{b_r + k_rp_r \mid k_r \in \mathbb{N}\} = \bigcup_{j=1}^{r}\{b + b_r + k_rp_r \mid k_r \in \mathbb{N}\}$. Hence, in both cases, S is a finite union of simple sets. □

Proposition B.5. *A $\mathcal{TEL}^{\circ}$-TBox is ultimately periodic w.r.t. quasimodels iff it is ultimately periodic w.r.t. concept inclusions.*

Proof. ($\Rightarrow$) Let $\mathcal{T}$ be ultimately periodic w.r.t. quasimodels and fix $A, B \in \mathsf{N_C}(\mathcal{T})$. Let m_P, p_P, m_F, p_F be such that (B.1)-(B.2) hold for π_A. By Lemma B.2, $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\} = \{n \in \mathbb{Z} \mid B \in \pi_A(n)\}$. We represent the latter set as a union of the following linear sets:

$$\mathcal{L}_n = \{n\} \qquad \text{for} \quad \begin{array}{l} n \in (-m_P, m_F) \\ B \in \pi_a(n) \end{array}$$

$$\mathcal{L}'_n = \{n - kp_P \mid k \in \mathbb{N}\} \qquad \text{for} \quad \begin{array}{l} n \in (-m_P - p_P, -m_P] \\ B \in \pi_a(n) \end{array}$$

$$\mathcal{L}''_n = \{n + kp_F \mid k \in \mathbb{N}\} \qquad \text{for} \quad \begin{array}{l} n \in [m_F, m_F + p_F) \\ B \in \pi_a(n) \end{array}$$

Thus, $\{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$ is semilinear.
($\Leftarrow$) Assume that $\mathcal{T}$ is ultimately periodic w.r.t. concept inclusions. We fix an $A \in \mathsf{N_C}(\mathcal{T})$. By the assumption and Lemma B.2, for each $B \in \mathsf{N_C}(\mathcal{T})$ the set $\{n \in \mathbb{Z} \mid B \in \pi_A(n)\} = \{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\}$ is semilinear. By Lemma B.4 it is equal to $\mathcal{L}_1 \cup \cdots \cup \mathcal{L}_m$, where $\mathcal{L}_i$ are some simple sets. Fix such a representation for each $B \in \mathsf{N_C}(\mathcal{T})$, and let $\mathcal{L}_1, \dots, \mathcal{L}_s$ be all simple sets that appear in these representations, with $\mathcal{L}_i = \{b_i + kp_i \mid k \in \mathbb{N}\}$.

$$m_P, m_F = \max_{1 \leqslant i \leqslant s} b_i \qquad\qquad p_P, p_F = \prod_{i=1}^{s} p_i$$

It is easy to see that the conditions (B.1)-(B.2) hold for π_A with these m_P, p_P, m_F, p_F. □

B.2 Proofs for Section 3.2

Lemma 3.2. *$\mathfrak{M}_{(\mathcal{T},\mathcal{D})} \models (\mathcal{T},\mathcal{D})$ and for every $\mathfrak{M} \models (\mathcal{T},\mathcal{D})$, there is a homomorphism $h : \mathsf{N_I} \cup \mathsf{N_N} \mapsto \Delta_{\mathfrak{M}}$ from $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ to $\mathfrak{M}$.*

Proof. It is easy to check that $\mathfrak{M}_{(\mathcal{T},\mathcal{D})} \models (\mathcal{T},\mathcal{D})$: $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ is a model of all facts in $\mathcal{D}$ by construction of $\mathcal{M}_0$, and if $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ was not a model of some concept inclusion of $\mathcal{T}$, or if there was a rigid role name whose interpretation changed over time, this would imply that a rule of one the forms (i)–(v) is applicable in $\mathcal{M}$, contradicting the definition of $\mathcal{M}$.

Let $\mathfrak{M}$ be a model of $(\mathcal{T},\mathcal{D})$. We show how to inductively construct a homomorphism h from $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ to $\mathfrak{M}$, i.e., a homomorphism h from $\mathcal{M}$ to the set of atoms $\mathcal{I}$ such that for every $n \in \mathbb{Z}$, $d \in A^{\mathfrak{M},n}$ iff $A(d,n) \in \mathcal{I}$ and $(d,e) \in r^{\mathfrak{M},n}$ iff $r(d,e,n) \in \mathcal{I}$. Let

$h_0 : \mathsf{N_I} \mapsto \Delta_{\mathfrak{M}}$ be the identity (recall that $\mathsf{N_I} \subseteq \Delta_{\mathfrak{M}}$ by the standard name assumption). Since $\mathfrak{M} \models \mathcal{D}$, $\mathcal{D} \subseteq \mathcal{I}$ so h_0 is a homomorphism from $\mathcal{M}_0 = \mathcal{D}$ to $\mathcal{I}$. Assume that we have built a homomorphism $h_i : \mathsf{N_I} \cup \{e \mid e \in \mathsf{N_N}, e \text{ occurs in } \mathcal{M}_i\} \mapsto \Delta_{\mathfrak{M}}$ from $\mathcal{M}_i$ to $\mathcal{I}$ and consider $\mathcal{M}_{i+1}$. We distinguish two cases:

- If $\mathcal{M}_{i+1}$ has been obtained from $\mathcal{M}_i$ by applying a rule of one of the forms (i)–(iv), let $h_{i+1} = h_i$. It is easy to verify that in any case, h_{i+1} is a homomorphism from $\mathcal{M}_{i+1}$ to $\mathcal{I}$. This follows from the facts that h_i is a homomorphism from $\mathcal{M}_i$ to $\mathcal{I}$ and that $\mathfrak{M} \models \mathcal{T}$ and respects rigid roles.

- Otherwise, $\mathcal{M}_{i+1}$ has been obtained from $\mathcal{M}_i$ by applying a rule of form (v): there are $A(a,n) \in \mathcal{M}_i$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$, and $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{r(a,b,n), B(b,n)\}$ for some $b \in \mathsf{N_N}$ which does not occur in $\mathcal{M}_i$. Since h_i is a homomorphism from $\mathcal{M}_i$ to $\mathcal{I}$, $A(h_i(a),n) \in \mathcal{I}$. Hence, since $\mathfrak{M} \models A \sqsubseteq \exists r.B$, there is $d \in \Delta_{\mathfrak{M}}$ such that $r(h_i(a),d,n)$ and $B(d,n)$ are in $\mathcal{I}$. We define $h_{i+1} : \mathsf{N_I} \cup \{e \mid e \in \mathsf{N_N}, e \text{ occurs in } \mathcal{M}_{i+1}\} \mapsto \Delta_{\mathfrak{M}}$ by $h_{i+1}(x) = h_i(x)$ for every $x \in \mathsf{N_I} \cup \{e \mid e \in \mathsf{N_N}, e \text{ occurs in } \mathcal{M}_i\}$ and $h_{i+1}(b) = d$. It is easy to check that h_{i+1} is a homomorphism from $\mathcal{M}_{i+1}$ to $\mathcal{I}$.

We obtain a homomorphism $h : \mathsf{N_I} \cup \mathsf{N_N} \mapsto \Delta_{\mathfrak{M}}$ from $\mathfrak{M}_{(\mathcal{T},\mathcal{D})}$ to $\mathfrak{M}$ by setting $h = \bigcup_{i \geqslant 0} h_i$ and extending h to the nulls that do not occur in $\mathcal{M}$ by mapping them to any element of $\Delta_{\mathfrak{M}}$. □

Proposition 3.3. *For every $\mathcal{TEL}^\circ$ TBox $\mathcal{T}$, database $\mathcal{D}$, $A, B \in \mathsf{N_C}$, $a \in \mathsf{N_I}$, and $n, k \in \mathbb{Z}$:*

- $(\mathcal{T},\mathcal{D}) \models A(a,n)$ *iff* $(\mathcal{T},\mathcal{D}) \vdash A(a,n)$, *and*
- $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ *iff* $(\mathcal{T}, \{A(a,k)\}) \vdash B(a,k+n)$.

Proposition 3.3 follows from the next two lemmas.

Lemma B.6. *For every $A \in \mathsf{N_C}$, $a \in \mathsf{N_I}$, and $n \in \mathbb{Z}$, $(\mathcal{T},\mathcal{D}) \models A(a,n)$ iff $(\mathcal{T},\mathcal{D}) \vdash A(a,n)$.*

Proof. Since $\mathcal{M}_0 = \mathcal{D}$ and rules of form (i)–(v) correspond exactly to derivation rules of form (3.7)–(3.11), it is easy to see that $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$ iff $A(a, n) \in \mathcal{M}$, i.e., $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$ iff $a \in A^{\mathfrak{M}_{(\mathcal{T},\mathcal{D})},n}$. Since by Lemma 3.2, $a \in A^{\mathfrak{M}_{(\mathcal{T},\mathcal{D})},n}$ implies that $a \in A^{\mathfrak{M},n}$ for every $\mathfrak{M} \models (\mathcal{T}, \mathcal{D})$, the result follows. □

Lemma B.7. *For every $A, B \in \mathsf{N_C}$, $a \in \mathsf{N_I}$, and $n, k \in \mathbb{Z}$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ iff $(\mathcal{T}, \{A(a, k)\}) \vdash B(a, k + n)$.*

Proof. By Lemma B.6, $(\mathcal{T}, \{A(a, k)\}) \vdash B(a, k + n)$ iff $(\mathcal{T}, \{A(a, k)\}) \models B(a, k + n)$. We show that $(\mathcal{T}, \{A(a, k)\}) \models B(a, k + n)$ iff $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$.
($\Leftarrow$) If $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$, every model $\mathfrak{M}$ of $(\mathcal{T}, \{A(a, k)\})$ is such that $\mathfrak{M} \models A(a, k)$ and $\mathfrak{M} \models A \sqsubseteq \bigcirc^n B$, hence $\mathfrak{M} \models B(a, k + n)$.
($\Rightarrow$) Assume that $\mathcal{T} \not\models A \sqsubseteq \bigcirc^n B$: there exists a model $\mathfrak{M} = (\mathcal{I}_i)_{i\in\mathbb{Z}}$ of $\mathcal{T}$ with $e \in \Delta_{\mathfrak{M}}$ and $j \in \mathbb{Z}$ such that $e \in A^{\mathfrak{M},j}$ and $e \notin B^{\mathfrak{M},j+n}$. Let $\mathfrak{M}' = (\mathcal{I}'_i)_{i\in\mathbb{Z}}$ be the interpretation obtained from $\mathfrak{M}$ by switching e and a in all concept and role interpretations (note that since $a \in \mathsf{N_I}$, $a \in \Delta_{\mathfrak{M}}$), and let $\mathfrak{M}'' = (\mathcal{I}''_i)_{i\in\mathbb{Z}}$ where $\mathcal{I}''_i = \mathcal{I}'_{i+k-j}$ for every $i \in \mathbb{Z}$. It is easy to see that $\mathfrak{M}'' \models (\mathcal{T}, \{A(a, k)\})$ while $\mathfrak{M}'' \not\models B(a, k + n)$. Hence, $(\mathcal{T}, \{A(a, k)\}) \not\models B(a, k + n)$. □

B.3 Additional notation and conventions

Derivations Recall that a derivation witnessing $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$ is a sequence $(\mathcal{F}_0, \ldots, \mathcal{F}_m)$ such that $\mathcal{F}_0 = \mathcal{D} \cup \mathcal{T}$, $A(a, n) \in \mathcal{F}_m$ and $\mathcal{F}_i$ is obtained from $\mathcal{F}_{i-1}$ by applying a rule of the form (3.7)–(3.11). It will be convenient to represent such $(\mathcal{F}_0, \ldots, \mathcal{F}_m)$ as

$$\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$$

where each *rule application* is represented by a pair of sets of formulas $f_i = (\mathsf{prem}(f_i), \mathsf{conc}(f_i))$ where $\mathsf{prem}(f_i) \subseteq \mathcal{F}_{i-1}$ is the *premise* of f_i and $\mathsf{conc}(f_i) = \mathcal{F}_i \setminus \mathcal{F}_{i-1}$ is its *conclusion*, which match, respectively, the left and the right sides of the rule applied to get $\mathcal{F}_i$

from $\mathcal{F}_{i-1}$. We say that f_i *uses* the formulas in $\mathsf{prem}(f_i)$ and *produces* those in $\mathsf{conc}(f_i)$. Similarly, given $G = (N, \Sigma, R)$, a derivation witnessing $G \vdash X(w)$ can be represented as

$$\mathcal{G}_0 \xrightarrow{g_1} \ldots \xrightarrow{g_m} \mathcal{G}_m$$

where $\mathcal{G}_0 = \{c(c) \mid c \in \Sigma\}$, $X(w) \in \mathcal{G}_m$, and each rule application g_i refers to one of the rules defined by deduction rules of Section 2.2.

Moreover, *we extend the notion of derivation to derivations of formulas of the form $A(b, n)$ where $b \in \mathsf{N_N}$* (instead of $\mathsf{N_I}$) from a set of formulas $\mathcal{F}_0$ in which b occurs. Hence, one can write, e.g., $(\mathcal{T}, \{(A(b, k)\}) \vdash B(b, n + k)$ for $b \in \mathsf{N_N}$.

Extension and use of Proposition 3.3 In the proofs given in the next sections, Proposition 3.3 allows us to equivalently write $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$, $(\mathcal{T}, \{A(a, 0)\}) \models B(a, n)$, or $(\mathcal{T}, \{A(a, 0)\}) \vdash B(a, n)$. Moreover, one can use $a \in \mathsf{N_I} \cup \mathsf{N_N}$ in the last formula since it is easy to see by considering derivations that $(\mathcal{T}, \{A(a, 0)\}) \vdash B(a, n)$ for $a \in \mathsf{N_I}$ iff $(\mathcal{T}, \{A(b, 0)\}) \vdash B(b, n)$ for $b \in \mathsf{N_N}$.

B.4 Proofs for Section 3.4

B.4.1 Proof of Theorem 3.6

We start with some lemmas. Recall that given a $\mathcal{TEL}^\bigcirc$-TBox $\mathcal{T}$, $\mathcal{T}_{rig}$ is the TBox defined right after Theorem 3.6 in Section 3.4.

Lemma B.8. *The following statements hold.*

(i) Given a derivation $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ witnessing $(\mathcal{T}, \{A(a, 0)\}) \vdash B(a, n)$, if $r(b_1, b_2, k) \in \mathcal{F}_i$ and $r(b_1, b_2, k') \in \mathcal{F}_j$, for some $i, j \in \{1, \ldots, m\}$ and $r \in \mathsf{N_R^{loc}}$, then $k' = k$.

(ii) Given a derivation $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ witnessing $(\mathcal{T}_{rig}, \{A(a, 0)\}) \vdash B(a, n)$, if $C_r(b, k) \in \mathcal{F}_i$ and $C_r(b, k') \in \mathcal{F}_j$, for some $i, j \in \{1, \ldots, m\}$, then $k' = k$.

Proof. (i) Since $\mathcal{F}_0 = \{A(a,0)\} \cup \mathcal{T}$, by the form of the derivation rules (cf. (3.7)–(3.11)), $b_2 \in \mathsf{N_N}$. Suppose $k \neq k'$ and let i, j be the least indexes such that $r(b_1, b_2, k) \in \mathcal{F}_i$ and $r(b_1, b_2, k') \in \mathcal{F}_j$. Since a rule application produces several role facts only if the rule is of form (3.7), and $r \in \mathsf{N_R^{loc}}$, it must be the case that $i \neq j$. Suppose $i < j$ and observe that $r(b_1, b_2, k')$ is produced by the application f_j of a rule of form (3.11). Then b_2 should not appear in $\mathcal{F}_i$, but it does. Hence $k = k'$.
(ii) The proof is analogous to that of *(i)*, using the fact that C_r only occurs in the right-hand side of concept inclusions of $\mathcal{T}_{rig}$ in concept inclusions of the form $A \sqsubseteq \exists r.C_r$, so that a fact of the form $C_r(b, \ell)$ can only be produced by an application of a rule of form (3.11) which introduces b as a new null not appearing before. □

Lemma 3.7. *Let $\mathcal{T}$ be a $\mathcal{TEL}^\circ$ TBox. For any $A, B \in \mathsf{N_C}(\mathcal{T})$ and $n \in \mathbb{Z}$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ if and only if $\mathcal{T}_{rig} \models A \sqsubseteq \bigcirc^n B$.*

Proof. ($\Rightarrow$) Let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$. We build a derivation witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$. Let $(h_1, \ldots, h_p)$ be the sequence of rule applications obtained from $(f_1, \ldots, f_m)$ by applying the following steps.

1. Substitute every application of a rule of form (3.11) with $r \in \mathsf{N_R^{loc}}$, $f_i = (\{C \sqsubseteq \exists r.D, C(e,\ell)\}, \{r(e,d,\ell), D(d,\ell)\})$, by consecutive applications of rules of form (3.11) and (3.8), f_i', f_i'', where $f_i' = (\{C \sqsubseteq \exists r.D_r, C(e,\ell)\}, \{r(e,d,\ell), D_r(d,\ell)\})$ and $f_i'' = (\{D_r \sqsubseteq D, D_r(e,\ell)\}, \{D(d,\ell)\})$.

2. For every application of rule of form (3.10) with $r \in \mathsf{N_R^{loc}}$, $f_i = (\{r(e,d,\ell), C(d,\ell), \exists r.C \sqsubseteq D\}, \{D(e,\ell)\})$, by Lemma B.8, there is no $\ell' \neq \ell$ such that $r(e,d,\ell')$ belongs to any $\mathcal{F}_j$, so since $r(e,d,\ell)$ has been produced by the application of a rule of form (3.11), f_j, with $j < i$, there exists E_r such that $E_r(d,\ell)$ is in the conclusion of f_j' defined in point 1. Substitute f_i by consecutive applications of rules of form (3.9) and (3.10),

f'_i, f''_i, where $f'_i = (\{C(d,\ell), E_r(d,\ell), C \sqcap E_r \sqsubseteq C'_r\}, \{C'_r(d,\ell)\})$ and $f''_i = (\{r(e,d,\ell), C'_r(d,\ell), \exists r.C'_r \sqsubseteq D\}, \{D(e,\ell)\})$.

3. Substitute every occurrence of r in the resulting sequence of rule applications by the $r' \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}}$ used in $\mathcal{T}_{rig}$.

Since all concept inclusions used in the premises of $h_1, \ldots, h_p$ belongs to $\mathcal{T}_{rig}$ by construction, we indeed obtain a derivation $\mathcal{F}'_0 \xrightarrow{h_1} \ldots \xrightarrow{h_p} \mathcal{F}'_p$ witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$ by setting $\mathcal{F}'_0 = \{A(a,0)\} \cup \mathcal{T}_{rig}$ and $\mathcal{F}'_i = \mathcal{F}'_{i-1} \cup \mathsf{conc}(h_i)$.

($\Leftarrow$) Suppose $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ is a derivation witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$. We build a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$. Let $(h_1, \ldots, h_p)$ be the sequence of rule applications obtained from $(f_1, \ldots, f_m)$ by applying the following steps.

1. Restore each r' of $\mathcal{T}_{rig}$ to the original $r \in \mathsf{N}_\mathsf{R}^{\mathsf{loc}}$ and omit every application of a rule of form (3.7) with r'.

2. Omit every application of a rule of form (3.8) that uses $D_r \sqsubseteq D$, and substitute every application of a rule of form (3.11) that uses $C \sqsubseteq \exists r.D_r$, $f_i = (\{C(e,\ell), C \sqsubseteq \exists r.D_r\}, \{r(e,d,\ell), D_r(d,\ell)\})$, with one using $C \sqsubseteq \exists r.D$ instead: $f'_i = (\{C(e,\ell), C \sqsubseteq \exists r.D\}, \{r(e,d,\ell), D(d,\ell)\})$.

3. Omit every application of a rule of form (3.9) that uses a concept inclusion of the form $C \sqcap D_r \sqsubseteq C'_r$.

4. For every application of a rule of form (3.10) that uses $\exists r.C'_r \sqsubseteq D$, $f_i = (\{r(e,d,\ell), C'_r(d,\ell), \exists r.C'_r \sqsubseteq D\}, \{D(e,\ell)\})$, since $C'_r(d,\ell) \in \mathcal{F}_{i-1}$, then

   (a) $C(d,\ell) \in \mathcal{F}_{i-1}$ and there exists E_r such that $E_r(d,\ell) \in \mathcal{F}_{i-1}$ (since C'_r occurs in the right-hand side of concept inclusions in $\mathcal{T}_{rig}$ only in concept inclusions of the form $C \sqcap E_r \sqsubseteq C'_r$), and

(b) by Lemma B.8, there is no $\ell' \neq \ell$ such that $E_r(d, \ell')$ belongs to any $\mathcal{F}_j$, so since $E_r(d, \ell)$ has been produced by the application of a rule of form (3.11), f_j, with $j < i-1$, $r(e, d, \ell) \in \mathcal{F}_{i-1}$ has been produced by f'_j defined in point 2[1] (note that since d has been introduced by f'_j, there cannot be any $r(e', d, \ell) \in \mathcal{F}_{i-1}$ with $e \neq e'$).

Substitute f_i with $f'_i = (\{r(e, d, \ell), C(d, \ell), \exists r.C \sqsubseteq D\}, \{D(e, \ell)\})$.

Once again, we can check that all concept inclusions used in the premises of $h_1, \ldots, h_p$ belongs to $\mathcal{T}$ so that we indeed obtain a derivation $\mathcal{F}'_0 \xrightarrow{h_1} \ldots \xrightarrow{h_p} \mathcal{F}'_p$ witnessing $(\mathcal{T}, \{A(a, 0)\}) \vdash B(a, n)$ by setting $\mathcal{F}'_0 = \{A(a, 0)\} \cup \mathcal{T}$ and $\mathcal{F}'_i = \mathcal{F}'_{i-1} \cup \mathsf{conc}(h_i)$. □

Now we are ready to prove Theorem 3.6.

Theorem 3.6 (TBoxes to Grammars)**.** *For every $\mathcal{TEL}^{\circ}_{future}$ TBox $\mathcal{T}$, one can construct in polynomial time a unary conjunctive grammar $G_{\mathcal{T}} = (N, \{c\}, R)$ such that for any $A, B \in \mathsf{N_C}(\mathcal{T})$, there is $\mathcal{N}_{AB} \in N$ such that $c^n \in L_{G_{\mathcal{T}}}(\mathcal{N}_{AB})$ iff $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$.*

Proof. Let $G_{\mathcal{T}}$ be the grammar defined from $\mathcal{T}$ in Definition 3.8. We show that for every $A, B \in \mathsf{N_C}(\mathcal{T}_{rig})$, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, and $n \in \mathbb{N}$, $G_{\mathcal{T}} \vdash \mathcal{N}_{AB}(c^n)$ iff $(\mathcal{T}_{rig}, \{A(a, 0)\}) \vdash B(a, n)$. The result will follow by Lemma 3.7 and the fact that $\mathsf{N_C}(\mathcal{T}) \subseteq \mathsf{N_C}(\mathcal{T}_{rig})$.

($\Leftarrow$) We show by induction on m that for every $A, B \in \mathsf{N_C}(\mathcal{T}_{rig})$, $a \in \mathsf{N_I} \cup \mathsf{N_N}$, and $n \in \mathbb{N}$, if there exists a derivation witnessing $(\mathcal{T}_{rig}, \{A(a, 0)\}) \vdash B(a, n)$ of length at most m, then $G_{\mathcal{T}} \vdash \mathcal{N}_{AB}(c^n)$.

In the base case, $m = 0$, the derivation consists only of $\mathcal{F}_0 = \mathcal{T}_{rig} \cup \{A(a, 0)\}$, so $B(a, n) \in \mathcal{F}_0$ implies that $A = B$ and $n = 0$, and by (3.14), $\mathcal{N}_{AA} \to \varepsilon$ is a rule of $G_{\mathcal{T}}$ so $G_{\mathcal{T}} \vdash \mathcal{N}_{AB}(\varepsilon)$.

Induction step: assume that the property holds for $m - 1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing $(\mathcal{T}_{rig}, \{A(a, 0)\}) \vdash$

[1] This ensures that $r(e, d, \ell)$ is not "lost" in the derivation when we omit every application of a rule of form (3.7) with r'.

$B(a,n)$. If $B(a,n) \in \mathcal{F}_{m-1}$, the result follows by induction hypothesis. Otherwise, there are three possible cases for the last rule application f_m that produces $B(a,n)$:

- $\mathsf{prem}(f_m) = \{A'(a,n-k), A' \sqsubseteq \bigcirc^k B\}$ for some $k \in \{0,\dots,n\}$, and f_m is the application of a rule of form (3.8). Since $(\mathcal{F}_0,\dots,\mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash A'(a,n-k)$, by induction hypothesis, $G_\mathcal{T} \vdash \mathcal{N}_{AA'}(c^{n-k})$. Moreover, since $A' \sqsubseteq \bigcirc^k B \in \mathcal{T}_{rig}$, by (3.14) or (3.15) depending on k, $\mathcal{N}_{A'B} \to c^k$ is a rule of $G_\mathcal{T}$. Hence $G_\mathcal{T} \vdash \mathcal{N}_{A'B}(c^k)$. Then, since by (3.18) $\mathcal{N}_{AB} \to \mathcal{N}_{AA'}\mathcal{N}_{A'B}$ is a rule of $G_\mathcal{T}$, we obtain $G_\mathcal{T} \vdash \mathcal{N}_{AB}(c^n)$.

- $\mathsf{prem}(f_m) = \{A'(a,n), A''(a,n), A' \sqcap A'' \sqsubseteq B\}$ and f_m is the application of a rule of form (3.9). Since $A'(a,n)$ and $A''(a,n)$ are in $\mathcal{F}_{m-1}$, $(\mathcal{F}_0,\dots,\mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash A'(a,n)$ and $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash A''(a,n)$. By the induction hypothesis, $G_\mathcal{T} \vdash \mathcal{N}_{AA'}(c^n)$ and $G_\mathcal{T} \vdash \mathcal{N}_{AA''}(c^n)$, and since $A' \sqcap A'' \sqsubseteq B \in \mathcal{T}_{rig}$, by (3.16), $\mathcal{N}_{AB} \to \mathcal{N}_{AA'} \& \mathcal{N}_{AA''}$ is a rule of $G_\mathcal{T}$ so $G_\mathcal{T} \vdash \mathcal{N}_{AB}(c^n)$.

- $\mathsf{prem}(f_m) = \{r(a,b,n), A'(b,n), \exists r.A' \sqsubseteq B\}$ and f_m is the application of a rule of form (3.10). By the form of the derivation rules, $r(a,b,n) \in \mathcal{F}_{m-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < m-1$ such that for some A'', B' and $k \in \mathbb{N}$, $\mathsf{prem}(f_i) = \{A'' \sqsubseteq \exists r.B', A''(a,k)\}$, $\mathsf{conc}(f_i) = \{r(a,b,k), B'(b,k)\}$. Thus $A''(a,k) \in \mathcal{F}_{i-1}$, and $(\mathcal{F}_0,\dots,\mathcal{F}_{i-1})$ is a derivation witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash A''(a,k)$. By the induction hypothesis, we obtain $G_\mathcal{T} \vdash \mathcal{N}_{AA''}(c^k)$. Moreover, one can extract from $(\mathcal{F}_i,\dots,\mathcal{F}_{m-1})$ a derivation of length at most $m-1$ witnessing $(\mathcal{T}_{rig}, \{B'(b,k)\}) \vdash A'(b,n)$. Since $\mathcal{T}$ is a $\mathcal{TEL}^\circ_{future}$-TBox, $k \leqslant n$. We get a derivation length at most $m-1$ witnessing $(\mathcal{T}_{rig}, \{B'(b,0)\}) \vdash A'(b,n-k)$ by shifting all timestamps in this derivation, and thus, by the induction hypothesis, $G_\mathcal{T} \vdash \mathcal{N}_{B'A'}(c^{n-k})$. Since $A'' \sqsubseteq \exists r.B'$ and

$\exists r.A' \sqsubseteq B$ are in $\mathcal{T}_{rig}$, by (3.17), $\mathcal{N}_{A''B} \to \mathcal{N}_{B'A'}$ is a rule of $G_\mathcal{T}$, so from $G_\mathcal{T} \vdash \mathcal{N}_{B'A'}(c^{n-k})$, we get $G_\mathcal{T} \vdash \mathcal{N}_{A''B}(c^{n-k})$. We combine this with $G_\mathcal{T} \vdash \mathcal{N}_{AA''}(c^k)$ and the fact that by (3.18), $\mathcal{N}_{AB} \to \mathcal{N}_{AA''}\mathcal{N}_{A''B}$ is a rule of $G_\mathcal{T}$ to establish $G_\mathcal{T} \vdash \mathcal{N}_{AB}(c^n)$.

($\Rightarrow$) We show by induction on m that for every $A, B \in \mathsf{N_C}(\mathcal{T}_{rig})$, $a \in \mathsf{N_I} \cup \mathsf{N_N}$, and $n \in \mathbb{N}$, if there exists a derivation witnessing $G_\mathcal{T} \vdash \mathcal{N}_{AB}(c^n)$ of length at most m, then $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$.

Since $\mathcal{G}_0 = \{c(c)\}$, the base case is $m = 1$, when the derivation consists of $\mathcal{G}_0 \xrightarrow{g_1} \mathcal{G}_1$. There are two possible cases for the rule application g_1 that produces $\mathcal{N}_{AB}(c^n)$ from $c(c)$:

- g_1 is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \varepsilon$. In this case, $n = 0$ and by (3.14), $A \sqsubseteq B \in \mathcal{T}_{rig}$ or $A = B$. In both cases, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,0)$, i.e. $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$.

- g_1 is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to c^n$. In this case, by (3.15), $A \sqsubseteq \bigcirc^n B \in \mathcal{T}_{rig}$. Hence, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$.

Induction step: assume that the property holds for $m-1$ and let $\mathcal{G}_0 \xrightarrow{g_1} \ldots \xrightarrow{g_m} \mathcal{G}_m$ be a derivation witnessing $G_\mathcal{T} \vdash \mathcal{N}_{AB}(c^n)$. If $\mathcal{N}_{AB}(c^n) \in \mathcal{G}_{m-1}$, the result follows by induction hypothesis. Otherwise, there are three possible cases for the last rule application g_m that produces $\mathcal{N}_{AB}(c^n)$:

- $\mathsf{prem}(g_m) = \{\mathcal{N}_{AC}(c^n), \mathcal{N}_{AD}(c^n)\}$, and g_m is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \mathcal{N}_{AC} \& \mathcal{N}_{AD}$. By (3.16), $C \sqcap D \sqsubseteq B \in \mathcal{T}_{rig}$. Then $(\mathcal{G}_0, \ldots, \mathcal{G}_{m-1})$ is a derivation witnessing $G \vdash \mathcal{N}_{AC}(c^n)$ and $G \vdash \mathcal{N}_{AD}(c^n)$, so by the induction hypothesis, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash C(a,n)$ and $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash D(a,n)$. Hence $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$.

- $\mathsf{prem}(g_m) = \{\mathcal{N}_{CD}(c^n)\}$, and g_m is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \mathcal{N}_{CD}$. By (3.17),

there exists r such that $A \sqsubseteq \exists r.C$ and $\exists r.D \sqsubseteq B$ are in $\mathcal{T}_{rig}$. Then $(\mathcal{G}_0, \ldots, \mathcal{G}_{m-1})$ is a derivation witnessing $G \vdash \mathcal{N}_{CD}(c^n)$ so by the induction hypothesis, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}_{rig}, \{C(a,0)\}) \vdash D(a,n)$. Let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}_p$ be obtained from that derivation by substituting a everywhere with a $b \in \mathsf{N_N}$ not appearing in the derivation. Then the following is a derivation witnessing $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$:

$$\mathcal{F} \xrightarrow{f_0} \mathcal{F}'_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}'_p \xrightarrow{f_{p+1}} \mathcal{F}_{p+1} \xrightarrow{f_{p+2}} \mathcal{F}_{p+2}$$

where

- $\mathcal{F} = \{A(a,0)\} \cup \mathcal{T}_{rig}$;
- f_0 is an application of the rule of form (3.11) with $\mathsf{prem}(f_0) = \{A(a,0), A \sqsubseteq \exists r.C\}$ and $\mathsf{conc}(f_0) = \{r(a,b,0), C(b,0)\}$;
- $\mathcal{F}'_i = \mathcal{F}_i \cup \{A(a,0), r(a,b,0)\}$ for $0 \leqslant i \leqslant p$;
- f_{p+1} is an application of the rule of form (3.7) with $\mathsf{prem}(f_{p+1}) = \{r(a,b,0)\}$ and $\mathsf{conc}(f_{p+1}) = \{r(a,b,n)\}$;
- f_{p+2} is an application of the rule of form (3.10) with $\mathsf{prem}(f_{p+2}) = \{r(a,b,n), D(b,n), \exists r.D \sqsubseteq B\}$ and $\mathsf{conc}(f_{p+2}) = \{B(a,n)\}$;
- $\mathcal{F}_{p+1} = \mathcal{F}'_p \cup \mathsf{conc}(f_{p+1})$ and $\mathcal{F}_{p+2} = \mathcal{F}_{p+1} \cup \mathsf{conc}(f_{p+2})$.

- $\mathsf{prem}(g_m) = \{\mathcal{N}_{AC}(c^{n-k}), \mathcal{N}_{CB}(c^k)\}$ for some $k \in \{0, \ldots, n\}$, and g_m is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \mathcal{N}_{AC}\mathcal{N}_{CB}$. By (3.18) $A, B, C \in \mathsf{N_C}(\mathcal{T}_{rig})$. Then $(\mathcal{G}_0, \ldots, \mathcal{G}_{m-1})$ is a derivation witnessing $G \vdash \mathcal{N}_{AC}(c^{n-k})$ and $G \vdash \mathcal{N}_{CB}(c^k)$. By the induction hypothesis, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash C(a, n-k)$ and $(\mathcal{T}_{rig}, \{C(a,0)\}) \vdash B(a,k)$, which is equivalent (by Proposition 3.3) to $(\mathcal{T}_{rig}, \{C(a,n-k)\}) \vdash B(a,n)$. Hence, $(\mathcal{T}_{rig}, \{A(a,0)\}) \vdash B(a,n)$. □

B.4.2 Proof of Theorem 3.10

Again, we start with somme lemmas. Recall that given a unary conjunctive grammar G, the $\mathcal{TEL}^{\circ}_{future}$-TBox $\mathcal{T}_G$ is defined by Definition 3.11.

Lemma B.9. *For $\iota = i_1 \dots i_k \in \mathcal{J}$, if $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_\iota$ then $n = n_1 + \dots + n_k$, such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ for $1 \leqslant j \leqslant k$. Moreover, if there exists a derivation for $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_\iota(a,n)$ of length p, then for every for $1 \leqslant j \leqslant k$, there exists a derivation for $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_j}(a,n_j)$ of length at most $p-1$.*

Proof. We show by induction on k that for every $\iota \in \mathcal{J}$ of length k, if $\iota = i_1 \dots i_k$, for every $n \in \mathbb{N}$, if $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_\iota$, then there exist $n_1, \dots, n_k$ such that $n = n_1 + \dots + n_k$ and $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ for $1 \leqslant j \leqslant k$, and that if there exists a derivation for $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_\iota(a,n)$ of length p, then for every for $1 \leqslant j \leqslant k$, there exists a derivation for $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_j}(a,n_j)$ of length at most $p-1$.

In the base case, $k = 1$ and $\iota = i_1$ (hence $n_1 = n$). If $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_{i_1}$, i.e., $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_{i_1}(a,n)$, since the only concept inclusion in $\mathcal{T}_G$ with C_{i_1} in the right-hand side is $B_{i_1} \sqsubseteq C_{i_1}$ (cf. Definition 3.11), any derivation $\mathcal{F}_0 \xrightarrow{f_1} \dots \xrightarrow{f_p} \mathcal{F}_p$ witnessing $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_\iota(a,n)$ must contain a derivation (of length at most $p-1$) witnessing $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_1}(a,n)$. Hence, $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B_{i_1}$.

Induction step: assume that the property holds for $k-1$ and let $\iota = i_1 \jmath \in \mathcal{J}$ be of length k. Note that $\jmath = i_2 \dots i_k \in \mathcal{J}$ and is of length $k-1$. Suppose $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_{i_1\jmath}(a,n)$ and let $\mathcal{F}_0 \xrightarrow{f_1} \dots \xrightarrow{f_p} \mathcal{F}_p$ be a derivation witnessing it. Necessarily, the rule application that produces $C_{i_1\jmath}(a,n)$ uses the concept inclusion $\exists r_{i_1\jmath}.C_\jmath \sqsubseteq C_{i_1\jmath}$ since it is the only one with $C_{i_1\jmath}$ in the right-hand side by construction of $\mathcal{T}_G$. Thus there exists b such that $r_{i_1\jmath}(a,b,n)$ and $C_\jmath(b,n)$ are in $\mathcal{F}_{p-1}$.

- Let f_l be the application of the rule of form (3.11) that produced the first fact of the form $r_{i_1\jmath}(a,b,n_1)$ ($n_1 \in \mathbb{N}$ and

$l \leqslant p-1$): by construction of $\mathcal{T}_G$, it must be the case that $\mathsf{prem}(f_l) = \{B_{i_1} \sqsubseteq \exists r_{i_1 \jmath}.A, B_{i_1}(a, n_1)\}$ and $\mathsf{conc}(f_l) = \{r_{i_1 \jmath}(a, b, n_1), A(b, n_1)\}$. From $(\mathcal{F}_l, \dots, \mathcal{F}_{p-1})$, one obtains a derivation of length at most $p-1$ witnessing $(\mathcal{T}_G, \{A(b,0)\}) \vdash C_\jmath(b, n-n_1)$ by shifting all timestamps by $-n_1$. By construction of $\mathcal{T}_G$ (in particular, because it is a $\mathcal{TEL}^{\bigcirc}_{future}$-TBox), it must be the case that $n - n_1 \geqslant 0$. Hence, by the induction hypothesis, there exist $n_2, \dots, n_k$ such that $n - n_1 = n_2 + \dots + n_k$, and for $2 \leqslant j \leqslant k$, $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ and there exists a derivation of length at most $p-1$ witnessing $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_j}(a, n_j)$.

- Furthermore, as $B_{i_1}(a, n_1) \in \mathsf{prem}(f_l) \subseteq \mathcal{F}_{l-1}$, $(\mathcal{F}_0, \dots, \mathcal{F}_{l-1})$ is a derivation of length at most $p-1$ witnessing $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_1}(a, n_1)$, and $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_1} B_{i_1}$.

Hence $n = n_1 + \dots + n_k$ and for $1 \leqslant j \leqslant k$, $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ and there exists a derivation of length at most $p-1$ witnessing $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_j}(a, n_j)$. □

Lemma 3.13. *For $\iota = i_1 \dots i_k \in \mathcal{J}$, $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_\iota$ if and only if $n = n_1 + \dots + n_k$, such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ for $1 \leqslant j \leqslant k$.*

Proof. $\Rightarrow$ The 'only if' direction is given by Lemma B.9.
$\Leftarrow$ We show by induction on k that for every $\iota \in \mathcal{J}$ of length k, if $\iota = i_1 \dots i_k$, for every $n \in \mathbb{N}$, if there exist $n_1, \dots, n_k$ such that $n = n_1 + \dots + n_k$ and $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ for $1 \leqslant j \leqslant k$, then $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_\iota$.

In the base case, $k = 1$ and $\iota = i_1$. If $(\mathcal{T}_G, \{A(a,0)\}) \models B_{i_1}(a, n)$, then $(\mathcal{T}_G, \{A(a,0)\}) \models C_{i_1}(a, n)$ since $B_{i_1} \sqsubseteq C_{i_1} \in \mathcal{T}_G$ by (25*).

Induction step: assume that the property holds for $k-1$ and let $\iota = i_1 \jmath \in \mathcal{J}$ be of length k. Note that $\jmath = i_2 \dots i_k \in \mathcal{J}$ and is of length $k-1$. Suppose that $n = n_1 + \dots + n_k$ and $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_j} B_{i_j}$ for $1 \leqslant j \leqslant k$.

- Since $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n_1} B_{i_1}$, for every $a \in \mathsf{N_I}$, $(\mathcal{T}_G, \{A(a,0)\}) \vdash$

$B_{i_1}(a, n_1)$. Let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing it.

- Since $i_1\jmath \in \mathcal{J}$, by (23^*), $B_{i_1} \sqsubseteq \exists r_{i_1\jmath}.A \in \mathcal{T}_G$. Let $\mathcal{F}_{m+1} = \mathcal{F}_m \cup \mathsf{conc}(f_{m+1})$ where $f_{m+1} = (\{B_{i_1} \sqsubseteq \exists r_{i_1\jmath}.A, B_{i_1}(a, n_1)\}, \{r_{i_1\jmath}(a, b, n_1), A(b, n_1)\})$.

- By the induction hypothesis: $\mathcal{T}_G \models A \sqsubseteq \bigcirc^{n'} C_\jmath$ for $n' = n_2 + \cdots + n_k$. Hence, $(\mathcal{T}_G, \{A(b, 0)\}) \vdash C_\jmath(b, n')$, and by Proposition 3.3, $(\mathcal{T}_G, \{A(b, n_1)\}) \vdash C_\jmath(b, n_1 + n')$. Let $\mathcal{F}'_0 \xrightarrow{h_1} \ldots \xrightarrow{h_p} \mathcal{F}'_p$ be a derivation witnessing it, $(f_{m+1}, \ldots, f_{m+p}) = (h_1, \ldots, h_p)$, and $\mathcal{F}_{m+1} = \mathcal{F}_m \cup \mathsf{conc}(f_{m+1})$,..., $\mathcal{F}_{m+p} = \mathcal{F}_{m+p-1} \cup \mathsf{conc}(f_{m+p})$. It is easy to check that for every $m + 1 \leqslant j \leqslant m + p$, $\mathsf{prem}(f_j) \subseteq \mathcal{F}_{j-1}$ and that $r_{i_1\jmath}(a, b, n_1)$ and $C_\jmath(b, n_1 + n')$ are in $\mathcal{F}_{m+p}$.

- Let $f_{m+p+1} = (\{r_{i_1\jmath}(a, b, n_1)\}, \{r_{i_1\jmath}(a, b, n)\})$ be the application of the corresponding rule of form (3.7) and $\mathcal{F}_{m+p+1} = \mathcal{F}_{m+p} \cup \mathsf{conc}(f_{m+p+1})$, so that $r_{i_1\jmath}(a, b, n)$ and $C_\jmath(b, n)$ are in $\mathcal{F}_{m+p+1}$.

- Finally, since $i_1\jmath \in \mathcal{J}$ and $\jmath \in \mathcal{J}$, by (24^*), $\exists r_{i_1\jmath}.C_\jmath \sqsubseteq C_{i_1\jmath} \in \mathcal{T}_G$. Let $f_{m+p+2} = (\{r_{i_1\jmath}(a, b, n), C_\jmath(b, n), \exists r_{i_1\jmath}.C_\jmath \sqsubseteq C_{i_1\jmath}\}, \{C_{i_1\jmath}(a, n)\})$ be the application of the corresponding rule of form (3.10) and $\mathcal{F}_{m+p+2} = \mathcal{F}_{m+p+1} \cup \mathsf{conc}(f_{m+p+2})$.

We obtain a derivation $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_{m+p+2}} \mathcal{F}_{m+p+2}$ witnessing $(\mathcal{T}_G, \{A(a, 0)\}) \vdash C_{i_1\jmath}(a, n)$, i.e., $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n C_\iota$. □

Lemma 3.14. *$\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B_i$ if and only if $G \vdash \mathcal{B}_i(c^n)$, for $i \in \{1, \ldots, m\}$.*

Proof. ($\Rightarrow$) We show by induction on p that for every $i \in \{1, \ldots, m\}$, if there exists a derivation of size p witnessing $(\mathcal{T}_G, \{A(a, 0)\}) \vdash B_i(a, n)$, then $G \vdash \mathcal{B}_i(c^n)$.

Since $A \neq B_i$ by construction of $\mathcal{T}_G$, the base case is $p = 1$. The only possible rule application that produces $B_i(a, n)$ using formulas from $\mathcal{T}_G \cup \{A(a, 0)\}$ is f_1 with $\mathsf{prem}(f_1) = \{A(a, 0), A \sqsubseteq \bigcirc^n B_i\}$. By (3.19*) or (3.20*) (depending on whether $n = 0$ or not), it follows that $\mathcal{B}_i \to c^n$ is a rule of G. Hence, $G \vdash \mathcal{B}_i(c^n)$.

Induction step: assume that the property holds for $p - 1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}_p$ be a derivation witnessing $(\mathcal{T}_G, \{A(a, 0)\}) \vdash B_i(a, n)$. If $B_i(a, n) \in \mathcal{F}_{p-1}$, the result follows by the induction hypothesis. Otherwise, there are three possible cases for the last rule application that produces $B_i(a, n)$.

- $\mathsf{prem}(f_p) = \{A(a, 0), A \sqsubseteq \bigcirc^n B_i\}$, and we conclude as in the base case that $G \vdash \mathcal{B}_i(c^n)$.
- $\mathsf{prem}(f_p) = \{C_{\iota(\alpha_1)}(a, n), C_{\iota(\alpha_1)} \sqsubseteq B_i\}$, and by (3.21*), $\mathcal{B}_i \to \alpha_1$ is a rule of G. Since $C_{\iota(\alpha_1)}(a, n) \in \mathcal{F}_{p-1}$, $(\mathcal{T}_G, \{A(a, 0)\}) \vdash C_{\iota(\alpha_1)}(a, n)$. Let $\alpha_1 = \mathcal{B}_{i_1} \ldots \mathcal{B}_{i_k}$. By Lemma B.9, $n = n_1 + \cdots + n_k$, and for $1 \leqslant j \leqslant k$, $(\mathcal{T}, \{A(a, 0)\} \vdash B_{i_j}(a, n_j)$ and there is a derivation of length at most $p - 2$ witnessing it. Hence, by the induction hypothesis, we conclude that $G \vdash \mathcal{B}_{i_j}(c^{n_j})$. Then, since $\mathcal{B}_i \to \mathcal{B}_{i_1} \ldots \mathcal{B}_{i_k}$ is a rule of G, we obtain $G \vdash \mathcal{B}_i(c^n)$.
- $\mathsf{prem}(f_p) = \{C_{\iota(\alpha_1)}(a, n), C_{\iota(\alpha_2)}(a, n), C_{\iota(\alpha_1)} \sqcap C_{\iota(\alpha_2)} \sqsubseteq B_i\}$, and by (3.22*), $\mathcal{B}_i \to \alpha_1 \& \alpha_2$ is a rule of G. The argument is analogous to the one above: we consider separately $\alpha_1 = \mathcal{B}_{i_1} \ldots \mathcal{B}_{i_k}$ and $\alpha_2 = \mathcal{B}'_{i'_1} \ldots \mathcal{B}'_{i'_{k'}}$ to obtain $n = n_1 + \cdots + n_k$ and $n = n'_1 + \cdots + n'_{k'}$ such that $G \vdash \mathcal{B}_{i_j}(c^{n_j})$ for every $1 \leqslant j \leqslant k$ and $G \vdash \mathcal{B}'_{i'_j}(c^{n'_j})$ for every $1 \leqslant j \leqslant k'$, and conclude using the fact that $\mathcal{B}_i \to \mathcal{B}_{i_1} \ldots \mathcal{B}_{i_k} \& \mathcal{B}'_{i'_1} \ldots \mathcal{B}'_{i'_{k'}}$ is a rule of G.

($\Leftarrow$) We show by induction on p that for every $i \in \{1, \ldots, m\}$, if there exists a derivation of size p witnessing $G \vdash \mathcal{B}_i(c^n)$, then $(\mathcal{T}_G, \{A(a, 0)\}) \vdash B_i(a, n)$.

In the base case, $p = 1$, and there are two possible cases for the rule application g_1 that produces $\mathcal{B}_i(c^n)$ from $c(c)$.

- g_1 is the application of a deduction rule that corresponds to $\mathcal{B}_i \to \varepsilon$. In this case, $n = 0$ and by (3.19*), $A \sqsubseteq B_i \in \mathcal{T}_G$.

- g_1 is the application of a deduction rule that corresponds to $\mathcal{B}_i \to c^n$. In this case, by (3.20*), $A \sqsubseteq \bigcirc^n B_i \in \mathcal{T}_G$.

In both cases, $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_i(a,n)$.

Induction step: assume that the property holds for $p-1$ and let $\mathcal{G}_0 \xrightarrow{g_1} \dots \xrightarrow{g_p} \mathcal{G}_p$ be a derivation witnessing $G \vdash \mathcal{B}_i(c^n)$. If $\mathcal{B}_i(c^n) \in \mathcal{G}_{p-1}$, the result follows by the induction hypothesis. Otherwise, there are two cases for g_p that produces $\mathcal{B}_i(c^n)$.

- $\mathsf{prem}(g_p) = \{\mathcal{B}_{i_1}(c^{n_1}), \dots, \mathcal{B}_{i_k}(c^{n_k})\}$ with $n = n_1 + \dots + n_k$ and g_p is the application of a deduction rule that corresponds to $\mathcal{B}_i \to \alpha_1$ with $\alpha_1 = \mathcal{B}_{i_1} \dots \mathcal{B}_{i_k}$. For $1 \leqslant j \leqslant k$, since $\mathcal{B}_{i_j}(c^{n_j}) \in \mathcal{G}_{p-1}$, so that $G \vdash \mathcal{B}_{i_j}(c^{n_j})$ via a derivation of length at most $p-1$, by the induction hypothesis $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_{i_j}(c^{n_j})$. Hence, by Lemma 3.13, $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_{\iota(\alpha_1)}(a,n)$. Since by (3.21*), $C_{\iota(\alpha_1)} \sqsubseteq B_i$, we obtain that $(\mathcal{T}_G, \{A(a,0)\}) \vdash B_i(a,n)$.

- $\mathsf{prem}(g_p) = \{\mathcal{B}_{i_1}(c^{n_1}), \dots, \mathcal{B}_{i_k}(c^{n_k}), \mathcal{B}'_{i'_1}(c^{n'_1}), \dots, \mathcal{B}'_{i'_{k'}}(c^{n'_{k'}})\}$ with $n = n_1 + \dots + n_k$, $n = n'_1 + \dots + n'_{k'}$ and g_p is the application of a deduction rule that corresponds to $\mathcal{B}_i \to \alpha_1 \& \alpha_2$ with $\alpha_1 = \mathcal{B}_{i_1} \dots \mathcal{B}_{i_k}$ and $\alpha_2 = \mathcal{B}'_{i'_1} \dots \mathcal{B}'_{i'_{k'}}$. As above, we obtain $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_{\iota(\alpha_1)}(a,n)$ and $(\mathcal{T}_G, \{A(a,0)\}) \vdash C_{\iota(\alpha_2)}(a,n)$, and conclude using the fact that by (3.22*), $C_{\iota(\alpha_1)} \sqcap C_{\iota(\alpha_1)} \sqsubseteq B_i$. □

Theorem 3.10 then follows from Definition 3.11 and Lemma 3.14.

Theorem 3.10 (Grammars to TBoxes)**.** *For every unary conjunctive grammar $G = (N, \{c\}, R)$, one can construct in polynomial time a $\mathcal{TEL}^{\bigcirc}_{future}$ TBox $\mathcal{T}_G$ and $A \in \mathsf{N_C}(\mathcal{T}_G)$, such that for every $\mathcal{B} \in N$ there is $B \in \mathsf{N_C}(\mathcal{T}_G)$ such that $\mathcal{T}_G \models A \sqsubseteq \bigcirc^n B$ iff $c^n \in L_G(\mathcal{B})$.*

B.5 Proofs for Section 3.5

Recall that in this section, we consider a $\mathcal{TEL}^{\circ}_{lin}$-TBox $\mathcal{T}$. We start with the lemmas.

Lemma 3.17. *If* $\mathsf{N_R}(\mathcal{T}) \subseteq \mathsf{N_R^{rig}}$, *for any* $A, B \in \mathsf{N_C}(\mathcal{T})$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ *iff there exists* $w \in L_{\Gamma_\mathcal{T}}(\mathcal{N}_{AB})$ *with* $\#c(w) - \#d(w) = n$.

Proof. The proof is similar to that of Theorem 3.6, but this time we have to care about two directions in time and two symbols.

($\Rightarrow$) We show by induction on m that for every $A, B \in \mathsf{N_C}(\mathcal{T})$, $a \in \mathsf{N_I} \cup \mathsf{N_N}$, and $n \in \mathbb{Z}$, if there exists a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$ of length at most m, then there is a word $w \in \{c,d\}^*$, such that $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AB}(w)$ and $\#c(w) - \#d(w) = n$.

In the base case, $m = 0$, the derivation consists only of $\mathcal{F}_0 = \mathcal{T} \cup \{A(a,0)\}$, so $B(a,n) \in \mathcal{F}_0$ implies that $A = B$ and $n = 0$, and by (3.14), $\mathcal{N}_{AA} \to \varepsilon$ is a rule of $\Gamma_\mathcal{T}$ so $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AB}(\varepsilon)$. We get $n = 0 = \#c(\varepsilon) - \#d(\varepsilon)$.

Induction step: assume that the property holds for $m-1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$. If $B(a,n) \in \mathcal{F}_{m-1}$, the result follows by induction hypothesis. Otherwise, there are two possible cases for the last rule application f_m that produces $B(a,n)$:

- $\mathsf{prem}(f_m) = \{A'(a, n-k), A' \sqsubseteq \bigcirc^k B\}$ for some $k \in \mathbb{Z}$ is the application of a rule of form (3.8). Since $(\mathcal{F}_0, \ldots, \mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash A'(a, n-k)$, by induction hypothesis, there is $u \in \{c,d\}^*$ such that $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AA'}(u)$, and $\#c(u) - \#d(u) = n - k$. Moreover, $A' \sqsubseteq \bigcirc^k B \in \mathcal{T}$, and, depending on k, we have the following two cases:
  - if $k \geqslant 0$, then by (3.14) or (3.15), $\mathcal{N}_{A'B} \to c^k$ is a rule of $\Gamma_\mathcal{T}$, and thus $\Gamma_\mathcal{T} \vdash \mathcal{N}_{A'B}(v)$ for $v = c^k$;
  - if $k < 0$, then by (3.15*), $\mathcal{N}_{A'B} \to d^{|k|}$ is a rule of $\Gamma_\mathcal{T}$, and thus $\Gamma_\mathcal{T} \vdash \mathcal{N}_{A'B}(v)$ for $v = d^{|k|}$;

Then, since by (3.18) $\mathcal{N}_{AB} \to \mathcal{N}_{AA'}\mathcal{N}_{A'B}$ is a rule of $\Gamma_\mathcal{T}$, we obtain $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AB}(uv)$, and $w = uv$ is such that $\#c(w) - \#d(w) = n$. Indeed,

- if $v = c^k$, then $\#c(w) = \#c(u) + k$ and $\#d(w) = \#d(u)$ so $\#c(w) - \#d(w) = \#c(u) + k - \#d(u) = n - k + k = n$, and
- if $v = d^{|k|}$, $\#c(w) = \#c(u)$ and $\#d(w) = \#d(u) + |k|$ so $\#c(w) - \#d(w) = \#c(u) - \#d(u) - |k| = n - k - |k| = n$ since in this case, $k < 0$.

- $\mathsf{prem}(f_m) = \{r(a,b,n), A'(b,n), \exists r.A' \sqsubseteq B\}$ is the application of a rule of form (3.10). By the form of the derivation rules, $r(a,b,n) \in \mathcal{F}_{m-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < m-1$ such that for some A'', B' and $k \in \mathbb{Z}$, $\mathsf{prem}(f_i) = \{A'' \sqsubseteq \exists r.B', A''(a,k)\}$, $\mathsf{conc}(f_i) = \{r(a,b,k), B'(b,k)\}$. Thus $A''(a,k) \in \mathcal{F}_i$, and $(\mathcal{F}_0, \ldots, \mathcal{F}_i)$ is a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash A''(a,k)$. By the induction hypothesis, we obtain a word u, $\#c(u) - \#d(u) = k$, such that $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AA''}(u)$. Moreover, one can extract from $(\mathcal{F}_i, \ldots, \mathcal{F}_{m-1})$ a derivation of length at most $m-1$ witnessing $(\mathcal{T}, \{B'(b,k)\}) \vdash A'(b,n)$. We get a derivation witnessing $(\mathcal{T}, \{B'(b,0)\}) \vdash A'(b, n-k)$ by shifting all timestamps in this derivation, and thus, by the induction hypothesis, a word v, $\#c(v) - \#d(v) = n - k$, $\Gamma_\mathcal{T} \vdash \mathcal{N}_{B'A'}(v)$. Since $A'' \sqsubseteq \exists r.B'$ and $\exists r.A' \sqsubseteq B$ are in $\mathcal{T}$, by (3.17), $\mathcal{N}_{A''B} \to \mathcal{N}_{B'A'}$ is a rule of $\Gamma_\mathcal{T}$, so from $\Gamma_\mathcal{T} \vdash \mathcal{N}_{B'A'}(v)$, we get $\Gamma_\mathcal{T} \vdash \mathcal{N}_{A''B}(v)$. We combine this with $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AA''}(u)$ and the fact that by (3.18), $\mathcal{N}_{AB} \to \mathcal{N}_{AA''}\mathcal{N}_{A''B}$ is a rule of $\Gamma_\mathcal{T}$ to establish $\Gamma_\mathcal{T} \vdash \mathcal{N}_{AB}(uv)$. Then $w = uv$ is such that $\#c(w) - \#d(w) = \#c(u) - \#d(u) + \#c(v) - \#d(v) = k + n - k = n$.

($\Leftarrow$) We show by induction on m that for every $A, B \in \mathsf{N_C}(\mathcal{T})$, $a \in \mathsf{N_I} \cup \mathsf{N_N}$, and $n \in \mathbb{Z}$, if there exists a word $w \in \{c,d\}^*$ such that

$\#c(w) - \#d(w) = n$, and a derivation witnessing $\Gamma_{\mathcal{T}} \vdash \mathcal{N}_{AB}(w)$ of length at most m, then $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$.

Since $\mathcal{G}_0 = \{c(c)\}$, the base case is $m = 1$, when the derivation consists of $\mathcal{G}_0 \xrightarrow{g_1} \mathcal{G}_1$. There are two possible cases for the rule application g_1 that produces $\mathcal{N}_{AB}(w)$ from $c(c)$:

- g_1 is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \varepsilon$. In this case, $n = 0$ and by (3.14), $A \sqsubseteq B \in \mathcal{T}$ or $A = B$. In both cases, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,0)$, i.e. $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$.

- g_1 is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to c^n$ $(n \geqslant 0)$ or to $\mathcal{N}_{AB} \to d^{|n|}$ $(n < 0)$. In this case, by (3.15) or (3.15*), respectively, $A \sqsubseteq \bigcirc^n B \in \mathcal{T}$. Hence, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$.

Induction step: assume that the property holds for $m-1$ and let $\mathcal{G}_0 \xrightarrow{g_1} \ldots \xrightarrow{g_m} \mathcal{G}_m$ be a derivation witnessing $\Gamma_{\mathcal{T}} \vdash \mathcal{N}_{AB}(w)$, $\#c(w) - \#d(w) = n$. If $\mathcal{N}_{AB}(w) \in \mathcal{G}_{m-1}$, the result follows by induction hypothesis. Otherwise, there are two possible cases for the last rule application g_m that produces $\mathcal{N}_{AB}(w)$:

- $\mathsf{prem}(g_m) = \{\mathcal{N}_{CD}(w)\}$, and g_m is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \mathcal{N}_{CD}$. By (3.17), there exists r such that $A \sqsubseteq \exists r.C$ and $\exists r.D \sqsubseteq B$ are in $\mathcal{T}$. Then $(\mathcal{G}_0, \ldots, \mathcal{G}_{m-1})$ is a derivation witnessing $G \vdash \mathcal{N}_{CD}(w)$ so by the induction hypothesis, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}, \{C(a,0)\}) \vdash D(a,n)$. Let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}_p$ be obtained from that derivation by substituting a everywhere with a $b \in \mathsf{N_N}$ not appearing in the derivation. Then the following is a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$:

$$\mathcal{F} \xrightarrow{f_0} \mathcal{F}'_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}'_p \xrightarrow{f_{p+1}} \mathcal{F}_{p+1} \xrightarrow{f_{p+2}} \mathcal{F}_{p+2}$$

  where

  - $\mathcal{F} = \{A(a,0)\} \cup \mathcal{T}$;

- f_0 is an application of the rule of form (3.11) with $\mathsf{prem}(f_0) = \{A(a,0), A \sqsubseteq \exists r.C\}$ and $\mathsf{conc}(f_0) = \{r(a,b,0), C(b,0)\}$;
- $\mathcal{F}'_i = \mathcal{F}_i \cup \{A(a,0), r(a,b,0)\}$ for $0 \leqslant i \leqslant p$;
- f_{p+1} is an application of the rule of form (3.7) with $\mathsf{prem}(f_{p+1}) = \{r(a,b,0)\}$ and $\mathsf{conc}(f_{p+1}) = \{r(a,b,n)\}$;
- f_{p+2} is an application of the rule of form (3.10) with $\mathsf{prem}(f_{p+2}) = \{r(a,b,n), D(b,n), \exists r.D \sqsubseteq B\}$ and $\mathsf{conc}(f_{p+2}) = \{B(a,n)\}$;
- $\mathcal{F}_{p+1} = \mathcal{F}'_p \cup \mathsf{conc}(f_{p+1})$ and $\mathcal{F}_{p+2} = \mathcal{F}_{p+1} \cup \mathsf{conc}(f_{p+2})$.

- $\mathsf{prem}(g_m) = \{\mathcal{N}_{AC}(u), \mathcal{N}_{CB}(v)\}$, where $\#c(u) - \#d(u) = n-k$ and $\#c(v) - \#d(v) = k$, for some $k \in \mathbb{Z}$, and g_m is the application of a deduction rule that corresponds to $\mathcal{N}_{AB} \to \mathcal{N}_{AC}\mathcal{N}_{CB}$. By (3.18) $A, B, C \in \mathsf{N_C}(\mathcal{T})$. Then $(\mathcal{G}_0, \ldots, \mathcal{G}_{m-1})$ is a derivation witnessing $G \vdash \mathcal{N}_{AC}(u)$ and $G \vdash \mathcal{N}_{CB}(v)$. By the induction hypothesis, for every $a \in \mathsf{N_I} \cup \mathsf{N_N}$, $(\mathcal{T}, \{A(a,0)\}) \vdash C(a, n-k)$ and $(\mathcal{T}, \{C(a,0)\}) \vdash B(a,k)$, so (by Proposition 3.3), $(\mathcal{T}, \{C(a,n-k)\}) \vdash B(a,n)$. Hence, $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$. □

Lemma 3.20. *Let $\mathcal{T}$ be a $\mathcal{TEL}^{\circ}_{lin}$ TBox. For any $A, B \in \mathsf{N_C}(\mathcal{T})$ and $n \in \mathbb{Z}$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ if and only if $\mathcal{T}^{lin}_{rig} \models A \sqsubseteq \bigcirc^n B$.*

Proof. ($\Rightarrow$) Let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$. We build a derivation witnessing $(\mathcal{T}^{lin}_{rig}, \{A(a,0)\}) \vdash B(a,n)$. Let $(h_1, \ldots, h_p)$ be the sequence of rule applications obtained from $(f_1, \ldots, f_m)$ as follows. For every application f_i of a rule of the form (3.10), where $f_i = (\{r(a,b,\ell), A'(b,\ell), \exists r.A' \sqsubseteq B'\}, \{B'(a,\ell)\})$ with $r \in \mathsf{N^{loc}_R}$, find the application f_j, $j < i$, of a rule of the form (3.11) that produced $r(a,b,\ell)$: $f_j = (\{A''(a,\ell), A'' \sqsubseteq \exists r.B''\}, \{r(a,b,\ell), B''(b,\ell)\})$. From $(\mathcal{F}_{j-1}, \ldots, \mathcal{F}_i)$, by Proposition 3.3, we get $\mathcal{T} \models A'' \sqsubseteq B'$. Hence, $A'' \sqsubseteq B' \in \mathcal{T}^{lin}_{rig}$. Substitute the sequence $(f_j, \ldots, f_i)$ with a single

application h of a rule of the form (3.8), where $h = (\{A''(a,\ell), A'' \sqsubseteq B'\}, \{B'(a,\ell)\})$.

Since all concept inclusions used in the premises of $h_1, \ldots, h_p$ belongs to $\mathcal{T}_{rig}^{lin}$ by construction, we indeed obtain a derivation $\mathcal{F}'_0 \xrightarrow{h_1} \ldots \xrightarrow{h_p} \mathcal{F}'_p$ witnessing $(\mathcal{T}_{rig}^{lin}, \{A(a,0)\}) \vdash B(a,n)$ by setting $\mathcal{F}'_0 = \{A(a,0)\} \cup \mathcal{T}_{rig}^{lin}$ and $\mathcal{F}'_i = \mathcal{F}'_{i-1} \cup \mathsf{conc}(h_i)$.

($\Leftarrow$) Suppose $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ is a derivation witnessing $(\mathcal{T}_{rig}^{lin}, \{A(a,0)\}) \vdash B(a,n)$. Let $\mathcal{T}_1 = \mathcal{T}_{rig}^{lin} \setminus \mathcal{T}$. Since $\mathcal{T}_{rig}^{lin} = \mathcal{T}_0 \cup \{A' \sqsubseteq B' \mid \mathcal{T} \models A' \sqsubseteq B'\}$, where $\mathcal{T}_0$ is obtained from $\mathcal{T}$ by removing concept inclusions that use local role names, $\mathcal{T}_1$ contains only concept inclusions of the form $A' \sqsubseteq B'$.

We build a derivation witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$. Let $(h_1, \ldots, h_p)$ be the sequence of rule applications obtained from $(f_1, \ldots, f_m)$ as follows. For each $f_i = (\{A'(b,\ell), A' \sqsubseteq B'\}, \{B'(b,\ell)\})$ such that $A' \sqsubseteq B' \in \mathcal{T}_1$, apply Proposition 3.3 to obtain a sequence $(f'_1, \ldots, f'_k)$ of rule applications corresponding to a derivation witnessing $(\mathcal{T}, \{A'(b,\ell)\}) \vdash B'(b,\ell)$. Then, substitute f_i with $(f'_1, \ldots, f'_k)$.

We can check that all concept inclusions used in the premises of $h_1, \ldots, h_p$ belong to $\mathcal{T}$, so that we indeed obtain a derivation $\mathcal{F}'_0 \xrightarrow{h_1} \ldots \xrightarrow{h_p} \mathcal{F}'_p$ witnessing $(\mathcal{T}, \{A(a,0)\}) \vdash B(a,n)$ by setting $\mathcal{F}'_0 = \{A(a,0)\} \cup \mathcal{T}$ and $\mathcal{F}'_i = \mathcal{F}'_{i-1} \cup \mathsf{conc}(h_i)$. □

Theorem 3.15. *For every $\mathcal{TEL}^{\circ}_{lin}$ TBox $\mathcal{T}$, there exists a context-free grammar $\Gamma_{\mathcal{T}} = (N, \{c,d\}, R)$, of size polynomial in $|\mathcal{T}|$, such that for any $A, B \in \mathsf{N_C}(\mathcal{T})$, there is $\mathcal{N}_{AB} \in N$ such that $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ iff there exists $w \in L_{\Gamma_{\mathcal{T}}}(\mathcal{N}_{AB})$ with $\#c(w) - \#d(w) = n$.*

Proof. Using Lemma 3.20 and the facts that $\mathsf{N_R}(\mathcal{T}_{rig}^{lin}) \subseteq \mathsf{N_R^{rig}}$ and that $|\mathcal{T}_{rig}^{lin}|$ is polynomial in $|\mathcal{T}|$, we can assume that $\mathsf{N_R}(\mathcal{T}) \subseteq \mathsf{N_R^{rig}}$. Let $\Gamma_{\mathcal{T}}$ be the grammar given in Definition 3.16. It is easy to check that the size of $\Gamma_{\mathcal{T}}$ is polynomial in $|\mathcal{T}|$, and by Lemma 3.17, for any $A, B \in \mathsf{N_C}(\mathcal{T})$, $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ iff there exists $w \in L_{\Gamma_{\mathcal{T}}}(\mathcal{N}_{AB})$ with $\#c(w) - \#d(w) = n$. □

B.6 Proofs for Section 3.6

B.6.1 Proof of Theorem 3.23

Theorem 3.23. *Query answering relative to $\mathcal{TEL}^{\circ}_{future}$ TBoxes is* P*-complete, both for combined and data complexity.*

Proof. The lower bounds hold already for the description logic $\mathcal{EL}$ (without temporal operators) [Calvanese et al., 2013]. For the upper bounds, we provide a polynomial reduction from the problem of deciding whether $(\mathcal{T}, \mathcal{D}) \models A(a, n)$ to that of checking whether a word belongs to the language of a conjunctive grammar, which can be tested in polynomial time (Theorem 2.3). Our reduction builds a $\mathcal{TEL}^{\circ}_{future}$-TBox $\mathcal{T}' \cup \mathcal{T}_{\mathcal{D}}$, an assertion $C_a(a, l)$ and a concept name A_n such that $(\mathcal{T}, \mathcal{D}) \models A(a, n)$ iff $(\mathcal{T}' \cup \mathcal{T}_{\mathcal{D}}, \{C_a(a, l)\}) \models A_n(a, n)$, then use Proposition 3.3 and Theorem 3.6 to conclude. The idea is to encode all information about a in $\mathcal{D}$ into the single fact $C_a(a, l)$ thanks to $\mathcal{T}_{\mathcal{D}}$.

Let $\mathrm{obj}(\mathcal{D})$ be the set of individual names that occur in $\mathcal{D}$, and $l, m \in \mathbb{Z}$ be the least and the greatest timestamps appearing in $\mathcal{D}$. We introduce concept names $\{C_a \mid a \in \mathrm{obj}(\mathcal{D})\}$ and $\{A_k \mid A \in \mathsf{N_C}(\mathcal{T}), l \leqslant k \leqslant m + 1\}$, and role names $\{\rho^r_{ab} \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}} \mid a, b \in \mathrm{obj}(\mathcal{D}), r \in \mathsf{N_R}(\mathcal{T})\}$. For the convenience of notation, we write A_k for all $k \geqslant l$, assuming that $A_k = A_{m+1}$ when $k > m$.

Let $\mathcal{T}'$ be the TBox containing the following concept inclusions for all $l \leqslant k \leqslant m + 1$.

$$A_k \sqsubseteq \bigcirc^s B_{k+s} \qquad \text{for } A \sqsubseteq \bigcirc^s B \in \mathcal{T} \tag{3.27}$$

$$A_k \sqcap A'_k \sqsubseteq B_k \qquad \text{for } A \sqcap A' \sqsubseteq B \in \mathcal{T} \tag{3.28}$$

$$A_k \sqsubseteq \exists r.B_k \qquad \text{for } A \sqsubseteq \exists r.B \in \mathcal{T} \tag{3.29}$$

$$\exists r.A_k \sqsubseteq B_k \qquad \text{for } \exists r.A \sqsubseteq B \in \mathcal{T} \tag{3.30}$$

Additionally, define a TBox $\mathcal{T}_{\mathcal{D}}$ to contain the following concept inclusions.

$$C_a \sqsubseteq \bigcirc^{k-l} A_k \qquad \text{for } A(a, k) \in \mathcal{D} \tag{3.31}$$

$$C_a \sqsubseteq \exists \rho^r_{ab} \,.\, C_b \qquad \text{for } r(a,b,\ell) \in \mathcal{D} \tag{3.32}$$

$$\exists \rho^r_{ab} \,.\, A_k \sqsubseteq B_k \qquad \text{for } \exists r.A \sqsubseteq B \in \mathcal{T}, r(a,b,\ell) \in \mathcal{D}, \text{ where } r \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}} \text{ or } \ell = k \tag{3.33}$$

Clearly, both $\mathcal{T}'$ and $\mathcal{T}_\mathcal{D}$ can be constructed in polynomial time w.r.t. $|\mathcal{T}| + |\mathcal{D}|$ and are expressed in $\mathcal{TEL}^\circ_{future}$ (since $\mathcal{T}$ is a $\mathcal{TEL}^\circ_{future}$-TBox and $k - l \geqslant 0$ for every $A(a,k) \in \mathcal{D}$ by definition of l).

Lemma B.10. *For all $A, B \in \mathsf{N}_\mathsf{C}(\mathcal{T})$ and $a \in \mathsf{N}_\mathsf{I}$:*

(i) for all $n \in \mathbb{N}$ and $l \leqslant k, \ell \leqslant m+1$,

$$(\mathcal{T}', \{A_k(a,0)\}) \models B_\ell(a,n) \text{ implies that } \ell = k+n$$

(or $\ell = m+1$ and $k+n > m$);

(ii) for all $s, n, k \in \mathbb{N}$,

$$(\mathcal{T}, \{A(a,s)\}) \models B(a,n) \text{ iff } (\mathcal{T}', \{A_{k+s}(a,s)\}) \models B_{k+n}(a,n)$$

(note that if $n < s$ the equivalence holds trivially since $\mathcal{T}'$ is a $\mathcal{TEL}^\circ_{future}$ TBox, and recall that if $k+n > m$, $B_{k+n} = B_{m+1}$, and that if $k+s > m$, $A_{k+s} = A_{m+1}$).

Proof. For point *(i)*, we show by induction on p that for all $A, B \in \mathsf{N}_\mathsf{C}(\mathcal{T})$, $a \in \mathsf{N}_\mathsf{I} \cup \mathsf{N}_\mathsf{N}$, $n \in \mathbb{N}$, and $l \leqslant k, \ell \leqslant m+1$, if there is a derivation witnessing $(\mathcal{T}', \{A_k(a,0)\}) \models B_\ell(a,n')$ of size p, then either $\ell = k+n$, or $B_\ell = B_{m+1}$ and $k+n > m$.

The base case, $p = 0$, is immediate, since in this case $n = 0$ and $B_\ell = A_k$, so $\ell = k+n$.

Induction step: assume that the property holds for $p-1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}_p$ be a derivation witnessing $(\mathcal{T}', \{A_k(a,0)\}) \models B_\ell(a,n)$. If $B_\ell(a,n) \in \mathcal{F}_{p-1}$, the result follows by induction hypothesis. Otherwise, there are three possible cases for the last rule application f_p that produces $B_\ell(a,n)$:

- $\mathsf{prem}(f_p) = \{C_j \sqsubseteq \bigcirc^{\ell-j} B_\ell, C_j(a, n-\ell+j)\}$, and f_p is the application of a rule of the form (3.8). Since $C_j(a, n-\ell+j) \in \mathcal{F}_{p-1}$, $(\mathcal{T}', \{A_k(a,0)\}) \models C_j(a, n-\ell+j)$ with a derivation of size $p-1$, so by induction hypothesis, either $j = n-\ell+j+k$, or $C_j = C_{m+1}$ and $n-\ell+j+k > m$. In the first case, $\ell = k+n$, and in the second case, since $\ell - j > 0$ as $\mathcal{T}'$ is a $\mathcal{TEL}^{\bigcirc}_{future}$-TBox, $\ell > j$ so $B_\ell = B_{m+1}$ and $k+n > m+\ell-j > m$.

- $\mathsf{prem}(f_p) = \{C_\ell \sqcap C'_\ell \sqsubseteq B_\ell, C_\ell(a,n), C'_\ell(a,n)\}$, and f_p is the application of a rule of the form (3.9). Since $C_\ell(a,n) \in \mathcal{F}_{p-1}$ and $C'_\ell(a,n) \in \mathcal{F}_{p-1}$, $(\mathcal{F}_0, \ldots, \mathcal{F}_{p-1})$ is a derivation witnessing $(\mathcal{T}', \{A_k(a,0)\}) \models C_\ell(a,n)$ and $(\mathcal{T}', \{A_k(a,0)\}) \models C'_\ell(a,n)$. By the induction hypothesis, we obtain that either $\ell = k+n$ or $C_\ell = C_{m+1}$, $C'_\ell = C'_{m+1}$ and $k+n > m$.

- $\mathsf{prem}(f_p) = \{\exists r.C_\ell \sqsubseteq B_\ell, r(a,b,n), C_\ell(b,n)\}$, and f_p is the application of a rule of the form (3.10). By the form of the derivation rules, $r(a,b,n) \in \mathcal{F}_{p-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < p-1$ such that for some A''_j, B'_j and j', $\mathsf{prem}(f_i) = \{A''_j \sqsubseteq \exists r.B'_j, A''_\ell(a,j')\}$, $\mathsf{conc}(f_i) = \{r(a,b,j'), B'_j(b,j')\}$. Thus $A''_j(a,j') \in \mathcal{F}_{i-1}$, and $(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1})$ is a derivation witnessing $(\mathcal{T}', \{A_k(a,0)\}) \models A''_j(a,j')$. By induction hypothesis, it follows that either (1) $j = k+j'$ or (2) $A''_j = A''_{m+1}$ and $k+j' > m$. Moreover, one can extract from $(\mathcal{F}_i, \ldots, \mathcal{F}_{p-1})$ a derivation for $(\mathcal{T}', \{B'_j(b,j')\}) \vdash C_\ell(b,n)$ of size at most $p-1$, from which we obtain a derivation for $(\mathcal{T}', \{B'_j(b,0)\}) \vdash C_\ell(b, n-j')$ by shifting all timestamps. Hence, by induction hypothesis, we have either (i) $\ell = j+n-j'$ or (ii) $C_\ell = C_{m+1}$ and $j+n-j' > m$. Moreover, since $\mathcal{T}'$ is a $\mathcal{TEL}^{\bigcirc}_{future}$-TBox, $n \geqslant j'$.

  - In case (1-i), $\ell = j+n-j' = k+j'+n-j' = k+n$.
  - In case (1-ii), $C_\ell = C_{m+1}$ and $j+n-j' > m$ so $k+j'+n-j' > m$, i.e., $k+n > m$.

– In case (2), $k + j' > m$ so since $n \geqslant j'$, $k + n > m$. Moreover, in this case, $A''_j = A''_{m+1}$ so $B'_j = B'_{m+1}$ and by the form of the concept inclusions in $\mathcal{T}$' (where concept names that occur in the right-hand side have always equal or higher indexes than those from the left-hand side), it must be the case that $C_\ell = C_{m+1}$.

For point (ii), we show the two directions of the equivalence in a similar way.
($\Rightarrow$) We show by induction on p that for all $A, B \in \mathsf{N_C}(\mathcal{T})$, $a \in \mathsf{N_I} \cup \mathsf{N_N}$, and $s, n \in \mathbb{N}$, if there exists a derivation of length p witnessing $(\mathcal{T}, \{A(a, s)\}) \vdash B(a, n)$, then for every k, $(\mathcal{T}', \{A_{k+s}(a, s)\}) \vdash B_{k+n}(a, n)$ (with $m + 1$ instead of $k + s$ and/or $k + n$ if they are larger than m).

The base case, $p = 0$, is immediate since in this case $B = A$ and $s = n = 0$, and it holds that $(\mathcal{T}', \{A_k(a, 0)\}) \vdash A_k(a, 0)$.

Induction step: assume that the property holds for $p - 1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}_p$ be a derivation witnessing $(\mathcal{T}, \{A(a, s)\}) \vdash B(a, n)$. If $B(a, n) \in \mathcal{F}_{p-1}$, the result follows by induction hypothesis. Otherwise, there are three possible cases for the last rule application f_p that produces $B(a, n)$:

- $\mathsf{prem}(f_p) = \{A'(a, n - \ell), A' \sqsubseteq \bigcirc^\ell B\}$ for some $\ell \in \{0, \ldots, n - s\}$ (since $\mathcal{T}$ is a $\mathcal{TEL}^\bigcirc_{future}$-TBox and $(\mathcal{T}, \{A(a, s)\}) \vdash A'(a, n - \ell)$), and f_p is the application of a rule of the form (3.8). Since $(\mathcal{F}_0, \ldots, \mathcal{F}_{p-1})$ is a derivation witnessing $(\mathcal{T}, \{A(a, s)\}) \vdash A'(a, n-\ell)$, by induction hypothesis, for every k, $(\mathcal{T}', \{A_{k+s}(a, s)\}) \vdash A'_{k+n-\ell}(a, n - \ell)$. Since $A' \sqsubseteq \bigcirc^\ell B \in \mathcal{T}$ it follows that $A'_{k+n-\ell} \sqsubseteq \bigcirc^\ell B_{k+n} \in \mathcal{T}'$ (and $A'_j \sqsubseteq \bigcirc^\ell B_{m+1}$ for every j such that $j + \ell \geqslant m + 1$), by (3.27). Hence, $(\mathcal{T}', \{A_{k+s}(a, s)\}) \vdash B_{k+n}(a, n)$.

- $\mathsf{prem}(f_p) = \{A'(a, n), A''(a, n), A' \sqcap A'' \sqsubseteq B\}$ and f_p is the application of a rule of form (3.9). Since $A'(a, n)$ and $A''(a, n)$ are in $\mathcal{F}_{p-1}$, $(\mathcal{F}_0, \ldots, \mathcal{F}_{p-1})$ is a derivation witnessing $(\mathcal{T}, \{A(a, s)\}) \vdash A'(a, n)$ and $(\mathcal{T}, \{A(a, s)\}) \vdash A''(a, n)$.

By the induction hypothesis, for every k, $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A'_{k+n}(a,n)$ and $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A''_{k+n}(a,n)$. Since $A' \sqcap A'' \sqsubseteq B \in \mathcal{T}$, $A'_{k+n} \sqcap A''_{k+n} \sqsubseteq B_{k+n} \in \mathcal{T}'$, by (3.28). Hence, $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash B_{k+n}(a,n)$.

- $\mathsf{prem}(f_p) = \{r(a,b,n), A'(b,n), \exists r.A' \sqsubseteq B\}$ and f_p is the application of a rule of form (3.10). By the form of the derivation rules, $r(a,b,n) \in \mathcal{F}_{p-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < p-1$ such that for some A'', B' and $j \geqslant s$ (again, because $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}_{future}$-TBox), $\mathsf{prem}(f_i) = \{A'' \sqsubseteq \exists r.B', A''(a,j)\}$, $\mathsf{conc}(f_i) = \{r(a,b,j), B'(b,j)\}$. Thus $A''(a,j) \in \mathcal{F}_{i-1}$, and $(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1})$ is a derivation witnessing $(\mathcal{T}, \{A(a,s)\}) \vdash A''(a,j)$. By the induction hypothesis, for every k, there is a derivation $\mathbf{der}_1$ witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A''_{k+j}(a,j)$. Moreover, we extract from $(\mathcal{F}_i, \ldots, \mathcal{F}_{p-1})$ a derivation witnessing $(\mathcal{T}, \{B'(b,j)\}) \vdash A'(b,n)$ (note that $n = j$ if $r \in \mathsf{N}_\mathsf{R}^{\mathsf{loc}}$). By the induction hypothesis, $(\mathcal{T}', \{B'_{k+j}(b,j)\}) \vdash A'_{k+n}(b,n)$. Let $\mathbf{der}_2$ be a derivation witnessing this. We obtain a derivation witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash B_{k+n}(a,n)$ as follows. Since $A'' \sqsubseteq \exists r.B' \in \mathcal{T}$, we have $A''_{k+j} \sqsubseteq \exists r.B'_{k+j} \in \mathcal{T}'$, by (3.29). Further, since $\exists r.A' \sqsubseteq B \in \mathcal{T}$, we have $\exists r.A'_{k+n} \sqsubseteq B_{k+n} \in \mathcal{T}'$, by (3.30). Thus, start from $\mathcal{F}'_0 = \mathcal{T}' \cup \{A_{k+s}(a,s)\}$, proceed as in $\mathbf{der}_1$ until you derive $A''_{k+j}(a,j)$. Apply a rule of the form (3.11) using $A''_{k+j} \sqsubseteq \exists r.B'_{k+j}$ to obtain $r(a,b,j)$ and $B'_{k+j}(b,j)$, and proceed as in $\mathbf{der}_2$ until you have $A'_{k+n}(b,n)$. If $r \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}}$, apply a rule of the form (3.7) to obtain $r(a,b,n)$. Otherwise, it means that $j = n$, so we already have $r(a,b,n)$. Finally, apply a rule of the form (3.10) using $\exists r.A'_{k+n} \sqsubseteq B_{k+n}$ to get $B_{k+n}(a,n)$.

($\Leftarrow$) We show by induction on p that for all $A, B \in \mathsf{N}_\mathsf{C}(\mathcal{T})$, $a \in \mathsf{N}_\mathsf{I} \cup \mathsf{N}_\mathsf{N}$, and $s, n, k \in \mathbb{N}$, if there exists a derivation of length p witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash B_{k+n}(a,n)$ (with $m+1$ instead of $k+s$ and/or $k+n$ if they are larger than m), then $(\mathcal{T}, \{A(a,s)\}) \vdash B(a,n)$.

The base case, $p = 0$, is immediate since in this case $B = A$ and $n = s$, and it holds that $(\mathcal{T}, \{A(a,s)\}) \vdash A(a,s)$.

Induction step: assume that the property holds for $p-1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_p} \mathcal{F}_p$ be a derivation witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash B_{k+n}(a,n)$. If $B_{k+n}(a,n) \in \mathcal{F}_{p-1}$, the result follows by induction hypothesis. Otherwise, there are three possible cases for the last rule application f_p that produces $B_{k+n}(a,n)$:

- $\mathsf{prem}(f_p) = \{A'_{n+k-\ell}(a, n+k-\ell), A'_{n+k-\ell} \sqsubseteq \bigcirc^{\ell} B_{k+n}\}$ for some $\ell \in \{0, \ldots, n+k-s\}$ (since $\mathcal{T}'$ is a $\mathcal{TEL}^{\bigcirc}_{future}$-TBox and $(\mathcal{T}', \{A_{k+s}(a,s)\} \models A'_{n+k-\ell}(a, n+k-\ell))$, and f_p is the application of a rule of the form (3.8). Since $(\mathcal{F}_0, \ldots, \mathcal{F}_{p-1})$ is a derivation witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A'_{n+k-\ell}(a, n+k-\ell)$, by induction hypothesis, $(\mathcal{T}, \{A(a,s)\}) \vdash A'(a, n-\ell)$. Since $A'_{n+k-\ell} \sqsubseteq \bigcirc^{\ell} B_{k+n} \in \mathcal{T}'$ it follows that $A' \sqsubseteq \bigcirc^{\ell} B \in \mathcal{T}$, by (3.27). We conclude that $(\mathcal{T}, \{A(a,s)\}) \vdash B(a,n)$ by applying a rule of the form (3.8).

- $\mathsf{prem}(f_p) = \{A'_{k+n}(a,n), A''_{k+n}(a,n), A'_{k+n} \sqcap A''_{k+n} \sqsubseteq B_{k+n}\}$ and f_p is the application of a rule of form (3.9). Since $A'_{k+n}(a,n)$ and $A''_{k+n}(a,n)$ are in $\mathcal{F}_{p-1}$, $(\mathcal{F}_0, \ldots, \mathcal{F}_{p-1})$ is a derivation witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A'_{k+n}(a,n)$ and $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A''_{k+n}(a,n)$. By the induction hypothesis, $(\mathcal{T}, \{A(a,s)\}) \vdash A'(a,n)$ and $(\mathcal{T}, \{A(a,s)\}) \vdash A''(a,n)$. Since $A'_{k+n} \sqcap A''_{k+n} \sqsubseteq B_{k+n} \in \mathcal{T}'$, $A' \sqcap A'' \sqsubseteq B \in \mathcal{T}$, by (3.28). We conclude that $(\mathcal{T}, \{A(a,s)\}) \vdash B(a,n)$ by applying a rule of the form (3.9).

- $\mathsf{prem}(f_p) = \{r(a,b,n), A'_{k+n}(b,n), \exists r.A'_{k+n} \sqsubseteq B_{k+n}\}$ and f_p is the application of a rule of form (3.10). By the form of the derivation rules, $r(a,b,n) \in \mathcal{F}_{p-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < p-1$ such that for some A''_{ℓ}, B'_{ℓ}, $\mathsf{prem}(f_i) = \{A''_{\ell} \sqsubseteq \exists r.B'_{\ell}, A''_{\ell}(a,\ell')\}$, $\mathsf{conc}(f_i) = \{r(a,b,\ell'), B'_{\ell}(b,\ell')\}$ (note that $\ell' = n$ if $r \in \mathsf{N}_{\mathsf{R}}^{\mathsf{loc}}$). Thus $A''_{\ell}(a,\ell') \in \mathcal{F}_{i-1}$, and $(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1})$ is a derivation witnessing $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash$

$A''_\ell(a,\ell')$. Since $(\mathcal{T}', \{A_{k+s}(a,0)\}) \vdash A''_\ell(a,\ell'-s)$, by point *(i)* of the lemma, $\ell = k+s+\ell'-s$, so $\ell = k+\ell'$. Hence, since $(\mathcal{T}', \{A_{k+s}(a,s)\}) \vdash A''_{k+\ell'}(a,\ell')$, by the induction hypothesis, there is a derivation $\mathbf{der}_1$ witnessing $(\mathcal{T}, \{A(a,s)\}) \vdash A''(a,\ell')$. Moreover, we extract from $(\mathcal{F}_i, \ldots, \mathcal{F}_{p-1})$ a derivation witnessing $(\mathcal{T}', \{B'_{k+\ell'}(b,\ell')\}) \vdash A'_{k+n}(b,n)$. By induction hypothesis, $(\mathcal{T}, \{B'(b,\ell')\}) \vdash A'(b,n)$. Let $\mathbf{der}_2$ be a derivation witnessing it. We obtain a derivation witnessing $(\mathcal{T}, \{A(a,s)\}) \vdash B(a,n)$ as follows. Since $A''_\ell \sqsubseteq \exists r.B'_\ell \in \mathcal{T}'$, we have $A'' \sqsubseteq \exists r.B' \in \mathcal{T}$, by (3.29). Further, since $\exists r.A'_{k+n} \sqsubseteq B_{k+n} \in \mathcal{T}'$, we have $\exists r.A' \sqsubseteq B \in \mathcal{T}$, by (3.30). Thus, start from $\mathcal{F}'_0 = \mathcal{T} \cup \{A(a,s)\}$, proceed as in $\mathbf{der}_1$ until you derive $A''(a,\ell')$. Apply a rule of the form (3.11) to obtain $r(a,b,\ell')$ and $B'(b,\ell')$, and proceed as in $\mathbf{der}_2$ until you have $A'(b,n)$. If $r \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}}$, apply a rule of the form (3.7) to obtain $r(a,b,n)$. Otherwise, it means that $\ell' = n$, so we already have $r(a,b,n)$. Finally, apply a rule of the form (3.10) to get $B(a,n)$. □

Lemma B.11. *For all $A \in \mathsf{N}_\mathsf{C}(\mathcal{T})$, $a \in \mathsf{N}_\mathsf{I}$ and $n \in \mathbb{N}$,*

$$(\mathcal{T},\mathcal{D}) \models A(a,n) \text{ iff } (\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \models A_n(a,n)$$

(recall that if $n > m+1$, $A_n = A_{m+1}$, and that if $n < l$, $(\mathcal{T},\mathcal{D}) \not\models A(a,n)$ and $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \not\models A_n(a,n)$ since both $\mathcal{T}$ and $\mathcal{T}' \cup \mathcal{T}_\mathcal{D}$ are $\mathcal{TEL}^\circ_{future}$-TBoxes).

Proof. ($\Rightarrow$) We show by induction on m that for all $A \in \mathsf{N}_\mathsf{C}(\mathcal{T})$, $a \in \mathsf{N}_\mathsf{I}$ and $n \in \mathbb{N}$, if there exists a derivation of length m witnessing $(\mathcal{T},\mathcal{D}) \vdash A(a,n)$, then $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A_n(a,n)$.

In the base case, $m = 0$, the derivation consists only of $\mathcal{F}_0 = \mathcal{T} \cup \mathcal{D}$, so $A(a,n) \in \mathcal{D}$. Hence, by (3.31), $C_a \sqsubseteq \bigcirc^{n-l} A_n \in \mathcal{T}_A$. It follows that $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A_n(a,n)$.

Induction step: assume that the property holds for $m-1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing $(\mathcal{T},\mathcal{D}) \vdash A(a,n)$. If $A(a,n) \in \mathcal{F}_{m-1}$, the result follows by induction hypothesis. Other-

wise, there are three possible cases for the last rule application f_m that produces $A(a,n)$:

- $\mathsf{prem}(f_m) = \{A'(a,n-k), A' \sqsubseteq \bigcirc^k A\}$ for some $k \in \{0,\dots,n-l\}$ (since $n-k$ cannot be less than l, given that $\mathcal{T}$ is a $\mathcal{TEL}^{\bigcirc}_{future}$-TBox and $(\mathcal{T},\mathcal{D}) \vdash A'(a,n-k)$), and f_m is the application of a rule of the form (3.8). Since $(\mathcal{F}_0,\dots,\mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T},\mathcal{D}) \vdash A'(a,n-k)$, by induction hypothesis, $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A'_{n-k}(a,n-k)$. Since $A' \sqsubseteq \bigcirc^k A \in \mathcal{T}$ it follows that $A'_{n-k} \sqsubseteq \bigcirc^k A_n \in \mathcal{T}'$, by (3.27). Hence, $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A_n(a,n)$.

- $\mathsf{prem}(f_m) = \{A'(a,n), A''(a,n), A' \sqcap A'' \sqsubseteq A\}$ and f_m is the application of a rule of form (3.9). Since $A'(a,n)$ and $A''(a,n)$ are in $\mathcal{F}_{m-1}$, $(\mathcal{F}_0,\dots,\mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T},\mathcal{D}) \vdash A'(a,n)$ and $(\mathcal{T},\mathcal{D}) \vdash A''(a,n)$. By the induction hypothesis, $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A'_n(a,n)$ and $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A''_n(a,n)$. Since $A' \sqcap A'' \sqsubseteq A \in \mathcal{T}$, $A'_n \sqcap A''_n \sqsubseteq A_n \in \mathcal{T}'$, by (3.28). Hence, $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A_n(a,n)$.

- $\mathsf{prem}(f_m) = \{r(a,b,n), A'(b,n), \exists r.A' \sqsubseteq A\}$ and f_m is the application of a rule of form (3.10). Then we have two subcases:

  - $b \in \mathrm{obj}(\mathcal{D})$. Since $A'(b,n) \in \mathcal{F}_{m-1}$, there is a derivation of length $m-1$ witnessing $(\mathcal{T},\mathcal{D}) \vdash A'(b,n)$. By the induction hypothesis, there is a derivation **der** witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_b(b,l)\}) \vdash A'_n(b,n)$. Furthermore, since the derivation rules can only add a fact of the form $r(a,b,\ell)$ with $a,b \in \mathsf{N_I}$ if r is rigid (cf. (3.7)–(3.11)), there exists $r(a,b,\ell) \in \mathcal{D}$ such that either $r \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}}$ or $\ell = n$. In both cases, by (3.32), $C_a \sqsubseteq \exists \rho^r_{a,b}.C_b \in \mathcal{T}_\mathcal{D}$, and by (3.33), since $\exists r.A' \sqsubseteq A \in \mathcal{T}$, it follows that $\exists \rho^r_{ab}\,.\,A'_n \sqsubseteq A_n \in \mathcal{T}_\mathcal{D}$. We obtain a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\})$ as follows. Start with $\mathcal{F}_0 = \mathcal{T}' \cup \mathcal{T}_\mathcal{D} \cup \{C_a(a,l)\}$. Apply a rule of the form (3.11) using $C_a \sqsubseteq \exists \rho^r_{a,b}.C_b$ to obtain

$\rho^r_{a,b}(a, b', l)$ and $C_b(b', l)$ for some $b' \in \mathsf{N_N}$. Proceed as in **der** to obtain $A'_n(b', n)$. Apply a rule of the form (3.7) to obtain $\rho^r_{a,b}(a, b', n)$, then a rule of the form (3.10) using $\exists \rho^r_{ab} \,.\, A'_n \sqsubseteq A_n$ to get $A_n(a, n)$.

– $b \in \mathsf{N_N}$. By the form of the derivation rules, $r(a, b, n) \in \mathcal{F}_{m-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < m - 1$ such that for some A'', B' and $k \geqslant l$ (again, because $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}_{future}$-TBox), $\mathsf{prem}(f_i) = \{A'' \sqsubseteq \exists r.B', A''(a, k)\}$, $\mathsf{conc}(f_i) = \{r(a, b, k), B'(b, k)\}$. Thus $A''(a, k) \in \mathcal{F}_{i-1}$, and $(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1})$ is a derivation witnessing $(\mathcal{T}, \mathcal{D}) \vdash A''(a, k)$. By the induction hypothesis, there is a derivation $\mathbf{der}_1$ witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a, l)\}) \vdash A''_k(a, k)$. Moreover, we extract from $(\mathcal{F}_i, \ldots, \mathcal{F}_{m-1})$ a derivation witnessing $(\mathcal{T}, \{B'(b, k)\}) \vdash A'(b, n)$ (note that $n = k$ if $r \in \mathsf{N}^{\mathsf{loc}}_{\mathsf{R}}$). Since $(\mathcal{T}, \{B'(b, k)\}) \vdash A'(b, n)$, by Lemma B.10, $(\mathcal{T}', \{B'_k(b, k)\}) \vdash A'_n(b, n)$. Let $\mathbf{der}_2$ be a derivation witnessing this. We obtain a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, C_a(a, l)) \vdash A_n(a, n)$ as follows. Since $A'' \sqsubseteq \exists r.B' \in \mathcal{T}$, we have $A''_k \sqsubseteq \exists r.B'_k \in \mathcal{T}'$, by (3.29). Further, since $\exists r.A' \sqsubseteq A \in \mathcal{T}$, we have $\exists r.A'_n \sqsubseteq A_n \in \mathcal{T}'$, by (3.30). Thus, start from $\mathcal{F}'_0 = \mathcal{T}' \cup \mathcal{T}_\mathcal{D} \cup \{C_a(a, l)\}$, proceed as in $\mathbf{der}_1$ until you derive $A''_k(a, k)$. Apply a rule of the form (3.11) using $A''_k \sqsubseteq \exists r.B'_k$ to obtain $r(a, b, k)$ and $B'_k(b, k)$, and proceed as in $\mathbf{der}_2$ until you have $A'_n(b, n)$. If $r \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}}$, apply a rule of the form (3.7) to obtain $r(a, b, n)$. Otherwise, it means that $k = n$, so we already have $r(a, b, n)$. Finally, apply a rule of the form (3.10) using $\exists r.A'_n \sqsubseteq A_n$ to get $A_n(a, n)$.

($\Leftarrow$) Again, we show by induction on m that for all $A \in \mathsf{N_C}(\mathcal{T})$, $a \in \mathsf{N_I}$ and $n, n' \in \mathbb{N}$, if there exists a derivation of length m witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a, l)\}) \vdash A_n(a, n')$, then $n' = n$ and $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$.

The base case is $m = 1$. Then $\mathcal{F}_0 = \mathcal{T}' \cup \mathcal{T}_\mathcal{D} \cup \{C_a(a, l)\}$ and $A_n(a, n') \in \mathcal{F}_1$. Thus, f_1 is an application of a rule of the form (3.8) using $C_a \sqsubseteq \bigcirc^{n-l} A_n$, so it must be the case that $n' = n$ and $C_a \sqsubseteq \bigcirc^{n-l} A_n \in \mathcal{T}_\mathcal{D}$. By (3.31), it follows that $A(a, n) \in \mathcal{D}$, so $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$.

Induction step: assume that the property holds for $m-1$ and let $\mathcal{F}_0 \xrightarrow{f_1} \ldots \xrightarrow{f_m} \mathcal{F}_m$ be a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a, l)\}) \vdash A_n(a, n')$. If $A_n(a, n') \in \mathcal{F}_{m-1}$, the result follows by induction hypothesis. Otherwise, there are five possible cases for the last rule application f_m that produces $A_n(a, n')$:

- $\mathsf{prem}(f_m) = \{C_b(a, l), C_b \sqsubseteq \bigcirc^{n-l} A_n\}$ and f_m is the application of a rule of the form (3.8). However, by the form of concept inclusions of $\mathcal{T}'$ and $\mathcal{T}_\mathcal{D}$, it must be the case that $C_b = C_a$ since the derivation rules cannot produce a fact of the form $C_b(a, \ell)$ with $a \in \mathsf{N_I}$ and $a \neq b$. Hence we obtain $n' = n$ and $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$ as in the base case.

- $\mathsf{prem}(f_m) = \{A'_{n-k}(a, n' - k), A'_{n-k} \sqsubseteq \bigcirc^k A_n\}$ for some $k \in \{0, \ldots, \min(n-l, n'-l)\}$ (since $\mathcal{T}' \cup \mathcal{T}_\mathcal{D}$ is a $\mathcal{TEL}^\bigcirc_{future}$-TBox so that $n' - k \geqslant l$ and $n - k \geqslant l$ by definition of $\mathcal{T}'$), and f_m is the application of a rule of the form (3.8). Since $(\mathcal{F}_0, \ldots, \mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a, l)\}) \vdash A'_{n-k}(a, n' - k)$, by induction hypothesis, $n' - k = n - k$, i.e., $n' = n$, and $(\mathcal{T}, \mathcal{D}) \vdash A'(a, n-k)$. Since $A'_{n-k} \sqsubseteq \bigcirc^k A_n \in \mathcal{T}'$ it follows that $A' \sqsubseteq \bigcirc^k A \in \mathcal{T}$, by (3.27). We conclude that $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$ by applying a rule of the form (3.8).

- $\mathsf{prem}(f_m) = \{A'_n(a, n'), A''_n(a, n'), A'_n \sqcap A''_n \sqsubseteq A_n\}$ and f_m is the application of a rule of form (3.9). Since $A'_n(a, n')$ and $A''_n(a, n')$ are in $\mathcal{F}_{m-1}$, $(\mathcal{F}_0, \ldots, \mathcal{F}_{m-1})$ is a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a, l)\}) \vdash A'_n(a, n')$ and $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a, l)\}) \vdash A''_n(a, n')$. By the induction hypothesis, $n' = n$ and $(\mathcal{T}, \mathcal{D}) \vdash A'(a, n)$ and $(\mathcal{T}, \mathcal{D}) \vdash A''(a, n)$. Since $A'_n \sqcap A''_n \sqsubseteq A_n \in \mathcal{T}'$, $A' \sqcap A'' \sqsubseteq A \in \mathcal{T}$, by (3.28). We conclude that $(\mathcal{T}, \mathcal{D}) \vdash A(a, n)$ by applying a rule of the form (3.9).

- $\mathsf{prem}(f_m) = \{r(a,b,n'), A'_n(b,n'), \exists r.A'_n \sqsubseteq A_n\}$ and f_m is the application of a rule of form (3.10). By the form of the derivation rules, $r(a,b,n') \in \mathcal{F}_{m-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Hence, there must be an index $i < m-1$ such that for some A''_k, B'_k and $k' \geqslant l$, $\mathsf{prem}(f_i) = \{A''_k \sqsubseteq \exists r.B'_k, A''_k(a,k')\}$, $\mathsf{conc}(f_i) = \{r(a,b,k'), B'_k(b,k')\}$. Thus $A''_k(a,k') \in \mathcal{F}_{i-1}$, and $(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1})$ is a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \vdash A''_k(a,k')$. By the induction hypothesis, $k' = k$ and there is a derivation $\mathbf{der}_1$ witnessing $(\mathcal{T}, \mathcal{D}) \vdash A''(a,k)$. Moreover, we extract from $(\mathcal{F}_i, \ldots, \mathcal{F}_{m-1})$ a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{B'_k(b,k)\}) \vdash A'_n(b,n')$. Since B'_k does not appear in the left-hand sides of concept inclusions in $\mathcal{T}_\mathcal{D}$, it is not hard to see that, in fact, $(\mathcal{T}', \{B'_k(b,k)\}) \vdash A'_n(b,n')$. Moreover, since $\mathcal{T}'$ is a $\mathcal{TEL}^\circ_{future}$-TBox, $k \leqslant n'$, thus $n' - k \in \mathbb{N}$ (note that $n' = k$ if $r \in \mathsf{N}_\mathsf{R}^{\mathsf{loc}}$). Since $(\mathcal{T}', \{B'_k(b,k)\}) \vdash A'_n(b,n')$, $(\mathcal{T}', \{B'_k(b,0)\}) \vdash A'_n(b,n'-k)$, so by point *(i)* of Lemma B.10, $n = n' - k + k = n'$, so $(\mathcal{T}', \{B'_k(b,k)\}) \vdash A'_n(b,n)$ and thus, by point *(ii)* of Lemma B.10, $(\mathcal{T}, \{B'(b,k)\}) \vdash A'(b,n)$. Let $\mathbf{der}_2$ be a derivation witnessing it. We obtain a derivation witnessing $(\mathcal{T}, \mathcal{D}) \vdash A(a,n)$ as follows. Since $A''_k \sqsubseteq \exists r.B'_k \in \mathcal{T}'$, we have $A'' \sqsubseteq \exists r.B' \in \mathcal{T}$, by (3.29). Further, since $\exists r.A'_n \sqsubseteq A_n \in \mathcal{T}'$, we have $\exists r.A' \sqsubseteq A \in \mathcal{T}$, by (3.30). Thus, start from $\mathcal{F}'_0 = \mathcal{T} \cup \mathcal{D}$, proceed as in $\mathbf{der}_1$ until you derive $A''(a,k)$. Apply a rule of the form (3.11) to obtain $r(a,b,k)$ and $B'(b,k)$, and proceed as in $\mathbf{der}_2$ until you have $A'(b,n)$. If $r \in \mathsf{N}_\mathsf{R}^{\mathsf{rig}}$, apply a rule of the form (3.7) to obtain $r(a,b,n)$. Otherwise, it means that $k = n$, so we already have $r(a,b,n)$. Finally, apply a rule of the form (3.10) to get $A(a,n)$.

- $\mathsf{prem}(f_m) = \{\rho^r_{a,b}(a,b',n'), A'_n(b',n'), \exists \rho^r_{a,b}.A'_n \sqsubseteq A_n\}$ and f_m is the application of a rule of form (3.10). Then $A'_n(b',n') \in \mathcal{F}_{m-1}$. By the form of the derivation rules, $\rho^r_{a,b}(a,b',n') \in \mathcal{F}_{m-1}$ has been produced by the application of a rule of form (3.7) or (3.11). Moreover, there must be an index $i < m -$

1 such that $\mathsf{prem}(f_i) = \{C_a \sqsubseteq \exists \rho^r_{a,b}.C_b, C_a(a,l)\}$ (note that by the form of the concept inclusions in $\mathcal{T}' \cup \mathcal{T}_\mathcal{D}$, $C_a(a,l)$ is the only fact of the form $C_a(a,\ell)$ that can be derived since $a \in \mathsf{N_I}$) and $\mathsf{conc}(f_i) = \{\rho^r_{a,b}(a,b',l), C_b(b',l)\}$. We extract a derivation witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_b(b',l)\}) \vdash A'_n(b',n')$ of length at most $m-1$. Renaming b' to b, we get a derivation of length at most $m-1$ witnessing $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_b(b,l)\}) \vdash A'_n(b,n')$. By the induction hypothesis, $n' = n$ and there is a derivation **der** witnessing $(\mathcal{T},\mathcal{D}) \vdash A'(b,n)$. We obtain a derivation witnessing $(\mathcal{T},\mathcal{D}) \vdash A(a,n)$ as follows. Since $C_a \sqsubseteq \exists \rho^r_{a,b}.C_b \in \mathcal{T}_\mathcal{D}$, we have $r(a,b,\ell) \in \mathcal{D}$, by (3.32). Further, since $\exists \rho^r_{a,b}.A'_n \sqsubseteq A_n \in \mathcal{T}_\mathcal{D}$, either $r \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}}$ or $\ell = n$, and $\exists r.A' \sqsubseteq A \in \mathcal{T}$, by (3.33). Thus, start with $\mathcal{F}_0 = \mathcal{T} \cup \mathcal{D}$. Use **der** to derive $A'(b,n)$. If $r \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}}$, apply a rule of the form (3.7) to obtain $r(a,b,n)$ (otherwise, it is already in $\mathcal{D}$). Conclude by applying a rule of the form (3.10) using $\exists r.A' \sqsubseteq A$. □

By Theorem 3.6, one can construct in polynomial time w.r.t. $|\mathcal{T}'| + |\mathcal{T}_\mathcal{D}|$ (hence in polynomial time w.r.t. $|\mathcal{T}| + |\mathcal{D}|$) a unary conjunctive grammar $G_{\mathcal{T}' \cup \mathcal{T}_\mathcal{D}} = (N, \{c\}, R)$ such that there exists $\mathcal{N}_{C_a A} \in N$ such that $c^{n-l} \in L_{G_{\mathcal{T}' \cup \mathcal{T}_\mathcal{D}}}(\mathcal{N}_{C_a A})$ iff $\mathcal{T}' \cup \mathcal{T}_\mathcal{D} \models (C_a \sqsubseteq \bigcirc^{n-l} A_n)$. By Proposition 3.3, $\mathcal{T}' \cup \mathcal{T}_\mathcal{D} \models (C_a \sqsubseteq \bigcirc^{n-l} A_n)$ iff $(\mathcal{T}' \cup \mathcal{T}_\mathcal{D}, \{C_a(a,l)\}) \models A_n(a,n)$, and by Lemma B.11, the latter is equivalent to $(\mathcal{T},\mathcal{D}) \models A(a,n)$. Hence, $(\mathcal{T},\mathcal{D}) \models A(a,n)$ iff $c^{n-l} \in L_{G_{\mathcal{T}' \cup \mathcal{T}_\mathcal{D}}}(\mathcal{N}_{C_a A})$. □

B.6.2 Proof of Theorem 3.24

We first recall several definitions and results from the literature.

A bound for Parikh images Let $G = (N, \Sigma, R)$ be a context-free grammar. If $\mathcal{N} \to \alpha \in R$, we write $\alpha/_N$ and $\alpha/_\Sigma$ for the words obtained from α by projecting on N and on Σ, respectively. The *degree* of G is the number $m = -1 + \max_{\mathcal{N} \to \alpha \in R} |\alpha/_N|$, and the *productivity* of G is the number $p = \sum_{\mathcal{N} \to \alpha \in R} |\alpha/_\Sigma|$.

Theorem B.12 (To [2010]). *Let M be a nondeterministic state automaton with s states over an alphabet of l letters. Then the Parikh image of $L(M)$ is a union of $\mathcal{O}\left(s^{l^2+3l+3}l^{4l+6}\right)$ linear sets with at most l period vectors; the maximum component of any offset vector is $\mathcal{O}\left(s^{3l+3}l^{4l+6}\right)$, and the maximum component of any period vector is at most s.*

Theorem B.13 (Esparza et al. [2011]). *If G is a context-free grammar with n nonterminals, degree m and productivity p, then one can construct, in polynomial space,[2] a nondeterministic finite state automaton M with $\binom{n+nm+1}{n} \cdot p$ states such that $L(G)$ and $L(M)$ have the same Parikh image.*

Corollary B.14. *For every context-free grammar $G = (N, \Sigma, R, \mathcal{S})$ of degree n and productivity p, with $|N| = n$ and $|\Sigma| = t$, the bounds of Theorem B.12 hold with $s = \binom{n+nm+1}{n} \cdot p$ and $l = t$.*

***datalog*$_{1\mathrm{S}}$.** We informally describe *datalog*$_{1\mathrm{S}}$ programs. For a formal definition, the reader is referred to Chomicki and Imieliński [1988].

Fix a dedicated *temporal variable* t. A *datalog*$_{1\mathrm{S}}$ program Π is a finite set of rules of the form

$$B(\overline{x}, t) \leftarrow A_1(\overline{x}_1, t + i_1), \ldots, A_k(\overline{x}_k, t + i_k) \tag{B.3}$$

where $B, A_1, \ldots, A_k$ are predicate symbols, $\overline{x}_i$ are tuples of variables and $i_1, \ldots, i_k \in \mathbb{Z}$ are given in unary.[3] Since we work with description logics, we assume that the arities of B and A_i are either 2 or 3.

Given a program Π and a temporal database $\mathcal{D}$, their *canonical model* (the least Herbrand model, in the terminology of Chomicki and Imieliński [1988]) is obtained by exhaustive application of the

[2] This is not stated explicitly in [Esparza et al., 2011], but follows from their construction.

[3] (Chomicki and Imieliński [1988] also allow atoms of the form $A(\overline{x}, k)$, for $k \in \mathbb{Z}$, but we will not need them).

rules of Π to $\mathcal{D}$. We write $(\Pi, \mathcal{D}) \models A(a, n)$ if $A(a, n)$ holds in the canonical model of Π and $\mathcal{D}$. We write $\Pi \models A \sqsubseteq \bigcirc^k B$ if $(\Pi, \mathcal{D}) \models A(a, n)$ implies $(\Pi, \mathcal{D}) \models B(a, n + k)$, for any temporal database $\mathcal{D}$.

We will use *datalog*$^{\bigcirc\Diamond}$, instead, and rely on the complexity results of Theorem 2.8. We are now ready to prove Theorem 3.24.

Theorem 3.24. *The following statements hold.*

(i) Every $\mathcal{TEL}^{\bigcirc}_{lin}$ TBox $\mathcal{T}$ is ultimately periodic, $\|\mathcal{T}\| \leqslant 2^{\text{poly}(|\mathcal{T}|)}$.

(ii) Query answering with $\mathcal{TEL}^{\bigcirc}_{lin}$ TBoxes is NL*-complete, for data complexity.*

(iii) Query answering with $\mathcal{TEL}^{\bigcirc}_{lin}$ TBoxes without local role names is in ExpSpace*, for combined complexity.*

Proof. (i) Let $\mathcal{T}$ be a $\mathcal{TEL}^{\bigcirc}_{lin}$-TBox. The proof that $\mathcal{T}$ is ultimately periodic is in the main text. For the bound on $\|\mathcal{T}\|$, fix any $A, B \in \mathsf{N_C}(\mathcal{T})$ and let $\Gamma_{\mathcal{T}} = (N, \{c, d\}, R, \mathcal{N}_{AB})$ be as in Theorem 3.15. Then by Definition 3.16, the parameters in Corollary B.14 are as follows: $n, m, p \leqslant \text{poly}(|\mathcal{T}|)$, and $t = 2$. Thus, $s \leqslant 2^{\text{poly}(|\mathcal{T}|)}$, and $l = 2$, rendering $p(L(G))$ to be a union of at most $2^{\text{poly}(|\mathcal{T}|)}$ linear sets with at most 2 periods, the maximum entry of any offset and any period is $2^{\text{poly}(|\mathcal{T}|)}$. It follows that $\|\mathcal{T}\| \leqslant 2^{\text{poly}(|\mathcal{T}|)}$.

(ii), (iii) The lower bound for data complexity is from linear $\mathcal{EL}$ [Dimartino et al., 2016]. For the upper bound, we recall from Gutiérrez-Basulto et al. [2016a] that every ultimately periodic (w.r.t. quasimodels) $\mathcal{TEL}^{\bigcirc}$-TBox $\mathcal{T}$ can be translated, in polynomial time, to a *datalog*$_{1\text{S}}$ program $\Pi_{\mathcal{T}}$, such that for any $\mathcal{D}$ and $A(a, n)$ we have $(\mathcal{T}, \mathcal{D}) \models A(a, n)$ iff $(\Pi_{\mathcal{T}}, \mathcal{D}) \models A(a, n)$. To understand the translation, the reader will need the definition of quasimodels given in Section B.1; Gutiérrez-Basulto et al. [2016a] also provide intuitive explanations that we omit here. The program $\Pi^1_{\mathcal{T}}$ consists of the following rules.

$$r(x, y, t) \leftarrow r(x, y, t \pm 1) \qquad \text{for } r \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}}(\mathcal{T}) \tag{B.4}$$

$$B(x, t) \leftarrow A(x, t), A'(x, t) \qquad \text{for } A \sqcap A' \sqsubseteq B \in \mathcal{T} \tag{B.5}$$

$$B(x,t) \leftarrow r(x,y,t), A(y,t) \qquad \text{for } \exists r.A \sqsubseteq B \in \mathcal{T} \tag{B.6}$$

Further, let $\mathfrak{Q} = \{\pi_d \mid d \in \mathsf{N_C}(\mathcal{T})\}$ be the canonical quasimodel of $(\mathcal{T}, \emptyset)$. For each trace π_A, with integers m_P , p_P, m_F , p_F given by ultimate periodicity of $\mathcal{T}$, the program $\Pi^2_{\mathcal{T}}$ contains the following rules with a new predicate F_A:

$$B(x,t) \leftarrow A(x,t-i) \quad \text{for } 0 \leqslant i < m_F,\ B \in \pi_A(i) \tag{B.7}$$
$$F_A(x,t) \leftarrow A(x,t-m_F) \tag{B.8}$$
$$F_A(x,t) \leftarrow F_A(x,t-p_F) \tag{B.9}$$
$$B(x,t) \leftarrow F_A(x,t-i) \ \text{ for } 0 \leqslant i < p_F, B \in \pi_A(m_F+i) \tag{B.10}$$

and symmetric rules with m_P, p_P and a predicate P_A. Intuitively, the rules of the form (B.7) replicate the initial part of π_A, from 0 to m_F. The rules of the forms (B.8) and (B.9) mark the start of each period with F_A while the rules of the form (B.10) replicate the period of π_B, starting from each marker F_A.

Finally, $\Pi_{\mathcal{T}} = \Pi^1_{\mathcal{T}} \cup \Pi^2_{\mathcal{T}}$.

Proposition B.15 (Gutiérrez-Basulto et al. [2016a])**.** *For any $\mathcal{TEL}^{\circ}$-TBox $\mathcal{T}$, temporal database $\mathcal{D}$ and TAQ $A(a,n)$, $(\mathcal{T},\mathcal{D}) \models A(a,n)$ iff $(\Pi_{\mathcal{T}},\mathcal{D}) \models A(a,n)$.*

Note that rules of the forms (B.5) and (B.6) may be not linear. Now, suppose $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}_{lin}$-TBox, which excludes rules of the form (B.5). Our goal is to rewrite $\Pi_{\mathcal{T}}$ into a linear *datalog*$^{\circ\Diamond}$ program $\Phi_{\mathcal{T}}$, using the additional power of operators $\Diamond/\Diamond^-$. We will also use our definition of ultimate periodicity (w.r.t. concept inclusions) and Theorem B.12 to have a program of reasonable size.

We define $\Phi^1_{\mathcal{T}}$ to contain, for each rule of the form (B.6), the following rules:

$$B(x) \leftarrow r(x,y), A(y) \qquad\qquad (\text{B.6}^{loc})$$
$$B(x) \leftarrow \Diamond r(x,y), A(y) \qquad \text{if } r \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}} \qquad (\text{B.6}^{rig}_{F})$$
$$B(x) \leftarrow \Diamond^- r(x,y), A(y) \qquad \text{if } r \in \mathsf{N}^{\mathsf{rig}}_{\mathsf{R}} \qquad (\text{B.6}^{rig}_{P})$$

Further, for every $A, B \in \mathsf{N_C}(\mathcal{T})$, we take a representation

$$\begin{aligned} \{n \in \mathbb{Z} \mid \mathcal{T} \models A \sqsubseteq \bigcirc^n B\} &= \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_m \\ \mathcal{L}_i &= \{b^i + k_1 p_1^i + \ldots k_l p_l^i \mid k_1, \ldots, k_l \in \mathbb{N}\}. \end{aligned} \tag{B.11}$$

Then, the program $\Phi^2_{\mathcal{T}}$ contains the following rules, $i \in \{1, \ldots, m\}$.

$$F_i^{AB}(x) \leftarrow \bigcirc^{-b^i} A(x) \tag{B.12}$$

$$F_i^{AB}(x) \leftarrow \bigcirc^{-p_j^i} F_i^{AB}(x) \qquad \text{for } 1 \leqslant j \leqslant l \tag{B.13}$$

$$B(x) \leftarrow F_i^{AB}(x) \tag{B.14}$$

Finally, $\Phi_{\mathcal{T}} = \Phi^1_{\mathcal{T}} \cup \Phi^2_{\mathcal{T}}$.

It is readily seen that $\Phi_{\mathcal{T}}$ is a linear program, and that $|\Phi_{\mathcal{T}}| \leqslant \mathrm{poly}(\|\mathcal{T}\|)$. From point *(i)* we get the bound $|\Phi_{\mathcal{T}}| \leqslant 2^{\mathrm{poly}(|\mathcal{T}|)}$. Moreover, if $\mathsf{N_R}(\mathcal{T}) \subseteq \mathsf{N_R^{rig}}$, $\Phi_{\mathcal{T}}$ can be effectively constructed from $\mathcal{T}$. Indeed, in this case, one obtains the grammar $\Gamma_{\mathcal{T}}$ of Theorem 3.15 in polynomial time. Then, using the automaton of Theorem B.13, one obtains representations (B.11) that respect the bounds of Corollary B.14.

Points *(ii)* and *(iii)* then follow from the fact that $(\Pi_{\mathcal{T}}, \mathcal{D}) \models A(a, n)$ iff $(\Phi_{\mathcal{T}}, \mathcal{D}) \models A(a, n)$. To see the latter, recall that when $\mathcal{T}$ is a $\mathcal{TEL}^{\circ}_{lin}$-TBox, $\Pi^1_{\mathcal{T}}$ does not contain rules of the form (B.5). It is immediate that $\Pi^1_{\mathcal{T}}$ is equivalent to $\Phi^1_{\mathcal{T}}$. The fact that $\Pi^2_{\mathcal{T}}$ is equivalent to $\Phi^2_{\mathcal{T}}$ follows from the lemma below, given the facts that rules of the forms (B.7)-(B.10) and (B.12)-(B.14) allow to derive $D(b, \ell')$ from $C(a, \ell)$ only if $a = b$.

Lemma B.16. *For any $\mathcal{TEL}^{\circ}_{lin}$-TBox $\mathcal{T}$ and $A, B \in \mathsf{N_C}(\mathcal{T})$, $\Pi^2_{\mathcal{T}} \models A \sqsubseteq \bigcirc^n B$ iff $\Phi^2_{\mathcal{T}} \models A \sqsubseteq \bigcirc^n B$.*

Proof. By the form of the rules (B.7)-(B.10), $\Pi^2_{\mathcal{T}} \models A \sqsubseteq \bigcirc^n B$ iff $B \in \pi_A(n)$. By Lemma B.2, $B \in \pi_A(n)$ iff $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$. Finally, by the form of the rules (B.12)-(B.14), $\mathcal{T} \models A \sqsubseteq \bigcirc^n B$ iff $\Phi^2_{\mathcal{T}} \models A \sqsubseteq \bigcirc^n B$. □

Appendix C

Proofs for Chapter 4

C.1 Proofs for Section 4.3

Lemma 4.3. *For any database D, $(\mathcal{O}, D) \models G(a)$ if and only if $(\Pi_Q, D) \models \mathsf{Goal}(a)$.*

Proof. Let $Q = (\mathcal{O}, G)$, with $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$, Π_Q a disjunctive datalog program obtained from Q as described in Section 4.3, D an arbitrary database, $a \in \text{obj}(D)$ and G' a new concept name not appearing in $\mathcal{O}$.

It suffices to prove that there exists a model of

$$\mathcal{K} = (\mathcal{O} \cup \{G' \sqcap G \sqsubseteq \bot\}, D \cup \{G'(a)\})$$

if and only if there exists a model of

$$\mathcal{K}' = (\Pi_Q \cup \{\bot \leftarrow G'(x) \wedge G(x)\}, D \cup \{G'(a)\})$$

($\Rightarrow$) Let $\mathcal{M} = (\Delta, \cdot^{\mathcal{M}})$ be a model of $\mathcal{K}$. We construct a model $\mathcal{N} = (\Delta, \cdot^{\mathcal{N}})$ for $\mathcal{K}'$ using the same domain Δ. Note that in $\mathcal{N}$ we should care about the interpretation of the constants d_{R^-} of Π_Q,

while $\mathcal{M}$ may interpret them in any way. For every $a \in \mathit{Const}$ which is not a d_{R^-}, we set $a^{\mathcal{N}} = a^{\mathcal{M}}$. For every role type $R \subseteq \mathrm{Rol}(\mathcal{O})$, choose some d from the set

$$\{d \in \Delta \mid \forall r \in R,\ \exists d' \in \Delta \colon (d', d) \in r^{\mathcal{M}}\}$$

If this set is empty, take any $d \in \Delta$. We set $d^{\mathcal{N}}_{R^-} = d$. Finally, for each concept name A set $A^{\mathcal{N}} = A^{\mathcal{M}}$ and for each role r set $r^{\mathcal{N}} = r^{\mathcal{M}}$ and $E^{\mathcal{N}}_r = \{d \in \Delta \mid \exists d' \in \Delta \colon (d, d') \in r^{\mathcal{M}}\}$. By construction, $a \notin G^{\mathcal{N}}$, and it can be easily verified that $\mathcal{N}$ is a model of $\mathcal{K}'$.

($\Leftarrow$) Conversely, suppose $\mathcal{N} = (\Delta, \cdot^{\mathcal{N}})$ is a model of $\mathcal{K}'$. We can choose $\mathcal{N}$ to be a Herbrand model, so that $\Delta = \mathrm{obj}(D) \cup \{d_{R^-} \mid R \subseteq \mathrm{Rol}(\mathcal{O})$ is a role type$\}$, with an obvious interpretation $a^{\mathcal{N}} = a$ for $a \in \mathrm{obj}(D)$ and $d^{\mathcal{N}}_{R^-} = d_{R^-}$ for any R.

We construct a model $\mathcal{M} = (\Delta, \cdot^{\mathcal{M}})$ for $\mathcal{K}$ using the same domain Δ. We also borrow the interpretation of constants from $\mathcal{N}$. Further, for each concept name A we set $A^{\mathcal{M}} = A^{\mathcal{N}}$.

The interpretation of roles is defined in three steps. First, for each role r and any $d, d' \in \Delta$ we let $(d, d') \in r^{\mathcal{M}}$ if $(d, d') \in r^{\mathcal{N}}$. This way we satisfy the facts $r(a, a') \in D$.

Second, we ensure that E_r relations in $\mathcal{N}$ correspond to $\exists r$ in $\mathcal{M}$. Consider any d such that $d \in E^{\mathcal{N}}_w$, for some role w. By (4.6), there exists a role type R such that $\mathcal{E}_R(d) \wedge \mathcal{E}_{R^-}(d_{R^-})$ is true in $\mathcal{N}$. We set $(d, d_{R^-}) \in r^{\mathcal{M}}$ for each $r \in R$.

Third, we ensure that roles of $\mathcal{M}$ satisfy the RBox $\mathcal{R}$. For this we "recover" the inferred roles from their encoding in $\mathcal{N}$ with pairs of E_r, E_r^- predicates. For a pair $d, d' \in \Delta$ define $\mathrm{Rol}_{\mathcal{M}}(d, d')$ to be the set of those roles r for which we have already established $(d, d') \in r^{\mathcal{M}}$ on the previous steps. Our goal is to enrich this set so that it forms a role type.

Note that if at least one of d and d' is anonymous, $\mathrm{Rol}_{\mathcal{M}}(d, d')$ is a role type by construction. So suppose $d, d' \in \mathrm{obj}(D)$. Then by (4.5) there exists a role type R such that $\mathcal{E}_R(d) \wedge \mathcal{E}_{R^-}(d')$ is true in $\mathcal{N}$. Finally, we set $(d, d') \in r^{\mathcal{M}}$ for all $r \in R$.

Now, for every two objects $d, d' \in \Delta$ the set $\{r \mid (d, d') \in r^{\mathcal{M}}\}$ is a role type. Therefore, $\mathcal{M} \models \mathcal{R}$.

It remains to verify that for each role r we have $\exists r^{\mathcal{M}} = E_r^{\mathcal{N}}$. Indeed, for each $d \in \Delta$, if $E_r(d)$ holds in $\mathcal{N}$ then we have $r(d, d_{R^-})$ in $\mathcal{M}$ for some type $R \ni r$. Therefore $d \in \exists r^{\mathcal{M}}$ by the semantics of description logic. Conversely, suppose that $r(d, d')$ holds in $\mathcal{M}$. If $r(d, d') \in D$, then we have $E_r(d), E_{r^-}(d')$ in $\mathcal{N}$ by the rule (4.7). If, on the other hand, $r(d, d')$ was added to $\mathcal{M}$ on the second or the third step of role interpretation, then by construction $E_r(d)$ and $E_{r^-}(d')$ must be true in $\mathcal{N}$.

Therefore, the interpretation of concepts in $\mathcal{M}$ coincides with the corresponding unary predicates of $\mathcal{N}$. Since $\mathcal{N}$ satisfies rules of type (4.4), $\mathcal{M} \models \mathcal{T}$. Therefore, $\mathcal{M}$ is a model of $\mathcal{K}$. □

C.2 Proofs for Section 4.4

Lemma 4.5. *For a flat TDL-Lite*^{4} *KB $\mathcal{K}$, $\mathcal{K}$ is consistent whenever $\mathcal{K}^\ddagger$ is consistent.*

Proof. ($\Rightarrow$) Let $\mathfrak{M} = (\Delta, \cdot^{\mathfrak{M}})$ be a model of $\mathcal{K}$. Then there is a model of $\mathcal{K}^\ddagger$. Indeed, consider the structure $\mathfrak{N} = (\Delta, \cdot^{\mathfrak{N}})$ with the same domain Δ, where the interpretation of constants, concept and role names is copied from $\mathfrak{M}$. Additionally, let $E_r^{\mathfrak{N},\ell} = \{d \mid \exists d' \in \Delta\colon (d, d') \in r^{\mathfrak{M},\ell}\}$. It is easy to see that $\mathfrak{N}$ satisfies the axiom (4.12) by construction. Furthermore, by the definition of the translation $\cdot^\ddagger$, $\mathfrak{N} \models \mathcal{T}^\ddagger$, and thanks to (4.12), $\mathfrak{N} \models \mathcal{R}^\ddagger$.

($\Leftarrow$) Conversely, suppose $\mathfrak{N} = (\Delta, \cdot^{\mathfrak{N}})$ is a model of $\mathcal{K}^\ddagger$. W.l.o.g., $d \in E_r^{\mathfrak{N},\ell}$ only when there is $d' \in \Delta$ such that $(d, d') \in r^{\mathfrak{N},\ell}$. We construct a structure $\mathfrak{M} = (\Delta, \cdot^{\mathfrak{M}})$, copy-pasting the interpretation of constants, concept and role names from $\mathfrak{N}$. It is straightforward that $\mathfrak{M} \models \mathcal{K}$. □

Lemma 4.6. *For a TDL-Lite^4 KB $\mathcal{K}$, $\mathcal{K}$ is consistent whenever $\mathcal{K}^\star$ is consistent.*

Proof. ($\Rightarrow$) Let $\mathfrak{M} = (\Delta, \cdot^{\mathfrak{M}})$ be a model of $\mathcal{K}$. We construct a model $\mathfrak{N} = (\Delta, \cdot^{\mathfrak{N}})$ for $\mathcal{K}^\star$ with the same domain Δ. Once again, we copy from $\mathfrak{M}$ the interpretation of constants, concept and role names. Additionally, $E_r^{\mathfrak{N},\ell} = \{d \mid \exists d' \in \Delta \colon (d, d') \in r^{\mathfrak{M},\ell}\}$. By construction, $\mathfrak{N} \models \mathcal{K}^\star$.

($\Leftarrow$) Conversely, suppose $\mathfrak{N} = (\Delta, \cdot^{\mathfrak{N}})$ is a model of $\mathcal{K}^\star$. We construct a model $\mathfrak{M} = (\Delta \times \mathbb{N}, \cdot^{\mathfrak{M}})$ for $\mathcal{K}$ with the domain $\Delta \times \mathbb{N}$. In other words, for each natural number n we create a copy of Δ, and the domain of $\mathfrak{M}$ is a disjoint union of these copies.

We now define the interpretation function. For each $a \in \mathrm{obj}(\mathcal{D})$, we set $a^{\mathfrak{M}} = (a^{\mathfrak{N}}, 0)$. Further, for each concept name A, $d \in \Delta$, $n \in \mathbb{N}$, we set $(d, n) \in A^{\mathfrak{M},\ell}$ if $d \in A^{\mathfrak{N},\ell}$. The interpretation of roles is done in two steps. Fix a $\ell \in \mathbb{Z}$. First, for each role r and each $d, d' \in \Delta$, and $n \in \mathbb{N}$, we let $\big((d, n), (d', n)\big) \in r^{\mathfrak{M},\ell}$ if $(d, d') \in r^{\mathfrak{N},\ell}$. Second, for each role r, consider any $d \in E_r^{\mathfrak{N},\ell}$. By (4.13) there exist a role type R and $d' \in \Delta$, such that $r \in R$, and $\mathfrak{N}, \ell \models \mathcal{E}_R(d)$ and $\mathfrak{N}, \ell \models \mathcal{E}_{R^-}(d')$. Then for each $w \in R$ and $n \in \mathbb{N}$ we set $\big((d, n), (d', n+1)\big) \in w^{\mathfrak{M},\ell}$. This finishes the construction of $\mathfrak{M}$.

It remains to verify that $\mathfrak{M}$ is a model of $\mathcal{K}$. Observe that, by construction, for each concept name A, $A^{\mathfrak{M},\ell} = A^{\mathfrak{N},\ell}$. Moreover, for each role r, $E_r^{\mathfrak{N},\ell} = \{d \mid \exists d' \in \Delta \colon (d, d') \in r^{\mathfrak{M},\ell}\}$. By the definition of $\cdot^\ddagger$ on concept inclusions, $\mathfrak{M} \models \mathcal{T}$. Then, consider any $(d, n), (d', m) \in \Delta \times \mathbb{N}$. If $n = m$, then the roles between the two elements were introduced on the first step and taken from $\mathfrak{N}$. By the fact that $\mathfrak{N} \models \mathcal{R}^\ddagger$, they form a role type. If $n \neq m$, then the roles between the two elements were introduced on the second step and by construction form a role type. It follows that $\mathfrak{M}$ satisfies $\mathcal{R}$. □

Lemma 4.7. *For a TDL-Lite^4 KB $\mathcal{K}$, $\mathcal{K}^\star$ is consistent whenever $\mathcal{K}^\bullet$ is consistent.*

Proof. ($\Rightarrow$) Let $\mathfrak{N} = (\Delta_{\mathfrak{N}}, \cdot^{\mathfrak{N}})$ be a model of $\mathcal{K}^\bullet$. We construct a

model $\mathfrak{M} = (\Delta_{\mathfrak{M}}, \cdot^{\mathfrak{M}})$ of $\mathcal{K}^\star$ as follows. Let $\mathfrak{N}[k] = (\Delta_{\mathfrak{N}[k]}, \cdot^{\mathfrak{N}[k]})$ denote a $\mathcal{QTL}$ interpretation with $\Delta_{\mathfrak{N}[k]} = \Delta_{\mathfrak{N}} \times \{k\}$ and the interpretation function $\cdot^{\mathfrak{N}}$ such that for every $\ell \in \mathbb{Z}$ the projection of $\cdot^{\mathfrak{N}[k],\ell}$ on $\Delta_{\mathfrak{N}}$ coincides with $\cdot^{\mathfrak{N},\ell+k}$. Further, let $\Delta_{\mathfrak{M}} = \Delta_{\mathfrak{N}} \cup \bigcup_{k\in\mathbb{Z}} \Delta_{\mathfrak{N}[k]}$. Define $\cdot^{\mathfrak{M}}$ so that on $\mathrm{obj}(\mathcal{D})$ it coincides with $\cdot^{\mathfrak{N}}$, and for relation symbols it coincides with $\cdot^{\mathfrak{N}}$ on $\Delta_{\mathfrak{N}}$ and with $\cdot^{\mathfrak{N}[k]}$ on $\Delta_{\mathfrak{N}[k]}$. Clearly, $\mathfrak{M}$ is a model of $\mathcal{K}^\bullet$. To see that $\mathfrak{M}$ is also a model of $\mathcal{K}^\star$, observe that it satisfies the formula (4.13), since for each r and each $\ell \in \mathbb{Z}$ it has an element (d_{R^-}, ℓ), such that

$$\Box\Box^- \forall x \; (E_r(x) \quad \rightarrow \bigvee_{r\in \mathrm{Tp}(R)} \big(\mathcal{E}_R(x) \wedge \mathcal{E}_R^-((d_{R^-}, \ell))\big) \qquad \text{(C.1)}$$

is true in $\mathfrak{M}$.

($\Leftarrow$) Let $\mathfrak{M} = (\Delta, \cdot^{\mathfrak{M}})$ be a model of $\mathcal{K}^\star$. We create a model $\mathfrak{N} = (\Delta_{\mathfrak{N}}, \cdot^{\mathfrak{N}})$ for $\mathcal{K}^\bullet$. For each role type R consider the following interpretation $\mathfrak{M}_R = (\Delta_R, \cdot^{\mathfrak{M}_R})$. There are two possible cases. Case A: there exists $\ell \in \mathbb{Z}$ and $\delta_R \in \Delta$, with $\delta_R \in (\mathcal{E}_R^-)^{\mathfrak{M},\ell}$. Then let $\mathfrak{M}_R = \mathfrak{M}[-\ell]$. Case B: there is no such ℓ and δ_R. Then let $\mathfrak{M}_R = \mathfrak{M}$.

We set $\Delta_{\mathfrak{N}} = \Delta_{\mathfrak{M}} \cup \bigcup_R \Delta_R$ and define $\cdot^{\mathfrak{N}}$ as follows. On $\mathrm{obj}(\mathcal{D})$ it coincides with $\cdot^{\mathfrak{M}}$. For each role type R, in Case A we set $P_R^{\mathfrak{N}}$ to be true, and $d_{R^-}^{\mathfrak{N}} = \delta_R$. In Case B we set $P_R^{\mathfrak{N}}$ to be false, and $d_{R^-}^{\mathfrak{N}}$ to be any $\delta \in \Delta_{\mathfrak{N}}$. For predicates, $\cdot^{\mathfrak{N}}$ coincides with $\cdot^{\mathfrak{M}}$ on Δ and $\cdot^{\mathfrak{M}_R}$ on Δ_R. Then $\mathfrak{N}$ satisfies (4.14) and (4.15), hence it is a model of $\mathcal{K}^\bullet$. $\square$

Appendix D

Proofs for Chapter 5

D.1 Proofs for Section 5.2

Lemma 5.2. *For any temporal CQ $q(x_1, \dots, x_k)$ without $\Diamond$ and $\Diamond^-$, $\mathcal{D}, 0 \models q(a_1, \dots, a_k)$ if and only if there is a homomorphism h from $\mathcal{D}_q$ to $\mathcal{D}$ such that $h(a_{x_i}) = a_i$, for $i = 1, \dots, k$.*

Proof. Consider the body $\varphi(x_1, \dots, x_k, u_1, \dots, u_s)$ of q, assuming w.l.o.g. that φ is a conjunction of temporalised atoms of the form $\bigcirc^p R(z_1, \dots, z_m)$. By definition, $\mathcal{D}, 0 \models q(a_1, \dots, a_k)$ iff there are $b_1, \dots, b_s \in \Delta_{\mathcal{D}}$ such that $\mathcal{D}, 0 \models \varphi(a_1, \dots, a_k, b_1, \dots, b_s)$. The mapping h defined as $h(a_{x_i}) = a_i, 1 \leqslant i \leqslant k,\ h(u_j) = b_j, 1 \leqslant j \leqslant s$, is the required homomorphism from $\mathcal{D}_q$ to $\mathcal{D}$. Conversely, if h is such a homomorphism satisfying the condition $h(a_{x_i}) = a_i, 1 \leqslant i \leqslant k$, then we obtain $b_1, \dots, b_s$ by setting $b_j = h(u_j)$. □

D.2 Proofs for Section 5.3

Lemma 5.4. *If M halts, (Π_M, G) is* FO(<)*-rewritable. Otherwise, answering it is* NL*-hard.*

Proof. (i) Suppose that M halts and let (5.7) be the (complete) computation (of length m). We will show that for any temporal database $\mathcal{D}$ over Σ:

$$(\Pi_M, \mathcal{D}) \models G(a_0, \ell) \Leftrightarrow \text{ there exists } w(x) \in \mathit{expand}(\Pi_M, G) \text{ with } \leqslant m+2 \text{ variables such that } \mathcal{D}, \ell \models w(a_0) \quad \text{(D.1)}$$

The direction ($\Leftarrow$) is immediate. For the other direction, suppose $(\Pi_M, \mathcal{D}) \models G(a_0, \ell)$ for some $a_0 \in \mathrm{obj}(\mathcal{D})$ and a timestamp ℓ. By (5.23), $(\Pi_M, \mathcal{D}) \models S_0(a_0, \ell)$ and $\mathcal{D}, \ell \models U_1(a_0, a') \wedge U_2(a_0, a')$ for some $a' \in \mathrm{obj}(\mathcal{D})$. Since $(\Pi_M, \mathcal{D}) \models S_0(a_0, \ell)$, three cases are possible:

(i) $(\Pi_M, \mathcal{D}) \models \Diamond^* RV(a_0, \ell)$ by (5.8). Then, $\mathcal{D}, \ell'$, for some $\ell' \geqslant \ell$, satisfies the body of one of the rules (5.9)-(5.11) with x equal to a_0. Suppose $\ell' > \ell$ and $\mathcal{D}, \ell'$ satisfies the body of (5.9) (the other cases are analogous). Then, since $m \geqslant 0$, $w(x) = \Diamond(T(x, y) \wedge \Diamond T(x, z)) \in \mathit{expand}(\Pi_M, G)$, and $\mathcal{D}, \ell \models w(a_0)$, we are done.

(ii) either $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_1^{\varepsilon_1}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^\alpha U_1(a_0, a')$ by (5.20) or $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_2^{\varepsilon_2}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^\beta U_2(a_0, a')$ by (5.21), for some transition $\delta(s_0, \alpha, \beta) = (s_j, \varepsilon_1, \varepsilon_2)$. We consider the first case only (the second is analogous). Furthermore, we assume that $\alpha = +$ and $\varepsilon_1 = -1$ (the cases for $\alpha = 0$ and $\varepsilon_1 \in \{0, 1\}$ are analogous). Then, $(\Pi_M, \mathcal{D}) \models NE_1^{\varepsilon_1}(a', \ell')$ for some $\ell' \geqslant \ell$ and $\mathcal{D}, \ell'' \models U_1(a_0, a')$ for some $\ell'' > \ell$. We suppose $\ell' > \ell$, the alternative case is analogous. Because $(\Pi_M, \mathcal{D}) \models NE_1^{\varepsilon_1}(a', \ell')$ and $\varepsilon_1 = -1$, $\mathcal{D}, \ell'$ satisfies the body of (5.12) or the next two rules in the list, with y equal to a'. Suppose the body of (5.12) is satisfied (the remaining two cases are analogous). Then, $\mathcal{D} \models U_1(a_0, a', \ell)$, for some $a' \in \mathrm{obj}(\mathcal{D})$. We take $w(x) = \Diamond U_1(x, y) \wedge \Diamond(U_1(y, z) \wedge U_1(y, x))$. As explained above, $\mathcal{D}, \ell \models w(a_0)$. Clearly, the number of variables in w is $\leqslant m + 2$ and we are done.

(*iii*) $\mathcal{D}, \ell \models T(a_0, a_1) \wedge \Diamond^{\alpha} U_1(a_0, a_1) \wedge \Diamond^{\beta} U_2(a_0, a_1)$ and $(\Pi_M, \mathcal{D}) \models S_i(a_1, \ell)$ for some $a_1 \in \mathrm{obj}(\mathcal{D})$ and transition $\delta(s_0, \alpha, \beta) = (s_i, \varepsilon_1, \varepsilon_2)$. If either $(\Pi_M, \mathcal{D}) \models \Diamond^* RV(a_0, \ell)$, or $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_1^{\varepsilon_1}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^{\alpha} U_1(a_0, a')$, or $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_2^{\varepsilon_2}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^{\alpha} U_2(a_0, a')$ for some $a' \in \mathrm{obj}(\mathcal{D})$, then we are done as we can construct $w(x)$ such that $\mathcal{D}, \ell \models w(a_0)$ as in the cases (*i*) and (*ii*). We assume neither of the above is the case. Observe that $\ell' \geqslant \ell$ and $\mathcal{D}, \ell' \models U_k(a_0, a_1)$ implies $\ell' = \ell$, for $k \in \{1, 2\}$. Indeed, recall that we have $\mathcal{D}, \ell \models U_k(a_0, a')$ for some $a' \in \mathrm{obj}(\mathcal{D})$. If $\mathcal{D}, \ell' \models U_k(a_0, a_1)$ for $\ell' > \ell$, then $(\Pi_M, \mathcal{D}) \models RV(a_0, \ell)$ by (5.10)-(5.11), which is a contradiction. It follows that $\alpha = \beta = 0$. Therefore, $i = 1$. We also require for the future that $\ell' \geqslant \ell$ and $\mathcal{D}, \ell' \models \exists x \, . \, U_k(a_1, x)$ implies $\ell' = 0 + \varepsilon_i$. Indeed, for the sake of contradiction, suppose there is $a' \in \mathrm{obj}(\mathcal{D})$ and $\ell' \geqslant \ell$ such that $\mathcal{D}, \ell' \models U_1(a_1, a')$ and $\ell' \neq 0 + \varepsilon_1$ (the case of $k = 2$ is analogous). Observe that in this case $\varepsilon_1 \neq -1$ because $\delta(s_0, 0, 0) = (s_1, \varepsilon_1, \varepsilon_2)$ is a transition. Suppose, first, $\varepsilon_1 = 0$, then $\ell' > \ell$. Then, since $\mathcal{D}, \ell \models U_1(a_0, a_1)$ and by (5.19), we have $(\Pi_M, \mathcal{D}) \models NE_1^{\varepsilon_1}(a_1, \ell)$, which is a contradiction. Suppose, now that $\varepsilon_1 = 1$, then $\ell' = \ell$ or $\ell' > \ell + 1$. For $\ell' = \ell$ we use (5.15) and for $\ell' > \ell + 1$ we use (5.17) to conclude that $(\Pi_M, \mathcal{D}) \models NE_1^{\varepsilon_1}(a_1, \ell)$, which is a contradiction. We conclude for $i = 1$ that

$$
\begin{aligned}
&(\Pi_M, \mathcal{D}) \models S_i(a_i, \ell) \\
&\mathcal{D}, \ell' \models \exists x \, . \, U_1(a_i, x) \text{ implies } \ell' = \ell + c_1^i, \\
&\mathcal{D}, \ell' \models \exists x \, . \, U_2(a_i, x) \text{ implies } \ell' = \ell + c_2^i,
\end{aligned} \tag{D.2}
$$

for all $\ell' \geqslant \ell$. Furthermore, we observe that there exists a partial expansion $w_i(x, x_i)$, containing (among others) variables x and x_i, such that

$$
w_i \text{ has } i + 1 \text{ variables and } \mathcal{D}, \ell \models w_i(a_0, a_i) \tag{D.3}
$$

Indeed, for $i = 1$ we take $w_i = T(x, x_i) \wedge U_1(x, x_i) \wedge U_2(x, x_i)$. That $\mathcal{D}, \ell \models w_1(a_0, a_1)$ follows from the proof above.

Suppose (D.2) and (D.3) hold for $i < m$. We will show that either there is $w(x) \in \mathit{expand}(\Pi_M, G)$ with $\leqslant m + 2$ variables such that $\mathcal{D}, \ell \models w(a_0)$, or that (D.2) and (D.3) hold with $i + 1$ in place of i. Since $(\Pi_M, \mathcal{D}) \models S_i(a_i, \ell)$, the following cases are possible (we observe that $(\Pi_M, \mathcal{D}) \models \Diamond^* RV(a_i, \ell)$ is not possible by (D.2)):

(i) either $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_1^{\varepsilon_1}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^\alpha U_1(a_i, a')$ by (5.20) or $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_2^{\varepsilon_2}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^\beta U_2(a_i, a')$ by (5.21), for some transition $\delta(s_i, \alpha, \beta) = (s_j, \varepsilon_1, \varepsilon_2)$. Assuming that (5.20) was the case (c.f. proof of (ii) above), $(\Pi_M, \mathcal{D}) \models NE_1^{\varepsilon_1}(a', \ell')$ for $\ell' > \ell$, $\alpha = +$, $\varepsilon_1 = -1$, and that $\mathcal{D}, \ell'$ satisfies the body of (5.13, left) with y equal to a' (the alternative cases are analogous), we obtain $w(x) = w_i(x, x_i) \wedge \Diamond U_1(x_i, y) \wedge \Diamond(U_1(y, z) \wedge U_1(y, x_i))$, for y, z not occurring in w_i. By (D.3), we conclude that $\mathcal{D}, \ell \models w(a_0)$ and that the number of variables in w is $\leqslant m + 2$. We are done.

(ii) $\mathcal{D}, \ell \models T(a_i, a_{i+1}) \wedge \Diamond^\alpha U_1(a_i, a_{i+1}) \wedge \Diamond^\beta U_2(a_i, a_{i+1})$ and $(\Pi_M, \mathcal{D}) \models S_j(a_{i+1}, \ell)$ for some $a_{i+1} \in \mathrm{obj}(\mathcal{D})$ and transition $\delta(s_i, \alpha, \beta) = (s_j, \varepsilon_1, \varepsilon_2)$. If either $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_1^{\varepsilon_1}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^\alpha U_1(a_i, a')$, or $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_2^{\varepsilon_2}(a', \ell)$ and $\mathcal{D}, \ell \models \Diamond^\alpha U_2(a_i, a')$ for some $a' \in \mathrm{obj}(\mathcal{D})$, then we are done as we can construct $w'(x)$ such that $\mathcal{D}, \ell \models w'(a_i)$ as in the case (i) above. We assume that neither of the above is the case. By (D.2), it follows that $\mathrm{sgn}(a_i) = \alpha$ and $\mathrm{sgn}(b_i) = \beta$, therefore $j = i + 1$. We now require for the future that $\ell' \geqslant \ell$ and $\mathcal{D}, \ell' \models \exists x U_1(a_{i+1}, x)$ implies $\ell' = \ell + a_i + \varepsilon_1$. Indeed, for the sake of contradiction, suppose there is $a' \in \mathrm{obj}(\mathcal{D})$ and $\ell' \geqslant \ell$ such that $\mathcal{D}, \ell' \models U_1(a_{i+1}, a')$ and $\ell' \neq \ell + a_i + \varepsilon_1$. Suppose, first, $\varepsilon_1 = -1$ (observe that by the construction of M, we have $a_i \geqslant 1$). It follows $\ell \leqslant \ell' < \ell + a_i - 1$ or $\ell' \geqslant \ell + a_i$. We consider the first option only (the second is left to the reader). Since $\mathcal{D}, \ell + a_i \models U_1(a_i, a_{i+1})$,

by (5.12) we obtain $(\Pi_M, \mathcal{D}) \models NE_1^{\varepsilon_1}(a_{i+1}, \ell')$ and then $(\Pi_M, \mathcal{D}) \models \Diamond^* NE_1^{\varepsilon_1}(a_{i+1}, \ell)$. This is a contradiction. We leave the cases $\varepsilon_1 \in \{0, 1\}$ to the reader. Similarly it is shown that $\ell' \geqslant \ell$ and $\mathcal{D}, \ell' \models \exists x U_2(a_{i+1}, x)$ implies $\ell' = \ell + b_i + \varepsilon_2$. Recall that $a_{i+1} = b_i + \varepsilon_1$ and $b_{i+1} = b_i + \varepsilon_2$, so we have obtained (D.2) for $i + 1$ in place of i. Furthermore, suppose $\alpha = 0$ and $\beta = +$ (the alternative three cases are left to the reader). Then consider the partial expansion $w_{i+1}(x, x_{i+1}) = w_i(x, x_i) \wedge T(x_i, x_{i+1}) \wedge U_1(x_i, x_{i+1}) \wedge \Diamond U_2(x_i, x_{i+1})$ for x_{i+1} not occurring in w_i. Since $\mathcal{D}, \ell \models w_i(a_0, a_i)$ by (D.3), then $\mathcal{D}, \ell \models w_{i+1}(a_0, a_{i+1})$ and we have also shown (D.3) for $i + 1$ in place of i.

Suppose $i = m$. By $(\Pi_M, \mathcal{D}) \models S_i(a_i, \ell)$ and (D.2), the following two cases are possible in principle: (i) above or (ii) above. Suppose (ii) is the case. By (D.2), it follows that $\mathrm{sgn}(a_i) = \alpha$ and $\mathrm{sgn}(b_i) = \beta$. So, there is a transition from $(s_m, \mathrm{sgn}(a_m), \mathrm{sgn}(b_m))$ to some $(s_j, \varepsilon_1, \varepsilon_2)$ which is a contradiction to the fact that M halts after m steps. So (i) is the case and from the proof of (i) we obtain $w(x) \in \mathit{expand}(\Pi_M, G)$ with $\leqslant m + 2$ variables such that $\mathcal{D}, \ell \models w(a_0)$.

(ii) Suppose now that M does not halt. We prove that (Π_M, G) is NL-hard by a reduction from the directed reachability problem. Let $G = (V, E)$ be a directed graph with the set of nodes $V = \{v_0, \ldots, v_n\}$, the start node v_s and the target node v_t. Let the first n steps of the computation of M (cf. (5.7)) be $(s_0, a_0, b_0), \ldots, (s_n, a_n, b_n)$. We construct a temporal database $\mathcal{D}_G$ with $\mathrm{obj}(\mathcal{D}_G) = V \times \{0, \ldots, n\}$, such that v_t is reachable from v_s in G iff $(\Pi_M, \mathcal{D}_G) \models G((v_s, 0), 0)$. Firstly, we add to $\mathcal{D}_G$ the assertions $U_1((v_t, k), (v_t, k), 0), U_1((v_t, k), (v_t, k), 1)$ for each $0 \leqslant k \leqslant n$. Then, for each $(v_i, v_j) \in E$ and $0 \leqslant k \leqslant n-1$, we add to $\mathcal{D}_G$ the assertions:

- $T((v_i, k), (v_j, k + 1), 0)$
- $U_1((v_i, k), (v_j, k + 1), a_k)$

- $U_2((v_i, k), (v_j, k+1), b_k)$.

We now show that v_t is reachable from v_s in G iff $(\Pi_M, \mathcal{D}_G) \models G((v_s, 0), 0)$. ($\Rightarrow$) Let $v_{i_0}, \ldots, v_{i_m}$ be a path in G with $i_0 = s$ and $i_m = t$. It follows by the construction of $\mathcal{D}_G$ and (5.9)-(5.11) that $\mathcal{D}_G, \Pi_M, 0 \models S_m((v_{i_m}, m))$. Then, by (5.22) it follows that $\mathcal{D}_G, \Pi_M, 0 \models S_{m-1}((v_{i_{m-1}}, m-1))$. Further applying (5.22) we obtain $\mathcal{D}_G, \Pi_M, 0 \models S_0((v_{i_0}, 0))$ and finally by (5.23) we obtain $\mathcal{D}_G, \Pi_M, 0 \models G((v_s, 0))$. The direction ($\Leftarrow$) is similar and left to the reader. □

D.3 Proofs for Section 5.4

We say that an automaton $\mathcal{A} = (S, \aleph, s^0, T, F)$, with the set of states S, input alphabet $\aleph$, initial state s^0, transition relation T, and the set of final states F, *admits reachability checking in* NL if there is a polynomial $p(x)$ and one-to-one encodings $u\colon S \to \{0,1\}^{p(\log|S|)}$ and $v\colon \aleph \to \{0,1\}^{p(\log|\aleph|)}$, such that:

> **(repr)** membership in each of the sets $u(S), \{u(s^0)\}$, $u(F)$, and $v(\aleph)$ is decidable in polynomial space;
>
> **(tran)** membership in the set $\{(u(s), v(\Lambda), u(s')) \mid (s, \Lambda, s') \in T\}$ is decidable in polynomial space.

In other words, there must be a way of checking whether a nondeterministically guessed state or transition is indeed one of $\mathcal{A}$, using space which is polynomial in the size of that guessed object.

Lemma 5.8. *For any connected datalog$^\circ_{lm}$-query (Π, G), the language Expand(Π, G) is regular.*

Proof. We show how to construct an automaton for *Expand*(π, G) of size $2^{\mathrm{poly}(|\pi|)}$ that, additionally, admits rewritability checking in NL. The correctness of an input word can be checked by a simple automaton of size $\mathcal{O}(1)$. So we will focus on an automaton that recognises words from *Expand*(π, G) among the correct words.

First, we construct a two-way automaton $\mathcal{A}_{tw} = (S_{tw}, \Omega, T_{tw}, s_{init}, s_{acc}, s_{rej})$ that simulates the rules of the program π. Recall that a two-way automaton has a head that can move along the tape in both directions and sees an input word α as $\vdash \alpha \dashv$, where $\vdash, \dashv \notin \Omega$ are the left and right end-markers. Initially, the automaton is in the state s_{init} with its head observing the first letter of α, one cell to the right of the left end-marker. The transition relation $T_{tw} \subseteq S_{tw} \times (\Omega \cup \{\vdash, \dashv\}) \times S_{tw} \times \{-1, 0, 1\}$ prescribes the movement of the head upon assuming the next state (-1 for one step to the left, 1 for a step to the right, and 0 for staying on the same cell). It is assumed that there is no transition that prescribes to move to the left of $\vdash$ or to the right of $\dashv$. The automaton accepts by assuming the state s_{acc} and moving all the way to the right end-marker, and rejects by either assuming the state s_{rej} or just never reaching s_{acc}, e.g. by looping.

We now define $\mathcal{A}_{tw}$. Let $S_{tw} = \{s_C, s'_C \mid C \in \mathit{IDB}(\Pi)\} \cup \{s_{init}, s_{acc}, s_{rej}\} \cup S_0$, with $s_{init} = s_G$. The additional states from S_0, with $|S_0| = \mathrm{poly}(|\pi|)$, will be used to perform operations such as "move the head k steps to the right" etc.

We now describe the transitions of $\mathcal{A}_{tw}$. For every recursive rule of π of the form

$$C(X) \leftarrow B(X, Y, \mathbf{U}) \wedge \bigcirc^k D(Y)$$

and every $\Lambda \in \Omega$ containing the rule body $B(X, Y, \mathbf{U}) \wedge \bigcirc^k D(Y)$, there is a transition:

$$\begin{array}{ll} (s_C, \Lambda, s_D, \mathit{right}), & \text{if } X \neq Y \\ (s_C, \Lambda, s'_D, \mathit{stay}), & \text{if } X = Y \end{array}$$

In the first of the cases above, if after the transition the head enters a horizontal segment, it should move along until it finds a letter containing $\bot$. In the second case, regardless of the observed letters, the head moves $|k|$ steps in the direction of the sign of k. When counting, the singleton $\{\top\}$ is ignored, and an attempt to go beyond the limits of the segment leads to a rejection. After this

shift, if observing a symbol Λ' such that $\top \in \Lambda'$, the automaton can nondeterministically decide to either stay or go all the way to the beginning of the next vertical segment. Then, it changes the state to s_D.

Finally, for every initialisation rule of π of the form

$$C(X) \leftarrow B(X, Y, \mathbf{U})$$

and every $\Lambda \in \Omega$ containing the rule body $B(X, Y, \mathbf{U})$, there is a transition $(s_C, \Lambda, s_{acc}, stay)$.

Note that $|S_{tw}| = \text{poly}\,(|\pi|)$. We convert $\mathcal{A}_{tw}$ to a one-way nondeterministic automaton, using the method of crossing sequences of Shepherdson Shepherdson [1959]. Since he only describes the conversion of a deterministic two-way machine to a deterministic one-way one, and we work with non-deterministic automata, we provide the whole construction.

Namely, let $\mathcal{A} = (S, \Omega, s_0, T, F)$ be a nondeterministic finite state automaton with $S = S_{tw} \times \{R \mid R \subseteq S_{tw} \times S_{tw}\}$. The idea behind the state-space is as follows. Assume $\mathcal{A}$ was given a word $\Lambda_1 \dots \Lambda_k \Lambda_{k+1} \dots \Lambda_n$, it has read the first k letters, reached some state (s_k, R_k), and is now observing the letter Λ_{k+1}. We want that $(s, s') \in R_k$ if and only if $\mathcal{A}_{tw}$, performing any available transition of the form $(s, \Lambda_{k+1}, s'', -1)$, can eventually return to Λ_{k+1} in the state s'. That is R_k is always defined deterministically for every $k \in \{0, 1, \dots, n\}$, assuming $\Lambda_0 = \vdash$, $\Lambda_{n+1} = \dashv$. Formally, let

$$R_0 = \left[T_{tw}(\cdot, \vdash, \cdot, 0)\right]^* \circ T_{tw}(\cdot, \vdash, \cdot, 1)$$
$$\tau(R, \Lambda) = \left[T_{tw}(\cdot, \Lambda, \cdot, 0) \cup R\right]^* \circ T_{tw}(\cdot, \Lambda, \cdot, 1)$$

where $P(\cdot, a, \cdot, b)$ is a binary relation obtained from a 4-ary relation P by fixing the second and the fourth arguments to the given, Q^* is a reflexive transitive closure of a binary relation Q, and $Q \circ Q'$ is a composition of two binary relations.

Then, $s_0 = \langle s_{init}, R_0 \rangle$ and T is such that

$$\left(\langle s, R \rangle, \Lambda, \langle s', R' \rangle\right) \in T \iff \begin{cases} R' = \tau(R, \Lambda) \\ (s, s') \in R' \end{cases}$$

Finally, $\langle s, R\rangle \in F$ whenever $s = s_{acc}$. We note that every relation R can be represented as an $|S_{tw}| \times |S_{tw}|$ Boolean matrix, and thus we have **(repr)**. For **(tran)**, we note that a reflexive transitive closure of such a relation or composition of two relations can be computed in polynomial space. □

Lemma 5.9. *For any connected datalog$^{\circ}_{lm}$-query (Π, G), the languages NotAccept(Π, G) and Accept(Π, G) are regular.*

Proof. Again, we construct an automaton of size $2^{\text{poly}(|\pi|)}$ that recognises *NotAccept*(Π, G) and admits rewritability checking in NL.

Let $\mathcal{D}$ be a temporal database and $a \in \text{obj}(\mathcal{D})$. We define $r(a)$ (and $-l(a)$) as the maximal (respectively, minimal) timestamp ℓ such that $\mathcal{D}_\ell$ contains an atom involving a. Clearly, $r(a), -l(a) \in \text{tem}(\mathcal{D})$ for all a. An *n-cut model* of $(\Pi, \mathcal{D})$, denoted $\mathfrak{M}_{(n)}$, is a temporal database over the joint EDB and IDB schema of Π obtained as an atomic diagram of a flat model $\mathfrak{M}$ of $\mathcal{D}$ and Π by discarding, for every object a, all the (IDB) atoms with timestamp larger than $r(a) + n$ and smaller than $-l(a) - n$. Intuitively, $\mathfrak{M}_{(n)}$ preserves all information about $\mathfrak{M}$ that is 'reachable' from $\mathcal{D}$ via a path of length n. The following lemma allows to focus on finite temporal databases instead of infinite models.

Lemma D.1. *Let $|\Pi| = n$. Then $(\Pi, \mathcal{D}) \models G(a_1, \dots, a_k, \ell)$ if and only if $G(a_1, \dots, a_k, \ell) \in \mathfrak{M}_{(n)}$ for every flat model $\mathfrak{M}$ of $\mathcal{D}$ and Π.*

Proof. The direction $(\Rightarrow)$ is trivial, since $\mathfrak{M}_{(n)}$ is obtained from a model of $\mathcal{D}$ and Π. For the other direction, let $\mathfrak{M}$ be an arbitrary flat model of $\mathcal{D}$ and Π and consider $\mathfrak{M}_{(n)}$. Since $G(a_1, \dots, a_k, \ell) \in \mathfrak{M}_{(n)}$, it must be that $\mathfrak{M}, \ell \models G(a_1, \dots, a_k, \ell)$. Since $\mathfrak{M}$ is arbitrary, we get $(\Pi, \mathcal{D}) \models G(a_1, \dots, a_k, \ell)$. □

Roughly speaking, the states of the automaton for *NotAccept*(Π, G) will be formed by the $|\Pi|$-cut models. We will encode them as words over an enriched version of Ω. The definition is as follows. Given a rule body B of Π, its *enrichment* is obtained by adding to B various atoms of the form $\bigcirc^k D(y)$,

where d is an IDB, y is a variable of B and $|k| \leqslant |\Pi|$. Moreover, we define $\Gamma_{\Pi,e}$ as the set of all enrichments of rule bodies of Π and $\Omega_e = 2^{\Gamma_{\Pi,e} \cup \{\bot,\top\}}$. An enrichment of a word $\alpha \in \Omega^*$ is a word $\alpha_e \in \Omega_e^*$ obtained by substituting each of rule bodies in α with one of its enrichments. Given an enriched word α_e, we build a temporal database $\mathcal{D}_{\alpha_e}$ just like we did it for non-enriched words. The schema of $\mathcal{D}_{\alpha_e}$ is thus $\mathit{EDB}(\Pi) \cup \mathit{IDB}(\Pi)$. We call α_e *legal* if $\mathcal{D}_{\alpha_e} = \mathfrak{M}_{(n)}$, where $n = |\Pi|$, for some model $\mathfrak{M}$ of Π and $\mathcal{D}_\alpha$.

Lemma D.2. *Given α_e, its legality can be decided in* PSPACE.

Proof. Recall that the objects of $\mathcal{D}_{\alpha_e}$ have the form a_y for y a variable of α. By the definition of enrichments, α_e provides information on IDB atoms at timestamps from $-l(a_y) - |\Pi|$ to $r(y) + |\Pi|$, for every $a_y \in \mathrm{obj}(\mathcal{D}_{\alpha_e})$. Let T_{a_y}, or the *timeline* of a_y, be the temporal database $\langle T_\ell^{a_y} \mid -l(a_y) - |\Pi| \leqslant \ell \leqslant r(a_y) + |\Pi| \rangle$ where T_ℓ contains exactly the IDB atoms of a_y present in $\mathcal{D}_{\alpha_e}$ at time ℓ. We say that T_{a_y} can be extended to an *LTL*-model under the rules of Π if there exists a model of T_{a_y} and the set of horizontal rules of Π, that coincides with T_{a_y} for $\ell \in \mathrm{tem}(T_{a_y})$.

Our goal is to decide whether there exists a flat model $\mathfrak{M}$ of $\mathcal{D}$ and Π such that for $n = |\Pi|$ we have $\mathcal{D}_{\alpha_e} = \mathfrak{M}_{(n)}$. The critical observation is that the rules of Π whose bodies involve EDB atoms can only be used to infer IDB atoms at timestamps within the distance of at most $|\Pi|$ from the borders of $\mathrm{tem}(\mathcal{D})$. Thus, the condition $\mathcal{D}_{\alpha_e} = \mathfrak{M}_{(n)}$ is equivalent to the conjunction of the following two statements:

1. $\mathcal{D}_{\alpha_e}$ satisfies all rules of Π;
2. for every $a_y \in \Delta_{\mathcal{D}_{\alpha_e}}$, its timeline T_{a_y} can be extended to an *LTL*-model under the horizontal rules of Π.

A direct check of the first condition fits into PSPACE. For the second condition, we use the standard *LTL* satisfiability checking algorithm Manna and Pnueli [1991], also working in polynomial space. □

Let us return to proving Lemma 5.9 and build the respective automaton $\mathcal{A}_{na}$ for $\mathit{NotAccept}(\Pi, G)$. A word α is in $\mathit{NotAccept}(\Pi, G)$ if either it is not correct, or $(\Pi, \mathcal{D}_\alpha) \not\models G(a_x, 0)$. The first condition can be checked by a simple automaton of size $\mathcal{O}(1)$. The second condition, by Lemma D.1, means that there exists a $|\Pi|$-cut model $\mathfrak{M}_{(n)}$ of $\mathcal{D}_\alpha$ and Π such that $G(a_x, 0) \notin \mathfrak{M}_{(n)}$. The idea, adapted from Cosmadakis et al. [1988], is to construct a nondeterministic automaton, that, given α, verifies that it satisfies the second condition. Namely, it guesses an enrichment α_e and checks that: (i) it is legal, (ii) it does not contain $G(x, 0)$.

The difficult part is (i). By Lemma D.2, the legality of α_e can be checked in polynomial space. However, we need to do it via a finite state automaton, i.e. in constant space. In doing this, we rely on the connectedness of Π. Intuitively, α_e is legal if all timelines can be extended to *LTL*-models and, moreover, every 'neighbourhood' in $\mathcal{D}_{\alpha_e}$ of 'size' $|\Pi|$ satisfies the rules of Π. We now give a precise definition to the notion of 'neighbourhood'. Let $\alpha \in \Omega^*$ be a correct word, and $\alpha_e = \Lambda_0 \dots \Lambda_n$, $\Lambda_i \in \Omega_e$, be its enrichment. We define an undirected graph $G_{\alpha_e} = (V_{\alpha_e}, E_{\alpha_e})$ so that $V_{\alpha_e} = \{i \mid \Lambda_i \text{ letter of } \alpha_e\}$, and E_{α_e} is as follows. Respective nodes are connected by an edge for any two consecutive letters of the same vertical or horizontal segment. If x_j is a vertical segment and y_j is a horizontal segment that follows x_j, then the node of the last letter of x_j is connected to that of the letter of y_j that is marked by $\bot$. Similarly, the node of the letter of y_j that is marked by $\top$ is connected with that of the first letter of the subsequent vertical segment x_{j+1}. The *distance* between two letters of α_e is that between the corresponding nodes in G_{α_e}, and the *d-neighbourhood* of the letter Λ_i in α_e, for $d \in \mathbb{N}$, is the word $N_d(i, \alpha_e)$ obtained from α_e by omitting all the letters that are at distance more than d from Λ_i. We further define the *left d-neighbourhood* of Λ_i as its d-neighbourhood in the word $\Lambda_0 \dots \Lambda_i$. The key observation is the following, given the connectedness of Π.

Lemma D.3. *An enrichment α_e is legal if and only if for every letter Λ_i in α_e, its neighbourhood $N_d(i, \alpha_e)$, $d = |\Pi|$, is legal as an*

enriched word over Ω_e.

We further observe that for every letter Λ_i, the neighbourhood $N_d(i, \alpha_e)$ is contained in the left $2d$-neighbourhood of the rightmost letter of α_e that belongs to $N_d(i, \alpha_e)$. Thus, reading α_e left to right, it is enough for $\mathcal{A}_{na}$ to ensure that every left $2d$-neighbourhood is legal for $d = |\Pi|$. To do so, the automaton can remember in its state, at every moment, the left $2d$-neighbourhood of the current letter and reject once it encounters a non-legal one. When reading a vertical segment, it suffices to remember the left neighbourhood of the previous letter and update it accordingly for the current one. In the case of reading a horizontal segment, left to right, the left neighbourhood changes substantially once we pass over the symbol $\perp$. To handle this, $\mathcal{A}_{na}$ must know both the left neighbourhood of the previous letter, as well as that of the last letter of the preceding vertical segment. Both can be described by the respective words over Ω_e.

It remains to show that $\mathcal{A}_{na}$ admits rewritability checking in NL. Clearly, every enriched letter $\Lambda \in \Omega_e$ can be encoded by a binary sequence of polynomial length. The word of enriched letters that describes a left $2d$-neighbourhood, for $d = |\Pi|$, is thus also representable in space poly $(|\Pi|)$. Then the property **(repr)** is given by Lemma D.2, and that of **(tran)** is given by construction of $\mathcal{A}_{na}$. This proves Lemma 5.9. □

D.4 Proofs for Section 5.5

Lemma 5.11. *If a query is vertically unbounded, then it is* L*-hard.*

Proof. Since $\mathit{Expand}(\Pi, G) \subseteq \mathit{Accept}(\Pi, G)$, and (Π, G) is vertically unbounded, for any k there is a word $\alpha \in \mathit{Accept}(\Pi, G)$, $\mathrm{height}(\alpha) > k$, such that the longest prefix of α of height k is in $\mathit{NotAccept}(\Pi, G)$. Let $\mathcal{A}_{\Pi,G}$ be the deterministic automaton that recognises $\mathit{Accept}(\Pi, G)$ (Lemma 5.9). By flipping its final states we obtain a deterministic automaton for $\mathit{NotAccept}(\Pi, G)$ with exactly

the same set of states and transition relation. For any k, let β_k, γ_k be the words such that $\text{height}(\beta_k) = k, \text{height}(\gamma_k) > 0$, the first segment of γ_k is vertical, $\beta_k \in \mathit{NotAccept}(\Pi, G)$, and $\beta_k\gamma_k \in \mathit{Accept}(\Pi, G)$. Furthermore, let s_k^m be the state of $\mathcal{A}_{\Pi,G}$ that is reached upon reading the mth vertical symbol of β_k. If k is sufficiently larger than the number of states of $\mathcal{A}_{\Pi,G}$, then there must be a repetition in the sequence $s_k^1, \dots, s_k^{k-1}$. Let i, j be such that $s_k^i = s_k^j$, and let ξ be the part of β_k from the beginning till the i-th vertical symbol, υ the part from that point till the j-th vertical symbol, and ζ—the remaining part. Then these words have the following properties:

1. $\text{height}(\xi\upsilon) < k, \text{height}(\upsilon) > 0$;
2. for any $i \geqslant 0$ we have $\xi\upsilon^i\zeta \in \mathit{NotAccept}(\Pi, G)$ and $\xi\upsilon^i\zeta\gamma \in \mathit{Accept}(\Pi, G)$;
3. words ξ, υ, and $\zeta\gamma$ are correct.

The latter property is ensured by the fact that the last symbols of ξ and υ are vertical, so we never 'cut' inside a horizontal segment.

For a correct word α, $\mathcal{D}_\alpha$ always contains the object a_x corresponding to the answer variable of the temporal CQ $\alpha(x)$. If α contains an IDB atom $D(y)$, we denote that y by y_α and, moreover, we write ℓ_α for the timestamp at which $D'(y_\alpha)$ would appear in $\mathcal{D}_\alpha$ if we substituted D with a new EDB predicate D'. Finally, let $\text{width}(\alpha) = |\text{tem}(\mathcal{D}_\alpha)|$.

Now we are ready to give a reduction from the undirected reachability problem. Let $\mathcal{G} = (V, E)$ be an undirected graph and $s, p \in V$. We build a temporal database $\mathcal{D}_\mathcal{G}$ with $s_0 \in \text{obj}(\mathcal{D}_\mathcal{G})$, so that $(\Pi, \mathcal{D}_\mathcal{G}) \models G(s_0, 0)$ if and only if s and p are connected by a path in $\mathcal{G}$. Assume w.l.o.g. that $(v, v) \in E$ for every $v \in V$ and let $V_i = \{v_i \mid v \in V\}$ be a copy of V indexed by $i \in \mathbb{N}$. The domain of $\mathcal{D}_\mathcal{G}$ will be the union of V_i for $0 \leqslant i < |V|$, plus some auxiliary objects. First, add a copy of $\mathcal{D}_\xi$ to $\mathcal{D}_\mathcal{G}$ so that y_ξ at time ℓ_ξ in $\mathcal{D}_\xi$ is glued to s_0 at time 0 in $\mathcal{D}_\mathcal{G}$. In the same fashion, for every $(u, v) \in E$ add a copy of $\mathcal{D}_\upsilon$, gluing x at 0 to u_i at $i \cdot \text{width}(\upsilon)$ and y_υ at ℓ_υ to

v_{i+1} at $(i+1)\cdot \mathrm{width}(\upsilon)$, for every i from 0 to $|V|-2$. Do the same, connecting v_i and u_{i+1}. Finally, add a copy of $\mathcal{D}_{\zeta\gamma}$ to $\mathcal{D}_{\mathcal{G}}$ so that x at 0 of $\mathcal{D}_{\zeta\gamma}$ is glued to $p_{|V|-1}$ at time $(|V|-1)\cdot \mathrm{width}(\upsilon)$ of $\mathcal{D}_{\mathcal{G}}$.

It is not hard to see that if s and p are connected in $\mathcal{G}$, then $(\Pi, \mathcal{D}_{\mathcal{G}}) \models G(s_0, 0)$. For the other direction, note that $(\Pi, \mathcal{D}_{\mathcal{G}}) \models G(s_0, 0)$ means there is $w \in \mathit{expand}(\Pi, G)$ and a homomorphism h from $\mathcal{D}_w$ to $\mathcal{D}_{\mathcal{G}}$ that maps a_x to s_0. If there exists a variable z of w such that a_z is mapped to $p_{|V|-1}$ (at any moment of time), we are done, since Π is connected. Otherwise, if no variable is mapped to $p_{|V|-1}$, and hence to any node of the copy of $\mathcal{D}_{\zeta\gamma}$ attached to it, we note that, by construction, there exists an m and a homomorphism g from $\mathcal{D}_w$ to $\mathcal{D}_{\xi\upsilon^m}$ with $g(a_x) = s_0$. Thus, $\xi\upsilon^m$ would be in $\mathit{Accept}(\Pi, G)$, a contradiction. □

Lemma 5.12. *If (Π, G) is vertically bounded, then the data complexity of answering (Π, G) coincides with that of checking membership in $\mathit{Accept}(\Pi, G)$, modulo reductions computable by constant-depth circuits.*

Proof. We first give a reduction from the decision problem for $\mathit{Accept}(\Pi, G)$ to answering (Π, G). Given a word $\alpha \in \Omega^*$, checking that it is correct can be done in AC^0. For a correct word, construct $\mathcal{D}_\alpha$. Then, by definition, $(\Pi, \mathcal{D}_\alpha) \models G(x, 0)$ if and only if $\alpha \in \mathit{Accept}(\Pi, G)$.

The reduction from answering (Π, G) to deciding the language $\mathit{Accept}(\Pi, G)$ is more involved. W.l.o.g. we will consider answering the query for $\ell = 0$. Recall that $(\Pi, \mathcal{D}) \models G(a, 0)$, for some $\mathcal{D}, a, 0$, if and only if there is $\alpha \in \mathit{Expand}(\Pi, G)$ and a homomorphism h from $\mathcal{D}_\alpha$ to $\mathcal{D}$ with $h(a_x) = a$. Since (Π, G) is vertically bounded, by definition there exists a number k, uniformly for all expansions, such that the longest prefix of α of height k is in $\mathit{Accept}(\Pi, G)$. Then we arrive to the following characterisation: $(\mathcal{D}, \Pi) \models G(a, 0)$ if and only if there exists a tuple $\langle \mu_1, \mu_2, \ldots, \mu_n \rangle$ of words in Ω composed of vertical letters and such that $|\mu_1| + \cdots + |\mu_n| \leqslant k$, and a tuple $\langle \upsilon_1, \upsilon_2, \ldots, \upsilon_n \rangle$ of words composed of horizontal letters, so

that the word $\alpha = \mu_1\upsilon_1 \dots \mu_n\upsilon_n$ belongs to $\mathit{Accept}(\Pi, G)$ and there is a homomorphism $\mathcal{D}_\alpha \to \mathcal{D}$ which maps a_x to a. We observe that there is a finite number of the tuples of the form $\langle \mu_1, \mu_2, \dots, \mu_n \rangle$ with $|\mu_1| + \dots + |\mu_n| \leqslant k$. We further note that $\mathit{Accept}(\Pi, G, k)$, by definition, is closed upwards under the relation $\preccurlyeq$: if $\alpha \preccurlyeq \beta$, then $\beta \in \mathit{Accept}(\Pi, G, k)$. Thus, checking $(\mathcal{D}, \Pi) \models G(a, 0)$ is reduced to a search in $\mathcal{D}$ for a tuple $\langle \mu_1, \dots, \mu_n \rangle$ of vertical words of joint height k and a $\preccurlyeq$-maximal tuple $\langle \upsilon_1, \upsilon_2, \dots, \upsilon_n \rangle$ of horizontal words such that $\mu_1\upsilon_1 \dots \mu_n\upsilon_n \in \mathit{Accept}(\Pi, G)$. The search can be done in AC^0, so the complexity of the whole procedure is determined by that of the decision problem for $\mathit{Accept}(\Pi, G)$. □

Lemma 5.13. *Checking if a connected datalog$^\circ_{lm}$ query is vertically bounded is in* PSPACE.

Proof. We take the automaton $\mathcal{A}_{na}$ for $\mathit{NotAccept}(\Pi, G)$, constructed in Lemma 5.9, and the (one-way) automaton $\mathcal{A}_e$ for $\mathit{Expand}(\Pi, G)$ from Lemma 5.8. Let $\mathcal{A}_{vb}$ be their Cartesian product. Then $\mathcal{A}_{vb}$ is $|\Pi|$-manipulatable. The condition that we need to check on $\mathcal{A}_{vb}$ is the following:

> $(*)$ For every k there is a word $\beta \in \mathit{NotAccept}(\Pi, G)$, $\mathrm{height}(\beta) = k$, and a word γ, $\mathrm{height}(\gamma) > 0$, such that the first letter of γ is vertical and $\beta\gamma \in \mathit{Expand}(\Pi, G)$.

In $\mathcal{A}_{vb}$, call a transition *vertical* if it is performed upon reading a vertical body and *horizontal* otherwise. Then $(*)$ is true if and only if there is a state q of $\mathcal{A}_{vb}$ that is accepting for $\mathcal{A}_{na}$ and that has two features:

1. There is a path from the initial state of $\mathcal{A}_{vb}$ to q that contains a cycle such that there is at least one vertical transition in that cycle.

2. There is a state q' and a vertical transition from q to q', and a path from q' to an accepting state of $\mathcal{A}_e$.

Due to $|\Pi|$-manipulatability, looking for paths and cycles can be done on the fly in nondeterministic space polynomial in $|\Pi|$. Since NPSPACE = PSPACE, we are done. □

Lemma 5.14. *Deciding boundedness of connected linear monadic queries in plain datalog is* PSPACE*-hard.*

Proof. Let p be a polynomial and M a Turing machine that, given an input y, works in space $p(|y|)$. Let $\mathcal{S}$ denote the set of states of M and Γ—its tape alphabet. Let x be a word in the input alphabet of M. In the computation of M on x, encode every configuration by a word over the alphabet $\Gamma_0 = (\Gamma \cup (\Gamma \times \mathcal{S}))^{p(|x|)}$ in the standard way, and the computation itself as a concatenation of these words separated by the symbol $\# \notin \Gamma$.

Introduce a unary EDB relation Q_a for every symbol $a \in \Gamma_0^{\#} = \Gamma_0 \cup \{\#\}$, a unary EDB relation *First* and a binary EDB relation *Next* to encode words over $\Gamma_0^{\#}$. For example, the word *abc* is encoded as $\mathit{First}(a), Q_a(a), \mathit{Next}(a, a'), Q_b(a'), \mathit{Next}(a', a''), Q_c(a'')$.

We define a connected linear monadic program $\Pi_{M,x}$ with two IDBs G (as usual, for *Goal*) and F (for *Finger*), so that (Π, G) is bounded if and only if M accepts x. We note that $(\Pi_{M,x}, F)$ will be always unbounded and thus we are talking about query boundedness, but not program boundedness.

Assume without loss of generality that M halts on x if and only if it accepts. The idea is to let F start from the *First* symbol in an encoding of M's computation on x and to slide along this encoding until a wrong transition is found. The idea is that all configurations have the same length $p(|x|)$. If d represents a tape cell in the current configuration, we can access the object that represents the same cell in the next one by a path of length $p(|x|)$ along the relation *Next*.

First, we derive G immediately if the first configuration is not the initial one, i.e. not the one that has the word x followed by $p(|x|)-|x|$ blank symbols, and then a #, with the machine observing the first

symbol of x:

$$G(x) \leftarrow \mathit{Next}(x, x_1) \wedge \mathit{Next}(x_1, x_2) \wedge \cdots \wedge \mathit{Next}(x_{k-1}, x_k) \wedge Q_a(x_k), \tag{D.4}$$

for every k from 0 to $p(|x|)$ and every $a \in \Gamma_0^{\#}$ which should not be in the kth place of the initial configuration.

Having ensured that we are observing the initial configuration, we begin the slide:

$$G(x) \leftarrow \mathit{First}(x) \wedge F(x). \tag{D.5}$$

Recall that there exists a 6-ary relation R such that if $u = a_1 \dots a_{p(|x|)}$ is a configuration encoding and $v = b_1 \dots b_{p(|x|)}, b_i \in \Gamma_0$, then v is an encoding of the subsequent configuration of u if and only if $(a_i, a_{i+1}, a_{i+2}, b_i, b_{i+1}, b_{i+2}) \in R$, $1 \leqslant i \leqslant p(|x|) - 2$. Let $R^{\#} = R \cup \{\#\} \times \Gamma_0^2 \cup \Gamma_0^2 \times \{\#\}$. We infer $F(x)$ every time there is an outgoing path of *Next* such that the three letters at x and those at distance $p(|x|)$ on this path do not belong to $R^{\#}$:

$$\begin{aligned} &F(x) \leftarrow \mathit{Next}(x, x_1) \wedge \cdots \wedge \mathit{Next}(x_{p(|x|)+1}, x_{p(|x|)+2}), \\ &Q_a(x) \wedge Q_b(x_1) \wedge Q_c(x_2) \wedge Q_d(x_{p(|x|)}) \wedge Q_e(x_{p(|x|)+1}) \wedge Q_f(x_{p(|x|)+2}), \end{aligned} \tag{D.6}$$

for all tuples $(a, b, c, d, e, f) \notin R^{\#}$. Furthermore, we infer F if there are more than one letter in a tape cell:

$$F(x) \leftarrow Q_a(x) \wedge Q_b(x), \text{ for } a \neq b. \tag{D.7}$$

Finally, we move F forward along *Next*, provided that the objects are labelled with some letters at all:

$$F(x) \leftarrow Q_a(x) \wedge \mathit{Next}(x, y) \wedge Q_b(y) \wedge F(y), \tag{D.8}$$

for all $a, b \in \Gamma_0^{\#}$.

Given a database D over the EDB schema $\Sigma = \{Q_a \mid a \in \Gamma_0^{\#}\} \cup \{\mathit{First}, \mathit{Next}\}$, we call a tuple $(a_0, \dots, a_{k-1}) \in \mathrm{obj}(D)^k$ a *Next-path*

of length k if $\mathit{Next}(a_0, a_1), \ldots, \mathit{Next}(a_{k-2}, a_{k-1}) \in D$. We say that it is *labeled by* a word $u \in (\Gamma_0^{\#})^k$ if $u = a_0 \ldots a_{k-1}$ and $Q_{a_i}(a_i) \in D$. Note that a *Next*-path may be labeled by more than one word.

Let u_{init} be the encoding of the initial configuration, $\Pi_{M,x}$ be the set of rules defined in (D.4)—(D.8), and D a database over Σ with $a_0 \in \mathrm{obj}(D)$. Suppose, $(\Pi, D) \models G(a_0)$. There are two possible cases. The first is that, by rule (D.4), there exists a *Next*-path of length at most $p(|x|) + 1$ starting at a_0 and labeled by a word u which is *not* a prefix of $u_{\mathit{init}}\#$. Otherwise, $\mathit{First}(a_0) \in D$ and $(\Pi, D) \models F(a_0)$. In the latter case there is $w \in \mathit{expand}(\Pi, F)$ such that $D \models w(a_0)$. Let w be the shortest expansion with this property. Then, from the form of the expansions of (Π, F) we conclude that no prefix of w is in $\mathit{accept}(\Pi, F)$. Let $x, x_1, \ldots, x_{k-1}, x_k$ be the variables of w in the order of appearance and h be the homomorphism from D_w to D, with $h(x) = a_0$ and $h(x_i) = a_i$, $1 \leqslant i \leqslant k-1$. Then $(a_0, \ldots, a_{k-1})$ is a *Next*-path in D which is labeled by a unique word u. Let u' be the prefix of u ending at the last # in u. Then u' is a prefix of an encoding of the full computation of M on x. The fact that it is a prefix of such is ensured by that any *Next*-path of length $p(|x|)$ starting in a_0 is labeled by an encoding of the initial configuration.

Thus, if M halts on x there is only finitely many possible u' and (Π, G) is bounded. Otherwise, it is unbounded. □

Proposition 5.15. *For every connected* $\mathrm{datalog}^{\circ}_{lm}$ *query* (Π, G) *there exist a datalog query* (Π_d, G) *and LTL query* (Π_t, G)*, such that:*

1. (Π, G) *is vertically bounded if and only if* (Π_d, G) *is bounded;*
2. *if* (Π, G) *is vertically bounded then the data complexity of answering it coincides with that for* (Π_t, G)*.*

Proof. Fix a linear connected query (Π, G). By Lemma 5.9, there is a deterministic finite state automaton, denoted $\mathcal{A}_{\Pi,G} =$

$(S, \Omega, s_0, \delta, F)$, that recognises the language $\mathit{Accept}(\Pi, G)$. We introduce an IDB S_i for each $s_i \in S$. Moreover, for every $\Lambda \in \Omega$ we have a unary EDB relation Λ and an additional unary EDB *End* to mark the end of the word. Then Π_t is a program composed of the following rules:

$$G(x) \leftarrow S_0(x), \text{ for the initial state } s_0 \tag{D.9}$$

$$S_i(x) \leftarrow \Lambda(x) \wedge \bigcirc S_j(x), \text{ for every transition } \delta(s_i, \Lambda) = s_j \tag{D.10}$$

$$S_i(x) \leftarrow \mathit{End}(x), \text{ for every } s_i \in F \tag{D.11}$$

$$S_i(x) \leftarrow \Lambda(x) \wedge \Lambda'(x), \text{ for every } s_i \in S \text{ and } \Lambda \neq \Lambda'. \tag{D.12}$$

By construction, $(\Pi_t, \mathcal{D}) \models G(a, \ell)$ if and only if there exists $\ell' \geqslant \ell$ such that:

1. for all $l, \ell \leqslant l < \ell'$, there is $\Lambda_l \in \Omega$ such that $\Lambda_l(a, l) \in \mathcal{D}$;
2. either there is $l, \ell \leqslant l < \ell'$, such that $\Lambda(a), \Lambda'(a, l) \in \mathcal{D}$ for two different Λ, Λ', or $\mathit{End}(a, \ell) \in \mathcal{D}$ and the word $\Lambda_\ell \Lambda_{\ell+1} \dots \Lambda_{\ell'-1}$ belongs to $\mathit{Accept}(\Pi, G)$.

The first property is checkable in AC^0, and the second coincides with the decision problem for $\mathit{Accept}(\Pi, G)$.

For Π_d, we introduce a binary EDB $\Lambda(x, y)$ for every vertical letter of Ω, and skip the horizontal letters. The rules are as follows:

$$G(x) \leftarrow S_0(x), \text{ for the initial state } s_0 \tag{D.13}$$

$$S_i(x) \leftarrow \Lambda(x, y) \wedge S_j(y), \text{ for every transition } \delta(s_i, \Lambda) = s_j \text{ with a vertical letter } \Lambda \tag{D.14}$$

$$S_i(x) \leftarrow \mathit{True}, \text{ for every } s_i \in F \tag{D.15}$$

$$S_i(x) \leftarrow S_j(y), \text{ whenever } \delta(s_i, \alpha) = s_j \tag{D.16}$$

for every word α composed of horizontal letters.

It remains to prove that (Π_d, G) is bounded whenever (Π, G) is vertically bounded. So assume that (Π, G) is vertically unbounded. Then for every k there is $\alpha \in \mathit{Expand}(\Pi, G) \subseteq \mathit{Accept}(\Pi, G)$, such

that height(α) $> k$ and if β is a prefix of α of height(k) then $\beta \in$ *NotAccept*(Π, G). Let $\rho \in$ *expand*(Π_d, G) correspond to the run of $\mathcal{A}_{\Pi,G}$ on α and ρ' be its prefix corresponding to that on β. By construction, $\rho' \in$ *notaccept*(Π_d, G), and, moreover, $|\rho| > k$ and $|\rho'| = k$. Since k is arbitrary, (Π_d, G) is unbounded.

For the other direction, let (Π_d, G) be unbounded and $\rho \in$ *expand*(Π_d, G) be such that its prefix of length k is in *notaccept*(Π_d, G). Take the $\alpha \in$ *Accept*(Π, G) that corresponds to the run of $\mathcal{A}_{\Pi,G}$ represented by ρ. Let β be the longest prefix of α of height k and ρ' be the prefix of ρ that corresponds to the run of $\mathcal{A}_{\Pi,G}$ on β. Then $|\rho'| = k$, and thus the related run is not accepting, hence $\beta \in$ *NotAccept*(Π, G). Thus, for every k we found a word $\alpha_k \in$ *Accept*(Π, G) of height greater than k, so that any prefix of α_k of height k is in *NotAccept*(Π, G). To finish the proof we need to show that such an α_k can be found in *Expand*(Π, G). Since $\alpha_k \in$ *Accept*(Π, G), there is $\gamma \in$ *Expand*(Π, G) and a homomorphism from $\mathcal{D}_\gamma$ to $\mathcal{D}_{\alpha_k}$ mapping a_x to a_x. Take $k = md + 1$, where $m > 0$ and d is the diameter of Π. Then any γ of height less then m would be mapped fully if the prefix of α_k of height k. Since the latter is in *NotAccept*(Π, G), the height of γ is greater than m. By the same line reasoning, the prefix of γ of height m is itself in *NotAccept*(Π, G). Since m is arbitrary, we are done. □

Bibliography

Martín Abadi and Zohar Manna. Temporal logic programming. *J. Symb. Comput.*, 1989. doi: 10.1016/S0747-7171(89)80070-7.

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. 1995. URL dl.acm.org/doi/10.5555/551350.

Aristotle. *De interpretatione*. URL en.wikipedia.org/wiki/On_Interpretation.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. 2009. URL dl.acm.org/doi/10.5555/1540612.

Alessandro Artale and Enrico Franconi. Temporal description logics. In *Handbook of Temporal Reasoning in Artificial Intelligence*, 2005. URL api.semanticscholar.org/CorpusID:16929398.

Alessandro Artale, Roman Kontchakov, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporalising tractable description logics. In *Proc. TIME*, 2007. doi: 10.1109/TIME.2007.62.

Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *J. Artif. Int. Res.*, 2009. URL dl.acm.org/doi/10.5555/1734953.1734954.

Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. The complexity of clausal fragments of

LTL. In *Logic for Programming, Artificial Intelligence, and Reasoning*, 2013a. doi: 10.1007/978-3-642-45221-5_3.

Alessandro Artale, Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev. Temporal description logic for ontology-based data access. In *Proc. IJCAI*, 2013b. URL `aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6824`.

Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. A cookbook for temporal conceptual data modelling with description logics. *ACM Trans. Comput. Log.*, 2014. doi: 10.1145/2629565.

Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Ontology-mediated query answering over temporal data: a survey. In *Proc. TIME*, 2017. doi: 10.4230/LIPIcs.TIME.2017.1.

Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. First-order rewritability of ontology-mediated queries in linear temporal logic. *Artif. Intell.*, 2021. doi: 10.1016/j.artint.2021.103536.

Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. First-order rewritability and complexity of two-dimensional temporal ontology-mediated queries. *J. Artif. Intell. Res.*, 2022. doi: 10.1613/JAIR.1.13511.

Alessandro Artale, Anton Gnatenko, Vladislav Ryzhikov, and Michael Zakharyaschev. A decidable temporal dl-lite logic with undecidable first-order and datalog-rewritability of ontology-mediated atomic queries (extended abstract). In *Proc. DL*, 2023. URL `ceur-ws.org/Vol-3515/abstract-3.pdf`.

Alessandro Artale, Anton Gnatenko, Vladislav Ryzhikov, and Michael Zakharyaschev. On deciding the data complexity of an-

swering linear monadic datalog queries with LTL operators. In *Proc. ICDT*, 2025. doi: 10.4230/LIPICS.ICDT.2025.31.

Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI*, 2005. URL `dl.acm.org/doi/10.5555/1642293.1642351`.

Franz Baader, Carsten Lutz, and Sebastian Brandt. Pushing the $\mathcal{EL}$ envelope further. In *OWLED*, 2008. URL `ceur-ws.org/Vol-496/owled2008dc_paper_3.pdf`.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The description logic handbook: theory, implementation and applications*. 2010. doi: 10.1017/CBO9780511711787.

Franz Baader, Stefan Borgwardt, and Marcel Lippmann. Temporalizing ontology-based data access. In *Proc. CADE*, 2013. doi: 10.1007/978-3-642-38574-2_23.

Marianne Baudinet. Temporal logic programming is complete and expressive. In *Proc. POPL*, 1989. doi: 10.1145/75277.75301.

Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *Proc. LICS*, 2015. doi: 10.1109/LICS.2015.36.

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.*, 2014. doi: 10.1145/2661643.

Camille Bourgaux, Anton Gnatenko, and Michaël Thomazo. Analysing temporal reasoning in description logics using formal grammars. In *Proc. ECAI*, 2025. doi: 10.3233/FAIA251004.

Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. On expressiveness and complexity in real-time model checking. In *Proc. ICALP*, 2008. doi: 10.1007/978-3-540-70583-3_11.

Ronald Brachman and Hector Levesque. *Knowledge representation and reasoning*. 2004. doi: dl.acm.org/doi/10.5555/2821570.

Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. Querying log data with metric temporal logic. *J. Artif. Int. Res.*, 2018. doi: 10.1613/jair.1.11229.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 2013. doi: 10.1016/J.ARTINT.2012.10.003.

Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.*, 1989. doi: 10.1109/69.43410.

Stefano Ceri, Georg Gottlob, and Letizia Tanca. Logic programming and databases. In *Surveys in Computer Science*, 1990. doi: 10.1007/978-3-642-83952-8.

Jan Chomicki. Polynomial time query processing in temporal deductive databases. In *Proc. PODS*, 1990. doi: 10.1145/298514.298589.

Jan Chomicki. Temporal query languages: a survey. In *Temporal Logic*, 1994. doi: 10.1007/BFb0014006.

Jan Chomicki and Tomasz Imieliński. Temporal deductive databases and infinite objects. In *Proc. PODS*, 1988. doi: 10.1145/308386.308416.

Jan Chomicki and David Toman. Temporal logic in information systems. In *Logics for Databases and Information Systems*. 1998. doi: 10.1007/978-1-4615-5643-5_3.

Jan Chomicki, David Toman, and Michael H. Böhlen. Querying ATSQL databases with temporal logic. *ACM Trans. Database Syst.*, 2001. doi: 10.1145/383891.383892.

Noam Chomsky. Three models for the description of language. *IRE Trans. Inf. Theory*, 1956. doi: 10.1109/TIT.1956.1056813.

Kevin J. Compton and Claude Laflamme. An algebra and a logic for nc1. *Inf. Comput.*, 1990. doi: 10.1016/0890-5401(90)90063-N.

Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Proc. STOC*, 1988. doi: 10.1145/62212.62259.

Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 2001. doi: 10.1145/502807.502810.

John Deacon and Freddie Mercury. Friends will be friends, 1986. URL `en.wikipedia.org/wiki/Friends_Will_Be_Friends`.

Stéphane Demri, Valentin Goranko, and Martin Lange. *Temporal logics in computer science: finite-state systems*. 2016. doi: 10.1017/CBO9781139236119.

Mirko Michele Dimartino, Andrea Calì, Alexandra Poulovassilis, and Peter T. Wood. Query rewriting under linear $\mathcal{EL}$ knowledge bases. In *Proc. RR*, 2016. doi: 10.1007/978-3-319-45276-0_6.

H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical logic*. 1994. doi: 10.1007/978-1-4757-2355-7.

Javier Esparza, Pierre Ganty, Stefan Kiefer, and Michael Luttenberger. Parikh's theorem: a simple and direct automaton construction. *Inf. Process. Lett.*, 2011. doi: 10.1016/j.ipl.2011.03.019.

John Etchemendy. Tarski on truth and logical consequence. *J. Symb. Log.*, 1988.

Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: a study

through datalog and group theory. *SIAM J. Comput.*, 1998. doi: 10.1137/S0097539794266766.

Cristina Feier, Antti Kuusisto, and Carsten Lutz. Rewritability in monadic disjunctive datalog, MMSNP, and expressive description logics. *Log. Methods Comput. Sci.*, 2019. doi: 10.23638/LMCS-15(2:15)2019.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: an evolving query language for property graphs. In *Proc. SIGMOD*, 2018. doi: 10.1145/3183713.3190657.

Carlo A. Furia and Paola Spoletini. Bounded variability of metric temporal logic. In *Proc. TIME*, 2014. doi: 10.1109/TIME.2014.18.

D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. Many-dimensional modal logics: theory and applications. *Studia Logica*, 2005. URL `philpapers.org/rec/GABMML-2`.

Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal logic (vol. 1): mathematical foundations and computational aspects*. 1994. URL `dl.acm.org/doi/10.5555/186908`.

Anton Gnatenko and Roman Kontchakov. Temporal $\mathcal{EL}$ and equations over sets of integers. In *Proc. DL*, 2026. To appear.

Georg Gottlob and Christos Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 2003. doi: 10.1016/S0890-5401(03)00012-9.

Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Ontological queries: rewriting and optimization. In *Proc. ICDE*, 2011. doi: 10.1109/ICDE.2011.5767965.

Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Roman Kontchakov. Temporalized $\mathcal{EL}$ ontologies for accessing temporal

data: complexity of atomic queries. In *Proc. IJCAI*, 2016a. URL ijcai.org/Abstract/16/160.

Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Roman Kontchakov. Temporalized $\mathcal{EL}$ ontologies for accessing temporal data: complexity of atomic queries. In *Proc. IJCAI*, 2016b. URL ijcai.org/Abstract/16/160.

Steve Harris and Andy Seaborne. Sparql 1.1 query language, 2013. URL w3.org/TR/sparql11-query.

Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. Undecidable boundedness problems for datalog programs. *J. Log. Program.*, 1995. doi: 10.1016/0743-1066(95)00051-K.

Wilfrid Hodges. *Model theory*. 1993. doi: 10.1017/CBO9780511551574.

Ian M. Hodkinson, Frank Wolter, and Michael Zakharyaschev. Monodic fragments of first-order temporal logics: 2000-2001 A.D. In *Proc. LPAR*, 2001. doi: 10.1007/3-540-45653-8_1.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation (3rd edition)*. 2006. URL dl.acm.org/doi/book/10.5555/1177300.

Alfred Horn. On sentences which are true of direct unions of algebras. *J. Symb. Log.*, 1951. URL jstor.org/stable/2268661.

Ismail Husein, Herman Mawengkang, Saib Suwilo, and Mardiningsih. Modeling the transmission of infectious disease in a dynamic network. *J. Phys.: Conf. Ser.*, 2019. doi: 10.1088/1742-6596/1255/1/012052.

Neil Immerman. *Descriptive complexity*. 1999. doi: 10.1007/978-1-4612-0539-5.

ISO/IEC. Information technology - database languages - SQL - part 2: foundation, 1999. URL `iso.org/standard/23532.html`.

Artur Jeż. Conjunctive grammars generate non-regular unary languages. *Int. J. Found. Comput. Sci.*, 2008. doi: 10.1142/S012905410800584X.

Artur Jeż and Alexander Okhotin. Conjunctive grammars over a unary alphabet: undecidability and unbounded growth. *Theory Comput. Syst.*, 2010. doi: 10.1007/s00224-008-9139-5.

Hans Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968. URL `philpapers.org/rec/KAMTLA-4`.

Paris C. Kanellakis. Elements of relational database theory. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. 1990. doi: 10.1016/b978-0-444-88074-1.50022-6.

Evgeny Kharlamov, Nina Solomakhina, Özgür Lütfü Özçep, Dmitriy Zheleznyakov, Thomas Hubauer, Steffen Lamparter, Mikhail Roshchin, Ahmet Soylu, and Stuart Watson. How semantic technologies can enhance data access at siemens energy. In *Proc. ISWC*, 2014. doi: 10.1007/978-3-319-11964-9_38.

Evgeny Kharlamov, Dag Hovland, Martin G. Skjæveland, Dimitris Bilidas, Ernesto Jiménez-Ruiz, Guohui Xiao, Ahmet Soylu, Davide Lanti, Martin Rezk, Dmitriy Zheleznyakov, Martin Giese, Hallstein Lie, Yannis Ioannidis, Yannis Kotidis, Manolis Koubarakis, and Arild Waaler. Ontology based data access in statoil. *J. Web Semant.*, 2017. doi: 10.1016/j.websem.2017.05.005.

Stanislav Kikot, Agi Kurucz, Vladimir V. Podolskii, and Michael Zakharyaschev. Deciding boundedness of monadic sirups. In *Proc. PODS*, 2021. doi: 10.1145/3452021.3458332.

Roman Kontchakov and Michael Zakharyaschev. An introduction to description logics and query rewriting. In *Proc. RW*, 2014. doi: 10.1007/978-3-319-10587-1_5.

Roman Kontchakov, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. Boolean Role Inclusions in DL-Lite With and Without Time. In *Proc. KR*, 2020. doi: 10.24963/kr.2020/58.

Norman Kretzmann, Anthony Kenny, and Jan Pinborg, editors. *The Cambridge history of later Medieval philosophy: from the rediscovery of Aristotle to the disintegration of scholasticism, 1100-1600.* 1982. doi: 10.1017/CHOL9780521226059.

Agi Kurucz, Vladislav Ryzhikov, Yury Savateev, and Michael Zakharyaschev. Deciding fo-rewritability of regular languages and ontology-mediated queries in linear temporal logic. *J. Artif. Intell. Res.*, 2023. doi: 10.1613/jair.1.14061.

Jerzy Łoś. Podstawy analizy metodologicznej kanonów milla. *Ann. Univ. Mariae Curie-Skłodowska, Sect. F.*, 1947. URL `bc.umcs.pl/Content/4085/czas4057_2_1947_5.pdf`.

Malcolm F. Lowe. Aristotle on the sea-battle: a clarification. *Analysis*, 1980. doi: 10.1093/analys/40.1.55.

Jan Łukasiewicz. O logice trójwartościowej. *Stud. Filoz.*, 1988. URL `https://philpapers.org/rec/UKAOLT-2`.

Carsten Lutz and Leif Sabellek. Ontology-mediated querying with the description logic el: trichotomy and linear datalog rewritability. In *Proc. IJCAI*, 2017. doi: 10.24963/ijcai.2017/164.

Carsten Lutz and Leif Sabellek. A complete classification of the complexity and rewritability of ontology-mediated queries based on the description logic $\mathcal{EL}$. *Artif. Intell.*, 2022. doi: 10.1016/j.artint.2022.103709.

Carsten Lutz and Ulrike Sattler. The complexity of reasoning with Boolean modal logics. In *Proc. AML*, 2002. doi: 10.1142/9789812776471_0018.

Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. Temporal description logics: A survey. In *Proc. TIME*, 2008. doi: 10.1109/TIME.2008.14.

Zohar Manna and Amir Pnueli. The temporal logic of reactive and concurrent systems. In *Springer: New York*, 1991. doi: 10.1007/978-1-4612-0931-7.

Marvin L. Minsky. *Computation: finite and infinite machines*. 1967. doi: 10.5555/1095587.

Michael J. Murray. Leibniz on divine foreknowledge of future contingents and human freedom. *Philos. Phenomenol. Res.*, 1995. URL `jstor.org/stable/2108310`.

Jeffrey F. Naughton. Data independent recursion in deductive databases. *J. Comput. Syst. Sci.*, 1989. doi: 10.1016/0022-0000(89)90003-2.

I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An introduction to the theory of numbers*. 1991. URL `books.google.com/books?id=Tt6kEAAAQBAJ&redir_esc=y`.

Alexander Okhotin. Conjunctive grammars. *J. Autom. Lang. Comb.*, 2001. doi: 10.25596/JALC-2001-519.

Alexander Okhotin. Whale Calf, a parser generator for conjunctive grammars. In *Proc. CIAA*, 2002. doi: 10.1007/3-540-44977-9_20.

Alexander Okhotin. A recognition and parsing algorithm for arbitrary conjunctive grammars. *Theor. Comput. Sci.*, 2003. doi: 10.1016/S0304-3975(02)00853-8.

Alexander Okhotin. A simple P-complete problem and its language-theoretic representations. *Theor. Comput. Sci.*, 2011. doi: 10.1016/j.tcs.2010.09.015.

Alexander Okhotin. Conjunctive and Boolean grammars: the true general case of the context-free grammars. *Comput. Sci. Rev.*, 2013. doi: 10.1016/j.cosrev.2013.06.001.

Alexander Okhotin and Christian Reitwießner. Parsing Boolean grammars over a one-letter alphabet using online convolution. *Theor. Comput. Sci.*, 2012. doi: 10.1016/j.tcs.2012.06.032.

Rohit J. Parikh. On context-free languages. *J. ACM*, 1966. doi: 10.1145/321356.321364.

Amir Pnueli. The temporal logic of programs. In *Proc. SFCS*, 1977. doi: 10.1109/SFCS.1977.32.

Arthur Prior. *Past, present and future*. 1967. doi: 10.1093/acprof:oso/9780198243113.001.0001.

M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 1959. doi: 10.1147/rd.32.0114.

Bin Ran and David Boyce. *Modeling dynamic transportation networks*. 1996. doi: 10.1007/978-3-642-80230-0.

Aleksandr Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Math. Notes Acad. Sci. USSR*, 1987. doi: 10.1007/BF01137685.

Juan L. Reutter, Adrián Soto, and Domagoj Vrgoc. Recursion in SPARQL. *Semant. Web*, 2021. doi: 10.3233/SW-200401.

Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 1970. doi: 10.1016/S0022-0000(70)80006-X.

Benjamin Schäfer, Dirk Witthaut, Marc Timme, and Vito Latora. Dynamically induced cascading failures in power grids. *Nat. Commun.*, 2018. doi: 10.1038/s41467-018-04287-5.

John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 1959. doi: 10.1147/rd.32.0198.

A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 1985. doi: 10.1145/3828.3837.

Brian Skyrms and Robin Pemantle. A dynamic model of social network formation. *Proc. Natl. Acad. Sci. U.S.A.*, 2000. doi: 10.1073/pnas.97.16.9340.

Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. STOC*, 1987. doi: 10.1145/28395.28404.

Richard Snodgrass. The temporal query language TQUEL. *ACM Trans. Database Syst.*, 1987. doi: 10.1145/22952.22956.

Richard Thomas Snodgrass. *Developing time-oriented database applications in SQL*. 1999. URL `dl.acm.org/doi/10.5555/320037`.

Howard Straubing. *Finite automata, formal logic, and circuit complexity*. 1994. doi: 10.1007/978-1-4612-0289-9.

Shawn Zheng Kai Tan, Shounak Baksi, Thomas Gade Bjerregaard, Preethi Elangovan, Thrishna Kuttikattu Gopalakrishnan, Darko Hric, Joffrey Joumaa, Beidi Li, Kashif Rabbani, Santhosh Kannan Venkatesan, Joshua Daniel Valdez, and Saritha Vettikunnel Kuriakose. Digital evolution: Novo Nordisk's shift to ontology-based data management. *J. Biomed. Semantics*, 2025. doi: 10.1186/s13326-025-00327-4.

Abdullah Uz Tansel, James Clifford, Shashi Gadia, Sushil Jajodia, Arie Segev, and Richard Snodgrass. *Temporal databases: theory, design, and implementation*. 1993. URL `dl.acm.org/doi/10.5555/151098`.

Alfred Tarski. The semantic conception of truth and the foundations of semantics. *Philos. Phenomenol. Res.*, 1943. doi: 10.7551/mitpress/4884.003.0028.

Alfred Tarski. On the concept of logical consequence. In *Logic, Semantics, Metamathematics*. 1983. URL `homepages.uc.edu/~martinj/History_of_Logic/Definition_of_Truth_and_Logical_Consequence`.

Marcin Tkaczyk and Tomasz Jarmużek. Jerzy łoś positional calculus and the origin of temporal logic. *Logic and Logical Philosophy*, 2019. doi: 10.12775/llp.2018.013.

A. W. To. *Model-checking infinite-state systems: generic and specific approaches*. PhD thesis, 2010. URL `era.ed.ac.uk/items/31c93b69-8d34-4215-ba04-039a5172b961`.

Valentina Urzua and Claudio Gutierrez. Linear recursion in G-CORE. In *Proc. FDM*, 2019. URL `ceur-ws.org/Vol-2369/short07.pdf`.

Ron van der Meyden. Predicate boundedness of linear monadic datalog is in PSpace. *Int. J. Found. Comput. Sci.*, 2000. doi: 10.1142/S0129054100000351.

Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. STOC*, 1982. doi: 10.1145/800070.802186.

Moshe Y. Vardi. Decidability and undecidability results for boundedness of linear recursive queries. In *Proc. PODS*, 1988. doi: 10.1145/308386.308470.

Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*. 1996. doi: 10.1007/3-540-60915-6_6.

Przemysław A. Wałęga, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. DatalogMTL: computational complexity and expressive power. In *Proc. IJCAI*, 2019. doi: 10.24963/ijcai.2019/261.

Przemysław A. Wałęga, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. DatalogMTL over the integer timeline. In *Proc. KR*, 2020. doi: 10.24963/kr.2020/79.

Jan Woleński. Logical ideas of jan łukasiewicz. *Studia Humana*, 2019. doi: 10.2478/sh-2019-0011.

Mincheng Wu, Chao Li, Zhangchong Shen, Shibo He, Lingling Tang, Jie Zheng, Yi Fang, Kehan Li, Yanggang Cheng, Zhiguo Shi, Guoping Sheng, Yu Liu, Jinxing Zhu, Xinjiang Ye, Jinlai Chen, Wenrong Chen, Lanjuan Li, Youxian Sun, and Jiming Chen. Use of temporal contact graphs to understand the evolution of COVID-19 through contact tracing data. *Commun. Phys.*, 2022. doi: 10.1038/s42005-022-01045-4.

Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: a survey. In *Proc. IJCAI*, 2018.

Mengkai Xu, Srinivasan Radhakrishnan, Sagar Kamarthi, and Xiaoning Jin. Resiliency of mutualistic supplier-manufacturer networks. *Sci. Rep.*, 2019. doi: 10.1038/s41598-019-49932-1.

Dmitriy Zhuk. Π_2^P vs PSpace dichotomy for the quantified constraint satisfaction problem. In *Proc. FOCS*, 2024. doi: 10.1109/FOCS61266.2024.00043.