

Foundations of Advanced Reasoning in Recursive SHACL

DISSERTATION

zur Erlangung des akademischen Grades

Doktorin der Technischen Wissenschaften

eingereicht von

Anouk Michelle Oudshoorn, MSc

Matrikelnummer 12229977

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof.in Dr.in techn. Magdalena Ortiz, MSc

Zweitbetreuung: Assistant Prof. Dr.techn. Mantas Šimkus, MSc

Diese Dissertation haben begutachtet:

Jan van den Bussche

Jean Christoph Jung

Wien, 4. November 2026

Anouk Michelle Oudshoorn

Foundations of Advanced Reasoning in Recursive SHACL

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der Technischen Wissenschaften

by

Anouk Michelle Oudshoorn, MSc

Registration Number 12229977

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof.in Dr.in techn. Magdalena Ortiz, MSc

Second advisor: Assistant Prof. Dr.techn. Mantas Šimkus, MSc

The dissertation has been reviewed by:

Jan van den Bussche

Jean Christoph Jung

Vienna, November 4, 2026

Anouk Michelle Oudshoorn

Declaration of Authorship

Anouk Michelle Oudshoorn, MSc

I hereby declare that I have written this thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work – including tables, maps and figures – which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix “Overview of Generative AI Tools Used” I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, November 4, 2026

Anouk Michelle Oudshoorn

Acknowledgements

I would like to thank Magdalena Ortiz and Mantas Šimkus: I could not have wished for better or more suitable supervisors. You are beautiful people who first of all guided me through my PhD in such a smooth and pleasant way. Every time I believe I understand how valuable you are, you are capable of positively surprising me again. You enthused me over and over and showed me mathematical problems I did not even know I could find so intriguing. I am proud I can call you my supervisors, and cannot capture in words how important it is to me you keep standing up and speaking out for your values. You show everyone how academia can be a better and more welcoming place.

I would also like to thank you for inviting me to visit you in Umeå on three different occasions, and your support for my preference to travel over land whenever possible, even when it takes much longer. In this light, I am grateful to Shqiponja Ahmetaj for her guidance and supervision in my first year in Vienna: with Mantas and Magdalena living in Sweden, it was invaluable to be able to rely on you.

I would like to thank my reviewers, Jan van den Bussche and Jean Christoph Jung for their willingness to review my thesis and their sharp observations, and that on such a tight schedule.

This PhD project would not have been the same without my research visits: first of all, I am grateful to Sebastian Rudolph for his invitation to visit his group in Dresden. That was a beautiful experience that initiated multiple research projects that continue up till this day. I would also like to thank Juan Reutter for welcoming me in Santiago and for how pleasant I experienced our collaboration. Moreover, your warm invitation that I am welcome to return whenever I want to helped me realise that moving to another continent is an actual possibility. In this regard, I would also particularly like to thank Camila and Paula, who made me feel at home in Santiago in such an effective way that I am returning soon, and Magdalena and Ino, for making me consider a research visit in Chile in the first place; I would not have wanted to miss out on this.

Furthermore, I am grateful to everyone who made it possible for me to even start this dissertation. Moreover, I would like to thank all my collaborators, the knowledge based systems group and the faculty of informatics for supporting me and making it possible to do my work and write this thesis. In particular, I would like to thank everyone in the office, and specifically my office mate Sanja, who made me happy to come there everyday: to simply work and have lunch together, to sing together, to play board games, or to plan international train journeys together.

My time in Vienna would not have been the same, and much more boring, without everyone around me. In particular, Aida and Sofia: I love dancing with you too! Mark, it is simply incredible to have you as a friend. Luc, thank you for being so supportive and for providing me the space to do what I needed to do, over and over. And Nelson, you made the last half year of my thesis so much calmer and happier than it would have been without you, simply by making music together and drinking tea, thank you for all your support. Finally, I would like to thank my parents, brothers and family for being just that: family, and all the support you have given to me. Lastly, I would like to thank Netidee for supporting my thesis project with a stipend, nr. 6794.

Kurzfassung

Die Shape Constraint Language (SHACL) ist einer der bedeutenden W3C-Standards zur Verwaltung von Resource Description Framework (RDF)-Daten, einem graphbasierten Datenmodell des Internets. Ihr Zweck besteht darin, Constraints für Datensätze zu beschreiben. Die zentrale Frage ist die Validierung gegebener Constraints auf einem Graphen zur Sicherstellung der Datenqualität. Der SHACL-Standard zielt auf eine Kombination mit der Ontology Web Language (OWL), ebenfalls ein W3C-Standard, ab. Es bleibt aber offen, wie die unterschiedlichen Sichtweisen der beiden Ansätze kombiniert werden können. Darüber hinaus beschreibt der SHACL Standard rekursive Constraints, ohne explizit festzulegen, welche Semantik ihnen zugrunde liegen. Dies ist insbesondere relevant wenn negierte Constraints sich gegenseitig referenzieren. Frühere Arbeiten bedienten sich dazu bekannten Ansätzen aus der logischen Programmierung wobei aber viele interessante Fragen offen blieben.

Diese Dissertation beschäftigt sich mit rekursiven SHACL und beleuchtet deren Kombination mit Ontologien und sowie mit logische Schlussfolgerungen jenseits von Validierung. Was ist eine natürliche Semantik für Validierung von SHACL mit OWL Ontologien? Für welche Fragmente sind konkrete Ansätze bekannt? Welche Komplexität hat die statische Analyse von Problemen wie Implikation und Erfüllbarkeit von SHACL? Die erste Hälfte dieser Arbeit behandelt die Kombination mit Ontologien und gibt einen Überblick verschiedener Semantiken von rekursiven SHACL Constraints. Die zweite Hälfte präsentiert neue Komplexitätsresultate für die statische Analyse der beschriebenen Semantiken.

Durch die Beantwortung dieser Fragen schaffen wir die Grundlage für eine weitergehende Anpassung von SHACL zur Erkennung von Dateninkonsistenzen, während gleichzeitig die naheliegenden Lücken herausgefiltert werden, die durch das Domänenwissen abgedeckt werden. Dabei stellen wir interessante Techniken vor. Dazu gehören eine einfachere Methode zur Konstruktion von universellen Kernmodellen als durch die Verwendung des Core Chase für bestimmte Horn-Logiken; eine neue Art von Ersetzungssystemen, die SHACL-Dokumente verarbeiten kann und dadurch neue Erkenntnisse darüber liefert wie Rekursion und stratifizierte Negation in Ersetzungssystemen behandelt werden können; sowie eine neue Technik, die Mengen geeigneter Modelle konstruiert, um minimale Antworten über alle Modelle hinweg zu zählen. In der zweiten Hälfte festigen wir zunächst die enge Verbindung zwischen SHACL und Description Logics (DLs) im Hinblick auf die Erfüllbarkeit. Darauf folgen Unentscheidbarkeitsergebnisse für die Implikation von SHACL-Dokumenten unter zwei der wichtigsten Semantiken. Abschließend präsentieren wir ein Entscheidbarkeitsresultat für die Implikation von SHACL-Dokumenten unter

der wohlfundierten Semantik, das durch eine direkte Übersetzung der wohlfundierten Semantik in den vollständigen hybriden μ -Kalkül erzielt wird.

Abstract

The Shape Constraint Language (SHACL) is one of the prominent W3C standards for managing Resource Description Framework (RDF) data, a graph-based data model of the web. Its purpose: to describe constraints on datasets. The main task of interest is validation of a given a set of constraints on a given graph, which ensures data quality. The SHACL standard notes that its combination with the Ontology Web Language (OWL), also a W3C standard, is desirable, omits however how to combine the different world views of both. Moreover, this document describes recursive constraints in SHACL, without making explicit what the supposed semantics should be. This is even more interesting in light of constraints possibly referring to each other in a negated manner. Existing literature has focussed on filling the latter gap by considering different semantics based on well-known Logic Programming techniques. However, many interesting questions remained unanswered.

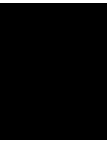
This thesis focuses on recursive SHACL; partly about its combination with ontologies, and partly about reasoning services beyond validation: what can be considered a natural semantics for validation of SHACL in presence of OWL ontologies? For which fragments are concrete techniques known? And can we determine the complexity of static analysis problems like implication and satisfiability for SHACL? The first two questions are answered in the first half of this thesis, including an overview of the different semantics for recursive SHACL seen as a logic. The second half focusses on static analysis, and presents a list of new complexity results, considering the different semantics for recursive SHACL.

By answering these questions, we set the stage for a broader adaptation of SHACL to detect data inconsistencies, while at the same time filtering out the straightforward gaps, taken care of by the domain knowledge. While doing so, we present exciting techniques. Those contain a simpler way to build core universal models than using the core chase for certain Horn logics; a new type of rewriting technique that can handle SHACL documents, which sheds light on ways to handle recursion and stratified negation in rewriting approaches; and a new technique which builds sets of good models for counting minimal answers over all models. In the second half, we first solidify the tight connection between SHACL and DLs in regard of satisfiability. This is followed by undecidability results for SHACL document implication under two of the main semantics. Finally, we present a decidability result for SHACL document implication under the well-founded semantics, obtained via a direct translation of the well-founded semantics into the full hybrid μ -calculus.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 SHACL and Reasoning	2
1.2 Static Analysis	4
1.3 Main Results	5
2 Preliminaries	11
2.1 Description Logics	12
2.2 Universal Models	18
2.3 Some Complexity Theory	21
3 SHACL as a Logic	25
3.1 Non-Recursive SHACL	25
3.2 Recursive SHACL	27
3.3 Conclusion	31
4 Validation in the Presence of Ontologies	33
4.1 Validation with Ontologies	34
4.2 Finite and Infinite Cores	41
4.3 Stratified SHACL	46
4.4 Rewriting for Positive SHACL	52
4.5 Rewriting for Stratified SHACL	60
4.6 Pure Rewriting	65
4.7 Complexity Results	69
4.8 Discussion and Conclusion	72
4.9 Related Work	75
5 Towards Practical Validation	77
5.1 $DL\text{-}Lite_{\mathcal{R}}$	77
	xiii

5.2	Data-Aware Good Successor Configuration	79
5.3	Core Universal Model for $\mathcal{ELHI}$	82
5.4	Semi-Positive SHACL Validation	83
5.5	Conclusion	85
5.6	Related Work	86
6	Counting Over Regular Expressions	87
6.1	Problem Statement	89
6.2	MINCOUNT is Hard	89
6.3	Fragment of DL-Lite $\mathcal{R}$	95
6.4	Polynomial-time Algorithm	96
6.5	Decision Algorithm	102
6.6	Conclusion	105
6.7	Related Work	106
7	SHACL satisfiability via Description Logics	107
7.1	SHACL Satisfiability	107
7.2	SHACL to OWL and back again	111
7.3	Inverses, Nominals and Counting	113
7.4	Equality and Disjointness	117
7.5	Conclusion	118
7.6	Related Work	119
8	Static Analysis: Well-Founded Validation	121
8.1	Fragment and Normal Form	122
8.2	Static Analysis of SHACL Specifications	122
8.3	Full Hybrid μ -calculus	125
8.4	Translating SHACL into μ -calculus	127
8.5	Complexity Results	136
8.6	Conclusion	144
8.7	Related Work	145
9	Conclusion	147
	Overview of Generative AI Tools Used	151
	List of Figures	153
	List of Tables	155
	Bibliography	157



Introduction

As our current society relies more and more on collecting and analysing huge amounts of data in many different formats, an increased need for efficient and flexible tools and standards to handle this data is observed. The *Resource Description Framework* (RDF) is a graph-based model for managing data on the Web, introduced by the World Wide Web Consortium (W3C). At the moment, it is the standard way how web data is published and shared. It is for instance used in sharing health care information: Fast Healthcare Interoperability Resources (FHIR), [HL711], helps in deciding the best possible treatment for patients. Other use cases contain for instance assistance in failure analysis for semi-conductors.

As the amount of data published on the web is growing constantly, a demand surfaced for a language to describe and validate the correctness or quality of RDF graphs. Specifically in the current moment, as LLMs are used to generate vast amounts of data that are uploaded on the web, having a logic based method that guarantees certain quality constraints on your data has become essential. In 2017, the W3C published their solution: the Shapes Constraint Language, better known as SHACL [W3C17].

SHACL primarily consists of *constraints*, sometimes also called *shapes*. These are descriptions of how parts of the data are supposed to look like, like “this node needs to have an outgoing edge labelled with *hasChild* to a node that also has an outgoing *hasChild*-edge.” Constraints also come with *targets*: a set of nodes that needs to validate (or conform, or satisfy) the constraint. For instance, we may want all nodes labelled with *Grandparent*, *Grandmother* or *Grandfather* to validate the example constraint.

Next to mentioning the type of in- or outgoing edges, SHACL constraints may also talk about more local properties like the label of nodes, or more global properties like equality or disjointness of sets of paths. In this way, we can check whether all nodes labelled with *Grandmother* are also labelled with *Grandparent*, or check whether *hasChild* · *hasChild* indeed coincides with *hasGrandchild*. Furthermore, one may count over regular expressions. Or, by using shape names as place holders for constraints, we can refer to other constraints inside others, both in a positive

as in a negated way. The SHACL standard [W3C17] describes the syntax and semantics of constraints in a somewhat convoluted way: it is not straightforward to parse. The first logical formalisation was performed by Corman et al. [CRS18]. Later followed more works highlighting the proximity to Description Logics (DLs) [BJVdB22b, Ort23] and a thorough analysis of the expressivity of the different features used to build constraints [BJVdB22a].

In recursive SHACL there is no limit on the creation of cyclic definitions of constraints, as opposed to non-recursive SHACL where cycles are forbidden. As is well-known from the Logic Programming literature, the combination of recursion and negation can be tricky. The SHACL standard is not of much help here: it leaves the choice of semantics for recursive constraints to the implementers, which leads to ad hoc implementations of which it is unclear what is tested exactly. However, in the Logic Programming literature many solutions have already been presented, which can be extended to SHACL: the main different semantics for recursive SHACL considered in the literature are based on the supported model [CRS18], stable-model [GL88, ACO⁺20] and well-founded semantics [VGRS91, OŠ24].

Thus, a big first step has been made by providing a range of choice of validation semantics, based on the different ways of handling the interaction of negation and recursion. However, many more questions remain. With this thesis, we aim to make further steps; first, we will look at SHACL validation in presence of ontologies, whereas in the second half of this thesis we will consider static analysis problems of recursive SHACL under all of the main semantics.

1.1 SHACL and Reasoning

Like SHACL, the Web Ontology Language (OWL) [W3C12] is another prominent W3C standards for managing RDF data. Both standards are based on fundamentally different assumptions and designed to be complementary. OWL was standardised shortly after RDF, with the key aim of enhancing RDF datasets with domain knowledge that enables the inference of missing facts from potentially incomplete data graphs: OWL enables to capture repeating or generalisable knowledge in a precise manner. This is because OWL and its profiles are based on *Description Logics (DLs)* [BHLS17].

Just as other classical logics, DLs or OWL make the *open-world assumption (OWA)*. Intuitively, this means that the data only presents an incomplete description of the domain of interest: it asserts facts that are known to be true, but does not rule out that additional facts may also be true, as long as they are consistent with the current world description. OWL has been adopted in a wide range of applications over the years, and thousands of OWL ontologies have been developed (see e.g., [HL711]). SHACL, in contrast, was created for a different purpose: to describe and validate constraints on datasets. Recall that the main task of interest is *validation* of a given a set of constraints paired with a selection of *target* nodes or concepts from a given graph. Unlike OWL, SHACL operates under the *closed-world assumption (CWA)*: it assumes that the given data graph is complete, and validators evaluate the constraints over the input graph as is.

The first natural question we address is how to do validation in the presence of both OWL ontologies and SHACL constraints. That is, if we have a possibly incomplete graph and ontological

knowledge that implies additional facts, can we validate given SHACL constraints over graphs containing the implied facts?

The first major challenge we must face is that the semantics of SHACL constraints in the presence of ontologies is not obvious, as we must simultaneously account for the open-world semantics of description logics and for the closed-world view of SHACL. The knowledge in the ontology implies additional facts, which can be added to the data graph so that it satisfies all the ontological axioms. This is not a new problem: several works addressing this exist [KAH11, MHS07b, MHS07a, FG10].

A related problem is how to evaluate queries over description logic knowledge bases. This has been the subject of huge research efforts over almost two decades. Here, the *certain answer semantics* has played a prominent role in elegantly handling information incompleteness for large classes of queries: answering a query consists of computing the intersection of query answers over all models of the input knowledge base (KB). This problem can easily become computationally very costly for expressive DLs, and it is provably more expensive than standard reasoning in most of the popular extensions of $\mathcal{ALC}$ [OŠ12]. But fortunately, this computational hurdle can usually be avoided in the so-called *Horn* DLs, which disallow ontological statements that involve disjunction, leading in turn to the *universal model property* [BO15, ORŠ11]. This property guarantees that a consistent KB always has a model that represents all models (technically, that can be homomorphically mapped into any model) and is thus sufficient to compute the certain answers to all conjunctive queries.

Although related, SHACL validation differs from query evaluation in the sense that SHACL constraints may contain recursion, and has a much stronger bigger role for negation, which is capable of detecting the difference between various universal models. In other words, SHACL constraints are not invariant under homomorphisms, and just taking any universal model is no longer an option.

Research Questions. The main topic of the first half of this dissertation can be summarised in one main question.

How can we combine SHACL and ontologies?

We break this down into three concrete subquestions.

Q1. What are adequate semantics for validating SHACL in presence of ontologies?

This means finding a meaningful combination of SHACL and different description logics, and aligning the technical details of both. For the smaller fragments, the main difficulty lies in finding an intuitive semantics. For more expressive fragments of SHACL and description logics the main challenge will be a computational one: whether it is possible and how to include the whole set of features of SHACL in a meaningful way, while preserving decidability and possibly tractability.

Q2. What is the complexity of these semantics?

The next step is to develop optimal algorithms for determining validation of SHACL combined with ontologies, to determine upper bounds of the complexity for the different fragments we are considering. For determining the lower bounds we can rely on straightforward methods: finding the right problem in a certain complexity class and reformulating this problem in our setting.

Q3. How can we validate these semantics?

Can we somewhat push the practicability of our techniques? It seems that the most promising path is reducing validation of SHACL with ontologies to plain validation of SHACL, by rewriting the constraints based on ontological reasoning. However, it seems that these algorithms will need quite some computational complexity power to take care of all the technical corner cases. Thus, we most likely will have to put some restrictions on the SHACL constraints, and possibly also on the ontology language, to find algorithms for answering whether a certain given SHACL document is validated by a given ontology on some RDF graph, and such that these algorithms are tractable in size of the combined complexity of the rules in the ontology and the RDF graph. Deciding which restrictions are the most meaningful, i.e., are mostly excluding some theoretical, complicated, constructions and not the main features used, is also part of solving this subquestion.

Chapters 4, 5 and 6 are dedicated to these questions. We start with answering Q1 in Chapter 4, whereas the second half of Chapter 4 already touches on Q2 and Q3. In Chapter 5, we take a closer look at Q3 specifically. Finally, in Chapter 6 we reconsider all questions for a specific feature of SHACL: counting over regular expressions.

1.2 Static Analysis

Static analysis problems are of major importance in the design and validation of SHACL-based solutions: as SHACL becomes more popular, substantive efforts are put into the adoption of SHACL. As part of this, we witness mining SHACL specifications from data [OTMH23, FGG22, RLH23], but how to assess the quality of these machine-generated constraints? And how to combine or compare multiple, possibly generated, specifications? We note that the basic necessary condition here is compatibility, which boils down to the satisfiability problem, and containment or implication of documents. Finally, both satisfiability and containment or implication, as static analysis tools, are prerequisites for more advanced services like optimisation, incremental validation and modularity.

Given the importance of these problems, there are remarkably few results concerning its decidability and complexity. Indeed, the most notable work in this direction is very coarse [PKM22]. It builds on a tailored fragment of predicate logic to identify decidability and complexity bounds, but the basic logic it considers is already close to the boundary of what could potentially be decidable in the presence of cardinality constraints. Put differently, the base logic is already rather expressive (for instance already containing nominals and inverses), which means that barely any features can be added for the logic to keep decidability of satisfiability (which is weaker than implication). That is, the positive results are mostly limited to formalisms that do not support

counting, and more often than not consider unrestricted (that is, potentially infinite) graphs, even though finite graphs are a more relevant setting here.

There are multiple versions of either problem that one can consider, ranging from considering just single constraints to whole SHACL documents, containing recursion and sets of targets. Moreover, recall the multiple semantics introduced in the SHACL literature. Given that not many works have appeared in this direction, a lot of problems remain open to be addressed.

In quite some cases, the tight connection between SHACL and DLs appears promising to derive new results, specifically for the supported model semantics. However, there are only limited works following this approach. Leinberger et al. [LSR⁺20] study constraint containment, which addresses whether one constraint subsumes another without considering any other constraints. However, as pointed out by Bogaerts et al. [BJVdB22b], there are some issues with their translation into DLs. Furthermore, Di Stefano and Šimkus [DSŠ24] study description logic terminologies under the stable model semantics, and conclude that complexity results for satisfiability of some fragments of SHACL under the stable model semantics transfer. For the well-founded semantics, no results have been published so far.

Research Question. The second half of this thesis, consisting of Chapters 7 and 8, is focussed on addressing the following question.

- Q4. Can we find decidable instances of (recursive) SHACL satisfiability and containment or implication?

SHACL satisfiability asks the question whether for a given set of constraints (possibly with targets) there exists a data graph that validates the targets. The containment problem requires two sets of constraints (again possibly with targets) and asks whether all graphs that satisfy the targets of the first set, also satisfy the targets of the second set. The main idea is a mere static analysis of the combination, for all of the main semantics: supported, stable model and well-founded. In practice, answering this question helps to find more efficient SHACL constraints and helps us understand what we are asking exactly with our constraints.

In Chapter 7, we will continue the work on the connection to DLs. This opens up decades of research in the fine-grained study of logical fragments and the effect that the interaction between different shapes of subformulas has on the complexity of reasoning. This is an especially promising approach for the supported model semantics. However, for the well-founded semantics, this is much less probable to work, and for the stable model semantics, similar results regarding description logic terminologies are much thinner spread.

1.3 Main Results

First, in Chapter 2, we provide some preliminary definitions to make this thesis self-contained. Next, in Chapter 3, we provide a summary of for this thesis relevant parts of existing literature on SHACL as a logic. The remaining chapters discuss the following results.

Chapter 4. We start with providing a semantics for validating SHACL in presence of ontologies, and argue why it is intuitive. For this, we introduce the notion of the austere universal model, a universal model which locally does not contain redundant structures, and advocate checking for validation over this specific model. As this austere universal model may be infinite, we show we can also provide a finite representation in the form of the *good successor configuration*. It turns out that the result of our construction coincides with the model construction proposed in [EOŠ⁺12] and is indeed universal.

Moreover, we show that the local minimality of the austere universal model suffices for global minimality: the austere universal model is a core, and it is the unique universal core model of a consistent Horn- $\mathcal{ALCHIQ}$ TBox. This provides strong evidence in favour of our chosen semantics, since cores are often advocated as the adequate choice for languages that are not closed under homomorphisms, but satisfy the weaker property of being closed under isomorphisms [AGP18, BF19, EKM22].

We define a fragment of recursive SHACL named *stratified SHACL*. Our notion of stratification is based on the well-known class of stratified logic programs [ABW88], and provide a *least fixed point semantics* that is admitted by stratified SHACL documents. This semantics coincides with both the stable [ACO⁺20] and the well-founded [OŠ24] semantics, but is simpler to compute. We also show that our fragment has a rather simple normal form with the same expressivity.

Furthermore, we are able to bring validation over the possibly infinite austere universal model back to validation of a rewritten set of constraints over an enriched ABox. We do this by combining the normal form of stratified SHACL with the information captured in the good successor configuration. We continue with exploring how to construct a rewriting of SHACL with ontologies into SHACL which can be evaluated over the original (not enriched) data graph. One of these techniques proposes an extension of SHACL, SHACL^b, which also allows to define labels for roles.

Finally, we determine the complexity of validating SHACL with ontologies. We find that in presence of Horn- $\mathcal{ALCHIQ}$ TBoxes, SHACL validation is EXPTIME complete in combined complexity, and PTIME complete in data complexity. Moreover, we show that validating a very simple fragment of SHACL over a rather light (less expressive than $DL-Lite_{\mathcal{R}}$) description logic ontology already suffices to find EXPTIME hardness in combined complexity.

This chapter is based on the following publication.

[OOŠ26a] Anouk M. Oudshoorn, Magdalena Ortiz, and Mantas Šimkus. SHACL validation in the presence of ontologies: Semantics and rewriting techniques. *Artif. Intell.*, 352:104483, 2026

Chapter 5. In this chapter, we explain how the presented techniques of Chapter 4 can be simplified for less expressive DLs: $DL-Lite_{\mathcal{R}}$ and $\mathcal{ELHI}$. In particular, this includes a *data-aware* version of a consequence-based calculus for $\mathcal{ELHI}$ KBs. Unlike previous inference procedures for $\mathcal{ELHI}$, the rules in our calculus are aware of input data, which assists in avoiding

the main drawback of the previous approaches: data-independent computations easily lead to best-case exponential behaviour.

After presenting our calculus, we use the data aware version to address validation of SHACL in the presence of ontologies. In this part, we concentrate on *semi-positive* SHACL shapes, which allow for negation of ontology predicates, but not of *shape names*. This fragment can be seen as semi-positive monadic Datalog queries with acyclic rule bodies and regular path expressions. This also shows the influence of referencing shape names negatively in the SHACL rewriting procedure, as the rewriting procedure can be drastically simplified compared to the procedure presented in Chapter 4. Moreover, this rewriting is no longer best-case exponential, as is the version of the previous chapter.

This chapter is based on the following publications.

- [AOŠ23] Shqiponja Ahmetaj, Magdalena Ortiz, Anouk M. Oudshoorn, and Mantas Šimkus. Reconciling SHACL and ontologies: Semantics and validation via rewriting. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Radulescu, editors, *Proc. of ECAI 2023*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 27–35. IOS Press, 2023
- [OŠ24] Anouk M. Oudshoorn, Magdalena Ortiz, and Mantas Šimkus. Reasoning with the core chase: the case of SHACL validation over $\mathcal{ELHI}$ knowledge bases. In Laura Giordano, Jean Christoph Jung, and Ana Ozaki, editors, *Proc. of DL 2024*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024

Chapter 6. In this chapter, we dive deeper into counting over regular expressions, one of the SHACL features. It asks for the amount of nodes reachable by a regular expression, independent of how many different paths lead there. In particular, we are interested in its behaviour over all the models of a given knowledge bases. As we are not counting homomorphisms, this is a problem for which existing related work does not provide any solutions. Instead, we consider nodes reachable, which means a more global view in the construction of models is required.

First, we extend the framework of *cardinality certain answers* [KR15, BMT20], originally designed for conjunctive queries over ontologies, to counting the number of nodes reachable via a regular path expression (over ontologies). We show that deciding whether this number is larger or smaller than a given bound is coNP-hard in data complexity under very mild assumptions. In particular, for $DL-Lite_{\mathcal{R}}$ ontologies, coNP-hardness holds even when regular expressions are restricted to concatenation, making it a rooted CCQ, as long as either queries or ontologies feature inverse roles. We show that for almost all more expressive ontology languages the problem is coNP-hard even without using inverses.

Nevertheless, there is also positive news; for the restricted case when neither queries nor ontologies can use inverse roles, we show how to obtain tractability. Our algorithm is not based on computing a particular model over which queries are evaluated. Instead, we prove the model with the minimal number of answers can be obtained by inspecting a small (polynomially sized, in data complexity) set of models, which we call a counting-sufficient set of models.

Finally, we show that counting-sufficient sets of models can be enumerated by focussing on a family of mappings that encode how blank nodes in a specific universal model are merged to obtain such a set of models. We leverage this family of mappings to show how to produce a logical formula that effectively allows us to reason about the number of reachable nodes without materialising any of the models in the counting-sufficient set.

This chapter is based on unpublished work in collaboration with Juan Reutter (under submission at ICDT 2027).

Chapter 7. From this chapter onwards, we focus on static analysis problems, starting with satisfiability for the supported model semantics. In this chapter, we build on the DL literature to pinpoint much tighter complexity bounds than previously known for SHACL, based on the close connection between satisfiability in DLs and SHACL satisfiability; we revisit this connection and explain how to translate complexity results in both ways. To emphasise this tight bond, we provide a DL inspired naming convention: we write $\mathcal{L}_S$ to denote the SHACL fragment similar to the DL $\mathcal{L}$.

Secondly, we add some lack of finite model property results to the landscape: we show this for $\mathcal{ALC}_S$ plus counting over regular path expressions, which also provides an alternative undecidability proof; and, we show that adding either $\text{eq}(E, r)$ or $\text{disj}(E, r)$ to $\mathcal{ALC}_S$ also breaks the finite model property of $\mathcal{ALC}_S$.

This chapter is based on the following publication.

[Oud25] Anouk M. Oudshoorn. SHACL satisfiability: What can we learn from DLs? In *Proc. of DL 2025*, volume 4091 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2025

Chapter 8. Finally, the last chapter provides the first dedicated study of the computational complexity of implication of SHACL documents in the presence of recursive constraints. That is, we study the problem of deciding, given two SHACL documents $\mathcal{S}_1$ and $\mathcal{S}_2$ containing shape definitions and targets for validation, whether every graph that validates $\mathcal{S}_1$ also validates $\mathcal{S}_2$. Our first major result is that, if one considers the supported or the stable model semantics, this problem is undecidable even when the language for shape expressions is restricted to $\mathcal{ALCIO}_S$. This is somewhat surprising, but becomes even more intriguing in the light of our second main result: under the well-founded semantics, this problem is not only decidable, but it is feasible in single exponential time and thus not harder than deciding satisfiability of a single shape expression.

The main stepping stone to obtaining our positive results is a translation of SHACL documents under the well-founded semantics into a formula of the full hybrid μ -calculus. This translation establishes a novel connection between fixed point logic on the one hand, and the well-founded semantics on the other. Interestingly, capturing the well-founded semantics requires limited alternation of fixed point operators. This connection is, to our knowledge, novel and of independent interest. We note that some early works on the well-founded semantics for logic programs have already established a connection for logics with fixed points [VGRS91, VG93]. Our setting here is different: their monotonic operator is composed of two anti-monotonic fixed point operators.

That is, the first operator produces an overestimate of the set of negative conclusions, whereas the second is an underestimate. This alternating, estimating behaviour is absent in our work, in the sense that we translate (directly) into (fragments of) the full hybrid μ -calculus.

We rely on the insights of our more constructive translation to derive tight complexity bounds. That is, our translation of a SHACL document into a μ -calculus formula unfolds the rules defining shapes into a single formula, potentially leading to an exponential blow-up. We therefore also provide an automata-based algorithm for evaluating (the μ -calculus translation of) a SHACL document that runs in time bounded by a single exponential function of the size of the input document. This allows us to establish optimal complexity bounds for SHACL static validation services in a range of fragments.

This chapter is based on the following publication.

[OOS26b] Anouk M. Oudshoorn, Magdalena Ortiz, and Mantas Simkus. Static analysis of recursive SHACL. In Renata Wassermann, Marie-Laure Mugnier, and Franz Baader, editors, *Proc. of KR 2026*, 2026

Preliminaries

In this chapter, we will lay out the necessary background for this thesis. After introducing some basic syntax, we will introduce the syntax and semantics of Description Logics. Furthermore, for completeness of this thesis, we will also address some automata and complexity theory.

Throughout this work, we will work with the following, countably infinite, mutually disjoint, alphabets. Note that these alphabets are specifically chosen to fit both DLs and the logic formalisations of SHACL. Moreover, it needs to be compatible with RDF graphs. To align this needs, we present a syntax that is closely related to syntax known for DLs.

First, we introduce two sets of names for nodes: the named versions, and the anonymous versions. The named versions, called *individuals* or *constants*, are used, among others, to represent known data, whereas the blank versions are mainly used in universal model constructions.

N_I indicates the set of *individuals* or *constants*: names for specific nodes in data graphs or ontologies, usually denoted by small letters $a, a', a_1, a_2, \dots, b, c, \dots$

N_B indicates the set of *blank nodes* (sometimes simply called *blanks*): anonymous nodes in data graphs, usually denoted by small letters $d, e, e', e_1, e_2, x, y, \dots$. Elements of the union of N_B and N_I are usually denoted like elements of N_B by small letters $d, e, e', e_1, e_2, \dots$

Furthermore, we introduce alphabets to indicate unary and binary predicates, as is standard in DL literature.

N_C indicates the set of *concept names*: names for unary predicates in data graphs or ontologies, usually denoted by capitalised letters $A, A', A_1, A_2, \dots, B, \dots$ or words like *Worm*, *Human* or *Carrot* in concrete examples. In RDF and SHACL, these elements are also referred to as *classes* or *class names*.

N_R indicates the set of *role names* or *roles*: names for binary predicates in data graphs or ontologies, usually denoted by small letters $p, p', \dots$. In RDF and SHACL, these elements are also referred to as *properties* or *property names*.

In the construction of SHACL constraints, we make use of both the individuals, the concept names as the property names. To give names to constraints, and indicate which nodes conform to which shapes, we introduce another set of unary predicates: the shape names.

N_S indicates the set of *shape names*: names for unary predicates in shape assignments, usually denoted by small letters $s, s', t, t', \dots$ or words like *petShape* in concrete examples.

Finally, in Chapter 8, we will make use of μ -calculus formulas. For building these, we introduce another disjoint set: the set of variables.

N_V indicates the set of *variables*: one of the building blocks of μ -calculus formulas, usually denoted by capitalised letters $X, X', X_s, X_{\bar{s}}, Y, Z, \dots$.

Moreover, we consider the sets $\bar{N}_C$ and $\bar{N}_R$; $\bar{N}_C$ indicates the set of concept names N_C extended with the concepts $\top$ and $\perp$, whereas $\bar{N}_R$ indicates the set of roles that also includes the inverse roles: for each $p \in N_R$, $\bar{N}_R$ contains both p and the inverse role p^- , usually denoted by small letters $r, r', \dots$. For every $p \in N_R$, let $(p^-)^- = p$, and for each set of roles $R \subseteq \bar{N}_R$, set $R^- := \{r^- \mid r \in R\}$.

2.1 Description Logics

Description Logics are a family of logics that offer a way of capturing generalisable knowledge in logical structures. In general these logics enable us to talk about sets of objects sharing attributes by using *concept names*, or relations between two objects by using the binary *roles*. Depending on the exact fragment, other options become available, like using the (unique) names of the objects by mentioning elements from the set of *individuals* or *constants*; or counting how many nodes are related via a certain role; or more intricate combinations of any of the aforementioned constructs.

We can state what is known, i.e. facts, by using (*ground*) *atoms* or *assertions*: statements that a specific individual has a relation to another constant. If the name for an exact node is not known, a *blank node*, an element from the set of place-holders, is used instead.

Definition 2.1. A (ground) atom is an expression of the form $A(e)$ or $p(e, e')$, for $A \in N_C$, $p \in N_R$ and $\{e, e'\} \subseteq N_I \cup N_B$. In case $\{e, e'\} \subseteq N_I$, we may also refer to expressions of the form $A(e)$ or $p(e, e')$ as assertions.

Note that a set of assertions $\mathcal{A}$ can be viewed as a graph with unary and binary labels, i.e., concept and role names. Such a finite set restricted to known individuals is called an *ABox*.

Definition 2.2. An ABox (or data graph) $\mathcal{A}$ is a finite set of atoms of the form $A(a)$ or $p(a, a')$, for $A \in N_C$, $p \in N_R$ and $\{a, a'\} \subseteq N_I$. Let the domain of $\mathcal{A}$ be defined as $\Delta^{\mathcal{A}} := \{a \mid A(a) \in \mathcal{A} \text{ for some } A \in N_C\} \cup \{a, a' \mid p(a, a') \in \mathcal{A} \text{ for some } p \in N_R\}$.

We also consider *interpretations*.

Definition 2.3. An interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set (called domain) and $\cdot^{\mathcal{I}}$ is a function that maps every $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, every $p \in N_R$ to a binary relation $p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every individual and blank node $e \in N_I \cup N_B$ to an element $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. We also call $e^{\mathcal{I}}$, $A^{\mathcal{I}}$, resp. $p^{\mathcal{I}}$ the extension of e , A , resp. p . Furthermore, let $(p^-)^{\mathcal{I}} := \{(e', e) \mid (e, e') \in p^{\mathcal{I}}\}$.

Interpretations may be finite or unrestricted. One way to define *finite* interpretations is by choosing a domain of finite size. However, sometimes, we need or want to force the domain to be of infinite size, and still be able to talk about finite interpretations. That is, we want only finitely many blank nodes or individuals to participate in a concept or role name.

Definition 2.4. An interpretation is *finite* if $A^{\mathcal{I}}$ and $p^{\mathcal{I}}$ are finite sets, for each $A \in N_C$ and $p \in N_R$, and only finitely many concepts and roles have a non-empty interpretation.

As it is the natural choice to make in context of SHACL, we make the *standard name assumption* and assume that the domain of an interpretation is a subset of $N_I \cup N_B$. This means that every node is always interpreted by itself in every interpretation; $e^{\mathcal{I}} = e$ for all interpretations $\mathcal{I}$, and all $e \in N_I \cup N_B$. This has the advantage that it is more straightforward to talk about the extension of the same node in different interpretations.

To specify and name an interpretation that exactly captures a set of atoms $\mathcal{A}$, we use the *canonical interpretation*. In some chapters, it will be beneficial to use sets of atoms and interpretations interchangeably. As the following translations demonstrate, in the next two definitions, the transformations back-and-forth are very straightforward and will not cause any problems.

Definition 2.5. Given a set of atoms $\mathcal{A}$, the canonical interpretation $\mathcal{I}_{\mathcal{A}}$ is given by setting $\Delta^{\mathcal{I}_{\mathcal{A}}} := N_I \cup N_B$, $A^{\mathcal{I}_{\mathcal{A}}} := \{e \mid A(e) \in \mathcal{A}\}$ for all $A \in N_C$, $p^{\mathcal{I}_{\mathcal{A}}} := \{(e, e') \mid p(e, e') \in \mathcal{A}\}$ for all $p \in N_R$, $e \in N_I \cup N_B$.

Note that with *canonical interpretation*, we intend the reading of “a canonical way to represent a set of atoms”, and not as a synonym of a universal model. Those models will be discussed in detail in Section 2.2.

Definition 2.6. Given an interpretation $\mathcal{I}$, with a domain of finite or infinite size over $N_I \cup N_B$, let $\mathcal{A}_{\mathcal{I}}$ be the smallest subset of atoms such that $A(e) \in \mathcal{A}_{\mathcal{I}}$ if $e \in A^{\mathcal{I}}$, and $p(e, e') \in \mathcal{A}_{\mathcal{I}}$ if $(e, e') \in p^{\mathcal{I}}$.

To illustrate the notion of a canonical interpretation, we consider the following example.

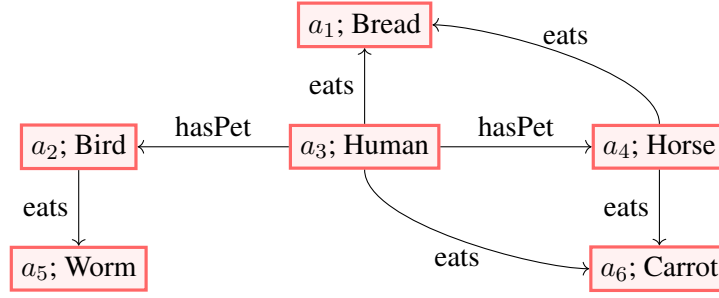


Figure 2.1: Graph representation of the ABox presented in Example 2.7.

Example 2.7. Consider the ABox $\mathcal{A}$ containing the following facts

$$\mathcal{A} := \{ \text{Bread}(a_1), \text{Bird}(a_2), \text{Human}(a_3), \text{Horse}(a_4), \text{Worm}(a_5), \text{Carrot}(a_6), \text{hasPet}(a_3, a_2), \\ \text{hasPet}(a_3, a_4), \text{eats}(a_2, a_5), \text{eats}(a_3, a_1), \text{eats}(a_3, a_6), \text{eats}(a_4, a_1), \text{eats}(a_4, a_6) \}.$$

The canonical interpretation $\mathcal{I}_{\mathcal{A}}$ is then given by $\Delta^{\mathcal{I}_{\mathcal{A}}} \supseteq \{a_1, \dots, a_6\}$, $\text{Bread}^{\mathcal{I}_{\mathcal{A}}} := \{a_1\}$, ..., $\text{Carrot}^{\mathcal{I}_{\mathcal{A}}} := \{a_6\}$, $\text{hasPet}^{\mathcal{I}_{\mathcal{A}}} := \{(a_3, a_2), (a_3, a_4)\}$ and lastly

$$\text{eats}^{\mathcal{I}_{\mathcal{A}}} := \{(a_2, a_5), (a_3, a_1), (a_3, a_6), (a_4, a_1), (a_4, a_6)\}.$$

Furthermore, in Figure 2.1, the ABox $\mathcal{A}$ /canonical interpretation $\mathcal{I}_{\mathcal{A}}$ is represented as a labelled graph. When graphically representing interpretations, domain objects are written in rectangular boxes (individuals in red boxes), followed by a semicolon and the concept names in whose interpretations it participates, if any. Roles are depicted as labelled arrows. $\heartsuit$

Relation to data graphs. RDF is intended as a human- and machine-readable language for storing data, and can in a certain sense also be thought of as labelled graphs. In this thesis, we will focus on RDF triples: (s, p, o) , containing a subject s , a role p and object o . Those triples can be thought of as atoms of binary roles in description logic data graphs. Take for instance $\text{eats}(a_2, a_5)$, one of the atoms contained in $\mathcal{A}$ as defined in Example 2.7. In triple shape, this atom looks like (a_2, eats, a_5) , that is, the translations back-and-forth are straightforward.

We also consider concept names. Take for instance the fact $\text{Worm}(a_5)$, which indicates that a_5 is an instance of ‘Worm’. As a triple, this is represented as $(a_5, \text{hasType}, \text{Worm})$, always with the role hasType . In the rest of this thesis, we will use concept names, that is, unary predicates, to denote these kind of triples.

Technically, there is no restriction on the type of objects that can occur in any of the positions of the triples. However, even when the same object, say r , is used both as a subject/object and role at the same time, then r appears both as a node name and a role name in RDF. That is, in such case, we can model this by introducing a constant a_r to talk about this role as the subject or object of a triple. Still, note that what is discussed here remains an approximation of full, untyped RDF.

Above, we considered concrete facts. However, another essential part of description logics is the possibility to capture generalised information in the shape of reasoning rules also known

as *axioms* or *general concept or role inclusions*. These axioms are collected in a *TBox*. The expressivity of these axioms depends on the particular description logic in the family of logics, of which we will introduce a handful below. We start with one of the most expressive description logics we discuss in this dissertation: $\mathcal{ALCOTQ}$, followed by the logics Horn- $\mathcal{ALCHI}$, $\mathcal{ELHI}$ and $DL\text{-}Lite_{\mathcal{R}}$.

2.1.1 The Description Logic $\mathcal{ALCOTQ}$

The description logic $\mathcal{ALCOTQ}$ is built up from the base description logic $\mathcal{ALC}$ (Attributive concept Language with Complements) extended with nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$) and qualified cardinality constraints ($\mathcal{Q}$), which allow to talk about names of specific nodes, access roles in a backwards manner, and count certain neighbours, respectively.

That is, an $\mathcal{ALCOTQ}$ concept C is defined recursively in the following way:

$$\begin{aligned} C &::= a \ (\mathcal{O}) \mid A \mid \top \mid \neg C \mid C \sqcap C \mid \forall r.C \mid \geq_n r.C \ (\mathcal{Q}) \\ r &::= p \mid p^- \ (\mathcal{I}), \end{aligned}$$

where $a \in N_I$, $A \in N_C$, $n \geq 1$ and $p \in N_R$. We use $C \sqcup C'$ resp. $\exists r.C$ as a shorthand for $\neg(\neg C \sqcap \neg C')$ resp. $\neg\forall r.\neg C$. Given any set $M = \{M_0, \dots, M_k\}$, let $\bigcap M := M_0 \sqcap \dots \sqcap M_k$.

Definition 2.8. An $\mathcal{ALCOTQ}$ TBox $\mathcal{T}$ is a set of axioms of the form $C \sqsubseteq C'$, for C and C' $\mathcal{ALCOTQ}$ concepts. These axioms are called *general concept inclusions*.

As is standard, we use $C \equiv C'$ as a shorthand for $C \sqsubseteq C'$ and $C' \sqsubseteq C$. For the semantics, we use interpretations.

Definition 2.9. Given an $\mathcal{ALCOTQ}$ TBox $\mathcal{T}$, we say an interpretation $\mathcal{I}$ is a model of $\mathcal{T}$ if for all $C \sqsubseteq C' \in \mathcal{T}$ we have $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$, where $C^{\mathcal{I}}$ is recursively defined as:

$$\begin{aligned} \top^{\mathcal{I}} &:= \Delta^{\mathcal{I}} & (\neg C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (C \sqcap C')^{\mathcal{I}} &:= C^{\mathcal{I}} \cap C'^{\mathcal{I}} \\ (\forall r.C)^{\mathcal{I}} &:= \{c \in \Delta^{\mathcal{I}} \mid (c, c') \in r^{\mathcal{I}} \text{ implies } c' \in C^{\mathcal{I}}\} \\ (\geq_n r.C)^{\mathcal{I}} &:= \{c \in \Delta^{\mathcal{I}} \mid |\{c' \in \Delta^{\mathcal{I}} \mid (c, c') \in r^{\mathcal{I}}, c' \in C^{\mathcal{I}}\}| \geq n\}. \end{aligned}$$

We use the notation $\mathcal{T} \models C \sqsubseteq C'$ in case $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. We call $C^{\mathcal{I}}$ the extension of C (in $\mathcal{I}$, might be omitted when it is clear from the context which $\mathcal{I}$ is intended). Every $\mathcal{ALCOTQ}$ TBox $\mathcal{T}$ that has at least one model is called *consistent*.

A knowledge base consist of the combination of explicit information in the form of an ABox, and implicit information, captured in a TBox. The idea of a model naturally extends to this setting: next to being a model of the given TBox, the concrete data of the ABox should also be reflected in the model.

Definition 2.10. Given an ABox $\mathcal{A}$ and an $\mathcal{ALCOTQ}$ TBox $\mathcal{T}$, the pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is an $\mathcal{ALCOTQ}$ knowledge base (KB). An interpretation $\mathcal{I}$ is a model of $\mathcal{K}$ if $\mathcal{I}$ is a model of $\mathcal{T}$ and, moreover,

- for each $A(a) \in \mathcal{A}$, we find $a \in A^{\mathcal{I}}$; and
- for each $p(a, a') \in \mathcal{A}$, we find $(a, a') \in p^{\mathcal{I}}$.

We call an $\mathcal{ALCCOIQ}$ KB $\mathcal{K}$ consistent if it has at least one model.

It is not for nothing that DLs are described as a family of logics: e.g., $\mathcal{ALC}$, which corresponds to $\mathcal{ALCCOIQ}$ where the syntax of concepts is restricted to

$$C ::= A \mid \top \mid \neg C \mid C \sqcap C \mid \forall p.C$$

where $A \in N_C$ and $p \in N_R$. That is, the features labelled with ‘ $(\mathcal{O})$ ’, ‘ $(\mathcal{I})$ ’ and ‘ $(\mathcal{Q})$ ’ are disallowed. The DLs in between $\mathcal{ALC}$ and $\mathcal{ALCCOIQ}$ of the form $\mathcal{ALCX}$, for $X \in \{\mathcal{O}, \mathcal{I}, \mathcal{Q}, \mathcal{IO}, \mathcal{IQ}, \mathcal{OQ}\}$ are constructed in a similar way: exactly the features labelled with letters occurring in X may be used in the construction of $\mathcal{ALCX}$ concepts. The semantics and other notions follow directly from restricting the definitions provided before.

Instead of the qualified cardinality constraints ‘ $\mathcal{Q}$ ’), that is, $\geq_n r.C$ in the syntax, it is also possible to consider weaker versions instead; we also consider unqualified cardinality constraints ‘ $\mathcal{N}$ ’), which allows concept constructions of the form $\geq_n r.\top$; and functionality ‘ $\mathcal{F}$ ’), which only allows unqualified counting up till one: $\leq_1 r.\top$.

For all these logics, we can now define some helpful notions. As before, we will define them for the full logic $\mathcal{ALCCOIQ}$, as for all the sublogics, the restrictions are well-defined.

Definition 2.11. Given an $\mathcal{ALCCOIQ}$ TBox $\mathcal{T}$, we say a concept C is satisfiable w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.

2.1.2 Normalised Horn- $\mathcal{ALCHIQ}$ and $\mathcal{ELHI}$

As we have seen above, the $\mathcal{O}$, $\mathcal{I}$, $\mathcal{Q}$, $\mathcal{F}$ and $\mathcal{N}$ in the name of a DL indicate which features are allowed in the concept construction. The $\mathcal{H}$ is somewhat different: it allows another type of inclusion axioms: the *role inclusion axioms*, or *role hierarchies*. That is, in an $\mathcal{ALCHIQ}$ TBox, we might find general concept inclusions of the form $C \sqsubseteq C'$, for C and C' $\mathcal{ALC}\mathcal{I}\mathcal{Q}$ concepts, and moreover role hierarchies of the form $r \sqsubseteq r'$ for $\{r, r'\} \subseteq \overline{N}_R$. If ‘ $\mathcal{I}$ ’ would not have been part of the name of the logic, we would have considered $\{r, r'\} \subseteq N_R$ instead.

However, we are not considering $\mathcal{ALCHIQ}$ TBoxes, but Horn- $\mathcal{ALCHIQ}$. One can think of the Horn-fragments as the *disjunction-free* versions of the logics; it is no longer allowed to force a model to choose between two branches as happens with for instance the set of axioms $\mathcal{T} = \{A \sqsubseteq B \sqcup C, B \sqsubseteq \neg C\}$. This will be essential when we are considering universal (core) models: as is shown in Section 4.2, every consistent Horn- $\mathcal{ALCHIQ}$ knowledge base has a unique universal core model, whereas any example containing $\mathcal{T}$ combined with for instance the data graph $\mathcal{A} = \{A(a)\}$ is already an example of a KB expressible in $\mathcal{ALC}$ that does not have a universal model (a model representing all other models, the precise definitions will be given in the next section).

Finally, we will be considering *normalised* Horn- $\mathcal{ALCHIQ}$ axioms in this thesis. This means that all axioms have to be given in a specific form, as will be provided by the following definition. For more details and a discussion regarding going beyond normalised Horn- $\mathcal{ALCHIQ}$, we refer the reader to Section 4.8.

Definition 2.12. *In a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ each concept inclusion takes one of the following forms:*

$$\begin{array}{ll} (F1) & A_0 \sqcap \dots \sqcap A_n \sqsubseteq B \\ (F2) & A \sqsubseteq \leq_1 r.B \\ (F3) & A \sqsubseteq \forall r.B \\ (F4) & A \sqsubseteq \exists r.B \end{array}$$

for $\{A, A_0, \dots, A_n, B\} \subseteq \overline{N}_C$ and $r \in \overline{N}_R$. Furthermore, $\mathcal{T}$ may contain role inclusions of the form $r \sqsubseteq r'$, for $\{r, r'\} \subseteq \overline{N}_R$.

As before, the semantics of normalised Horn- $\mathcal{ALCHIQ}$ is defined in terms of interpretations $\mathcal{I}$: a concept inclusion axiom $C \sqsubseteq C'$ is satisfied whenever $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$, and role inclusion axioms $r \sqsubseteq r'$, which are satisfied if $r^{\mathcal{I}} \subseteq r'^{\mathcal{I}}$.

All other notions naturally translate to this setting. To denote logical entailment, we may write $\mathcal{T} \models \gamma$ if every model of the TBox $\mathcal{T}$ is also a model of γ (where the latter may be any inclusion, a TBox, or an ABox).

Constructions by Kazakov [Kaz09] and Krötzsch et al. [KRH07] show that every Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ can be translated in an equivalent normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}'$ in the sense that every interpretation that is a model of $\mathcal{T}$ can be decorated with the concept names freshly introduced in the construction of $\mathcal{T}'$ to make it a model of $\mathcal{T}'$, and that every model of $\mathcal{T}'$ is a model of $\mathcal{T}$ too.

When disallowing axioms of the form (F2), we obtain TBoxes and KBs that fall within normalised $\mathcal{ELHI}$.

Definition 2.13. *In a normalised $\mathcal{ELHI}$ TBox $\mathcal{T}$ each concept inclusion takes one of the following forms:*

$$\begin{array}{ll} (F1) & A_0 \sqcap \dots \sqcap A_n \sqsubseteq B \\ (F3) & A \sqsubseteq \forall r.B \\ (F4) & A \sqsubseteq \exists r.B \end{array}$$

for $\{A, A_0, \dots, A_n, B\} \subseteq \overline{N}_C$ and $r \in \overline{N}_R$. Furthermore, $\mathcal{T}$ may contain role inclusions of the form $r \sqsubseteq r'$, for $\{r, r'\} \subseteq \overline{N}_R$.

Again, all other definitions naturally translate to this setting. Also, note that $A \sqsubseteq \forall r.B$ is equivalent to $\exists r^-.A \sqsubseteq B$, therefore, we might replace axioms of the form (F3) by axioms of the form $\exists r^-.A \sqsubseteq B$ without any consequences.

2.1.3 The Description Logic $DL\text{-}Lite_{\mathcal{R}}$

In the search for description logic fragments with good computational properties, next to the $\mathcal{EL}$ -family, the $DL\text{-}Lite$ -family was introduced [CGL⁺07]. In this thesis, we specifically focus on $DL\text{-}Lite_{\mathcal{R}}$, a fragment of $DL\text{-}Lite$ that contains role hierarchies.

Definition 2.14. A basic concept is a concept name $A \in N_C$ or an expression of the form $\exists r.\top$ such that $r \in \overline{N}_R$. A $DL\text{-}Lite_{\mathcal{R}}$ concept inclusion is an expression of the form $C \sqsubseteq C'$ or $C \sqsubseteq \neg C'$, for C and C' basic concepts. A $DL\text{-}Lite_{\mathcal{R}}$ role inclusion is an expression of the form $r \sqsubseteq r'$ or $r \sqsubseteq \neg r'$, where $\{r, r'\} \subseteq \overline{N}_R$. A $DL\text{-}Lite_{\mathcal{R}}$ TBox $\mathcal{T}$ is a set of concept and role inclusions. A $DL\text{-}Lite_{\mathcal{R}}$ knowledge base (KB) $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$ of a $DL\text{-}Lite_{\mathcal{R}}$ TBox $\mathcal{T}$ and ABox $\mathcal{A}$.

The semantics automatically follows from restricting the semantics of $\mathcal{ALCI}$ to this setting, extended with the following semantics for the role inclusion axioms: given an interpretation $\mathcal{I}$, $r \sqsubseteq r'$ is satisfied if $r^{\mathcal{I}} \subseteq r'^{\mathcal{I}}$, and $r \sqsubseteq \neg r'$ if $r^{\mathcal{I}} \cap r'^{\mathcal{I}} = \emptyset$. All other notions naturally translate to this setting too.

2.2 Universal Models

In the previous section, we have discussed which interpretations are considered models of a given knowledge base. In this section, we dive deeper in the different types of models that exist. In particular, we are interested in *universal models*; these are models that represent all models of a knowledge base (in the sense that they can be homomorphically mapped into every model of that KB) and have useful properties. They are one of the main tools used when intending to say something about all models of a KB. Note that knowledge bases expressible in $\mathcal{ALC}$ or more expressive DLs in general do not have universal models. Instead, in the context of universal models, we will be mainly interested in the DLs Horn- $\mathcal{ALCHIQ}$, $\mathcal{ELHI}$ and $DL\text{-}Lite_{\mathcal{R}}$.

We will define different morphisms as functions between interpretations. However, depending on the specific settings, we will also make use of morphisms between sets of atoms, in that case, the morphism is a function between the canonical interpretations of those sets of atoms. That is, let a homomorphism be defined as follows.

Definition 2.15. Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations. A homomorphism from $\mathcal{I}$ to $\mathcal{I}'$ is a function $h : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}'}$ such that for all $\{e, e'\} \subseteq N_I \cup N_B$, all $A \in N_C$ and all $p \in N_R$,

1. if $e \in \Delta^{\mathcal{I}} \cap N_I$, then $h(e) = e$,
2. if $e \in A^{\mathcal{I}}$, then $h(e) \in A^{\mathcal{I}'}$, and
3. if $(e, e') \in p^{\mathcal{I}}$, then $(h(e), h(e')) \in p^{\mathcal{I}'}$.

The homomorphic image of $\mathcal{I}$ under h is the interpretation $\mathcal{I}_h$ given by $\Delta^{\mathcal{I}_h} := \{h(e) \mid e \in \Delta^{\mathcal{I}}\}$, $A^{\mathcal{I}_h} := \{h(e) \mid e \in A^{\mathcal{I}}\}$, and $p^{\mathcal{I}_h} := \{(h(x), h(y)) \mid (x, y) \in p^{\mathcal{I}}\}$ for all $A \in N_C$ and $p \in N_R$.



Figure 2.2: (Strong) homomorphisms: there does not exist a strong homomorphism from $\mathcal{I}$ (left) into $\mathcal{I}'$ (right).

The main idea of a homomorphism is that all information included in $\mathcal{I}$ exists in some form, possibly decorated with more information, in $\mathcal{J}$ as well. If the extra decoration needs to be avoided, a strong homomorphism is required instead.

Definition 2.16. Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations. A strong homomorphism from $\mathcal{I}$ to $\mathcal{I}'$ is a function $h : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}'}$ such that for all $\{e, e'\} \subseteq N_I \cup N_B$, all $A \in N_C$ and all $p \in N_R$,

1. if $e \in \Delta^{\mathcal{I}} \cap N_I$, then $h(e) = e$,
2. $e \in A^{\mathcal{I}}$ iff $h(e) \in A^{\mathcal{I}'}$, and
3. $(e, e') \in p^{\mathcal{I}}$ iff $(h(e), h(e')) \in p^{\mathcal{I}'}$.

That is, a strong homomorphism really looks for a copy of the exact same structure in the other interpretation, as is illustrated in the following example.

Example 2.17. Let $\mathcal{I}$ and $\mathcal{I}'$ be given by $\Delta^{\mathcal{I}} = \{a, e\}$, $\Delta^{\mathcal{I}'} = \{a, e_1, e_2\}$, $p_1^{\mathcal{I}} = \{(a, e)\}$, $p_2^{\mathcal{I}'} = \{(a, e_1)\}$, and $p_1^{\mathcal{I}'} = \{(a, e_1), (e_1, e_2)\}$, such that $a \in N_I$ and $\{e, e_1, e_2\} \subseteq N_B$, graphically depicted in Figure 2.2. There exists one homomorphism h from $\mathcal{I}$ to $\mathcal{I}'$, given by $h(a) = a$ and $h(e) = e_1$. This is not a strong homomorphism: $(h(a), h(e)) \in p_2^{\mathcal{I}'}$, but $(a, e) \notin p_2^{\mathcal{I}}$. The function h' given by $h'(a) = e_1$ and $h'(e) = e_2$ does fulfil conditions 2. and 3. of Definition 2.16, but is no homomorphism, as $h'(a) = a$ is required. $\heartsuit$

A universal model is a model that represents all other models; when intending to answer some simple enough, positive queries (homomorphism invariant) over all models of a knowledge base, also known as the *certain answer semantics*, one of the standard approaches is to check the answer over a universal model. Because a universal model can be mapped homomorphically into every model of the given knowledge base, this means that if the required structure is present in the universal model, it is present in every model.

Definition 2.18. Given an $\mathcal{L}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, for $\mathcal{L}$ any description logic, and let $\mathcal{I}$ a model of $\mathcal{K}$. We say $\mathcal{I}$ is a universal model if for each model $\mathcal{J}$ of $\mathcal{K}$, there exists a homomorphism from $\mathcal{I}$ to $\mathcal{J}$.

The guaranteed existence of universal models is one of the important features that make Horn Description Logics so manageable.

Lemma 2.19 ([Kaz09]). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a consistent Horn- $\mathcal{ALCHIQ}$ KB, then $(\mathcal{T}, \mathcal{A})$ has a universal model.

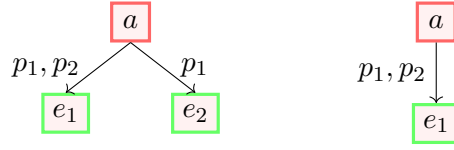


Figure 2.3: The interpretation $\mathcal{I}$ as described in Example 2.22 (left) and the core of $\mathcal{A}_{\mathcal{I}}$ (right)

Note that if $(\mathcal{T}, \mathcal{A})$ has a universal model, then it coincides with the universal model of $(\mathcal{T}^{pos}, \mathcal{A})$, where $\mathcal{T}^{pos}$ contains all axioms in $\mathcal{T}$ that do not contain $\perp$. Therefore, we may assume that there are no occurrences of $\perp$ if $(\mathcal{T}, \mathcal{A})$ is consistent when we are considering universal model constructions.

Moreover, we consider the following types of morphisms, as we will not only be considering positive, homomorphism invariant, structures over all models of a knowledge base. For more background information on different type of morphism invariants and types of universal models, we refer the reader to the work by Deutsch et al. and Ellmauthaler et al. [DNR08, EKM22].

Definition 2.20. Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations and h a homomorphism from $\mathcal{I}$ to $\mathcal{I}'$, we say

- h is an endomorphism if $\mathcal{I} = \mathcal{I}'$;
- h is an embedding if h is strong and injective; and
- h is an isomorphism if h is a surjective embedding. We write $\mathcal{I} \cong \mathcal{I}'$ to denote that $\mathcal{I}$ and $\mathcal{I}'$ are isomorphic.

With these definitions in place, we can define what a core is.

Definition 2.21. Let $\mathcal{A}$ be a set of atoms. $\mathcal{A}$ is a core when all endomorphisms on $\mathcal{I}_{\mathcal{A}}$ are embeddings. The core of a set of atoms $\mathcal{A}$ is a set of atoms $\mathcal{B} \subseteq \mathcal{A}$, such that

1. there exists an endomorphism h from $\mathcal{I}_{\mathcal{A}}$ to $\mathcal{I}_{\mathcal{B}}$;
2. $\mathcal{I}_{\mathcal{B}}$ is the homomorphic image of $\mathcal{I}_{\mathcal{A}}$ under h ; and
3. $\mathcal{B}$ is a core.

We write $\mathcal{A} \xrightarrow{\text{core}} \mathcal{B}$ to indicate that $\mathcal{A}$ is a core of $\mathcal{B}$.

Example 2.22. Let $\mathcal{I}$ be given by $\Delta^{\mathcal{I}} = \{a, e_1, e_2\}$, $p_1^{\mathcal{I}}\{(a, e_1), (a, e_2)\}$, $p_2^{\mathcal{I}} = \{(a, e_1)\}$, such that $a \in N_I$ and $\{e_1, e_2\} \subseteq N_B$, graphically depicted in Figure 2.3. The core of $\mathcal{A}_{\mathcal{I}}$ (the set of atoms corresponding to $\mathcal{I}$, see Definition 2.6) consists of the set of atoms $\{p_1(a, e_1), p_2(a, e_1)\}$.
 ♥

Lemma 2.23 ([HN92]). Each finite set of atoms has a core that is unique up to isomorphism.

2.3 Some Complexity Theory

Before we get into some basic automata theory and complexity classes, we first extend the notation convention with the following countably infinite and mutually disjoint sets used in the context of Turing machines and automata.

Q indicates the set of *states*: names for states in Turing machines and automata, usually indicated by small letters $q, q', q'', q_0, \dots$

Σ indicates an *alphabet* containing *letters* used in Turing machines and automata, usually denoted by small Greek letters $\sigma, \sigma', \dots$

Here, we will only consider non-deterministic automata over finite words, and not go into the standard definitions of (non-)deterministic Turing machines. Later in this thesis, we will make use of alternating Turing machines (Chapter 4) and two-way alternating parity tree automata (Chapter 8), and introduce those formalisms locally. For more information on different types of automata over infinite structures, and their connections to different logics and games, see for instance the book edited by Grädel et al. [GTW02].

2.3.1 Automata over Finite Words

An important aspect of SHACL is its usage of regular expressions to navigate through the data. These are used to indicate a set of paths over which other constraints can be expressed, as we will discuss in more detail in the next chapter.

Let E be any *regular expression* over some alphabet Σ , defined as

$$E ::= \epsilon \mid \sigma \mid E + E \mid E \cdot E \mid E^*$$

for ϵ the empty word and $\sigma \in \Sigma$. The language L_E defined by a regular expression E contains words $w \in \Sigma^*$, such that $L_r := \{r\}$, $L_{E+E'} := L_E \cup L_{E'}$, $L_{E \cdot E'} := \{w \cdot w' \in \Sigma^* \mid w \in L_E, w' \in L_{E'}\}$, and $L_{E^*} := \{w^* \in \Sigma^* \mid w \in L_E\}$.

We say E is a regular *path* expression, when $\Sigma = \overline{N}_R$. For each interpretation $\mathcal{I}$ and each regular path expression E over the alphabet $\overline{N}_R$, set $(e, e') \in E^{\mathcal{I}}$ if there exists $r_0 \dots r_n \in L_E$ and $\{e_1, \dots, e_n\} \subseteq \Delta^{\mathcal{I}}$ such that $(e, e_1) \in r_0^{\mathcal{I}}$, $(e_n, e') \in r_n^{\mathcal{I}}$ and for all $1 \leq i \leq n-1$, $(e_i, e_{i+1}) \in r_i^{\mathcal{I}}$. A pair $(e, e') \in E^{\mathcal{I}}$ if there exists $r_0 \dots r_n \in L_E$ and $\{e_1, \dots, e_n\} \subseteq \Delta^{\mathcal{I}}$ such that $(e, e_1) \in r_0^{\mathcal{I}}$, $(e_n, e') \in r_n^{\mathcal{I}}$ and for all $1 \leq i \leq n-1$, $(e_i, e_{i+1}) \in r_i^{\mathcal{I}}$.

For each language L_E , there exists a *non-deterministic finite automaton* $\mathcal{M} = (Q, \Sigma, q_I, \Delta, Q_F)$ (NFA) that accepts exactly all words in L_E . Here Q is a set of states, Σ the alphabet, $q_I \in Q$ and $Q_F \subseteq Q$ the initial and set of final states, and $\Delta \subseteq Q \times \Sigma \times Q$ the transition relation. A word $\sigma_0 \dots \sigma_n$ is accepted by the automaton when there exists a path $q_I, q_1, \dots, q_n$ starting in the initial state and ending in a final state $q_n \in Q_F$ such that the labels on the transitions form the word:

$$\{(q_I, \sigma_0, q_1), (q_1, \sigma_1, q_2), \dots, (q_{n-1}, \sigma_{n-1}, q_n)\} \subseteq \Delta.$$

We say $\mathcal{M}$ recognises L_E iff $\mathcal{M}$ accepts exactly all words in L_E .

2.3.2 Complexity of Reasoning

Next to considering validation of recursive SHACL in presence of ontologies, we will also study the complexity of validation, and hardness of satisfiability and containment or implication of different SHACL fragments, both in presence and without accompanying ontologies. For basic notions of complexity theory, we refer the reader to standard textbooks on the topic, e.g. the work by Papadimitriou [Pap94].

Decision problems are closely related to automata: they can be viewed as the combination of a language $L \subseteq \Sigma^*$, not per se defined by a regular expression, and the question whether an instance w belongs to this language or not. The aim is to find a procedure that decides whether $w \in L$, and complexity theory studies the size and time of this procedure.

To indicate how hard problems are, we will make use of *complexity classes*: a problem being a member of a class means that this problem can be reduced to any other problem of that class, whereas being hard for a class means that solving it is hard enough that every problem in the class can be reduced to it. More formally, a *reduction* from a language L to L' is a mapping $f : \Sigma^* \rightarrow \Sigma^*$ such that for all words w we find $w \in L$ iff $f(w) \in L'$. In this thesis, *polynomial* reductions will suffice, that means, we require f to be computable in polynomial time.

By using membership and hardness, we can under- and overestimate the set of problems belonging to certain classes. The set of problems that are both in the under- as in the overestimate are called *complete* for the class; they exactly define how hard the class is.

The most important complexity class is PTIME: the set of problems that can be decided in polynomial time. More precisely, a decision problem is in PTIME iff there exists a deterministic Turing machine (DTM) that can decide whether $w \in L$ in time (i.e., the amount of steps the machine takes) polynomial in the length of w , for each $w \in \Sigma^*$. Similarly, if this can be done by a *nondeterministic* Turing machine (NTM), the decision problem is in NP. The complement of languages decidable in NP is referred to as CONP; a language L is in CONP is deciding whether $\bar{L} := \{w \in \Sigma^* \mid w \notin L\}$ is decidable in NP. It is still unknown how these classes precisely relate to each other, that is, whether $\text{PTIME} \neq \text{NP}$, and whether $\text{NP} \neq \text{CONP}$, or not.

The class EXPTIME (resp. NEXPTIME) is defined as the set of problem that can be solved by a DTM in exponential time: L is in EXPTIME if for every $w \in \Sigma^*$ it is possible to construct a DTM (resp. NTM) that decides whether $w \in L$ in at most $2^{p(|w|)}$ steps, where $p(n)$ is a polynomial and $|w|$ the length of w . We obtain the classes 2-EXPTIME and 3-EXPTIME by replacing $2^{p(|w|)}$ by a *double*, resp. *triple* exponential time computation.

To summarise, it is the standard setting to look at how the size of the input (word) relates to the amount of steps the Turing machine needs to arrive to a decision. However, in the area of Description Logics, not only the full input to the problem is considered as the sole parameter: instead *data* and *combined complexity* are distinguished. The latter one, combined complexity, coincides with the original setting, whereas when studying the data complexity of a problem, the time the Turing machine takes is measured in the size of the data graph, that is, the amount of triples that make up the graph.

Finally, if it is impossible to provide any Turing machine, or similarly, any algorithm that can determine whether a word is contained in a language, independent of the available time, this language or problem is called *undecidable*. A well-known example of an undecidable problem is the *domino-tiling* problem [Ber66]. Naturally, if it is possible to reduce an undecidable problem like the domino-tiling problem to another problem, this other problem is undecidable too.

SHACL as a Logic

As discussed in the introduction, SHACL was introduced by the W3C to answer the need for a constraint language to manage data [W3C17]. In the rest of this chapter, we will describe how to view SHACL as a logic of constraints. The syntax will be related to the syntax used for DLs, however, this should not be confused with their separate aims. We will address both non-recursive as recursive SHACL. For more work that reflects on SHACL as logic, we refer the reader to the following articles [CRS18, ACO⁺20, BJVD22a, BJVD22b, PKM22, Ort23, OŠ24].

3.1 Non-Recursive SHACL

The SHACL standard is a document with a lot of explanations of different features and syntax in turtle that is not always straightforward to parse. The challenging task of extracting a formal syntax and semantics was first set out by Corman, Reutter and Savkovic [CRS18], and later refined by Bogaerts, Jakubowski and Van den Bussche [BJVD24].

We start by defining *shape expressions*. These are the bodies that will be used to define the constraints. That is, let

$$\varphi ::= c \mid A \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists_{\geq n} E.\varphi \mid \forall E.\varphi \mid \text{eq}(E, r) \mid \text{disj}(E, r) \mid \text{closed}(R),$$

where $c \in N_I$, $A \in N_C$, $n \geq 1$, R a finite subset of $\overline{N}_R$ and E a regular path expression. As we sometimes impose restrictions on the usage of negation in this thesis, we define for instance both ‘ $\varphi \vee \varphi$ ’ and ‘ $\varphi \wedge \varphi$ ’ explicitly in this syntax, and also ‘ $\forall \exists E.\varphi$ ’ is mentioned separately.

To give the constraints names, we recall the countably infinite set of *shape names*.

Definition 3.1. A (SHACL) constraint is an expression of the form $s \leftarrow \varphi$, for φ a shape expression, and $s \in N_S$ a shape name. Here, s is called the head of the constraint.

Let $\mathcal{C}$ be a set of constraints. Each $s \in N_S$ can be used at most one as the head of a constraint $s \leftarrow \varphi \in \mathcal{C}$.

$$\begin{aligned}
\mathcal{I}(c) &:= \{c^{\mathcal{I}}\} \\
\mathcal{I}(A) &:= A^{\mathcal{I}} \\
\mathcal{I}(\top) &:= \Delta^{\mathcal{I}} \\
\mathcal{I}(\neg\varphi) &:= \Delta^{\mathcal{I}} \setminus \mathcal{I}(\varphi) \\
\mathcal{I}(\varphi \wedge \varphi') &:= \mathcal{I}(\varphi) \cap \mathcal{I}(\varphi') \\
\mathcal{I}(\varphi \vee \varphi') &:= \mathcal{I}(\varphi) \cup \mathcal{I}(\varphi') \\
\mathcal{I}(\exists_{\geq n} E. \varphi) &:= \{e \in \Delta^{\mathcal{I}} \mid |\{e' \in \Delta^{\mathcal{I}} \mid (e, e') \in E^{\mathcal{I}} \wedge e' \in \mathcal{I}(\varphi)\}| \geq n\} \\
\mathcal{I}(\forall E. \varphi) &:= \{e \in \Delta^{\mathcal{I}} \mid \forall e' \in \Delta^{\mathcal{I}} : (e, e') \in E^{\mathcal{I}} \text{ implies } e' \in \mathcal{I}(\varphi)\} \\
\mathcal{I}(\text{eq}(E, r)) &:= \{e \in \Delta^{\mathcal{I}} \mid \{e' \in \Delta^{\mathcal{I}} \mid (e, e') \in E^{\mathcal{I}}\} = \{e' \in \Delta^{\mathcal{I}} \mid (e, e') \in r^{\mathcal{I}}\}\} \\
\mathcal{I}(\text{disj}(E, r)) &:= \{e \in \Delta^{\mathcal{I}} \mid \{e' \in \Delta^{\mathcal{I}} \mid (e, e') \in E^{\mathcal{I}}\} \cap \{e' \in \Delta^{\mathcal{I}} \mid (e, e') \in r^{\mathcal{I}}\} = \emptyset\} \\
\mathcal{I}(\text{closed}(R)) &:= \{e \in \Delta^{\mathcal{I}} \mid \text{there is no } r \in \overline{N}_R \setminus R \text{ such that } e \in (\exists r. \top)^{\mathcal{I}}\}
\end{aligned}$$

Figure 3.1: Evaluating shape expressions

Definition 3.2. Given an interpretation $\mathcal{I}$, we say a node $e \in N_I \cup N_B$ validates a shape expression φ , if $e \in \mathcal{I}(\varphi)$, where $\mathcal{I}(\varphi)$ is inductively defined in Figure 3.1.

Note that we do not (yet) allow shape names in shape expressions; this makes SHACL *non-recursive* and allows for the simple semantics we define above. Another way to define non-recursive SHACL is with shape names in shape expressions, but with a restriction on sets of constraints: there should not be any cyclic dependencies. Both definitions have the same expressivity; a simple unravelling of chains of shapes that refer to each other suffices to translate sets of constraints from the latter setting to the syntax presented here, at the cost of losing succinctness of presentation.

A SHACL document does not only contain constraints: it also specifies which constraints need to be validated in which target nodes. That is, nodes to be targetted can either be specified by mentioning a specific individual a (an element from N_I), a concept name A , which indicates that all nodes in the extension of A should validate the given constraint, or a role name $r \in \overline{N}_R$, which means that all nodes $e \in N_I \cup N_B$ that are in the extension of $\exists r. \top$ should validate the accompanying constraint. More precisely, let us define targets formally as follows.

Definition 3.3. A target is a pair (ℓ, s) , where $\ell \in N_I \cup N_C \cup \overline{N}_R$ and $s \in N_S$. A SHACL document is a pair $(\mathcal{C}, \mathcal{G})$, where $\mathcal{C}$ is a set of shape constraints and $\mathcal{G}$ is a set of targets.

SHACL documents are tested against data graphs, to test whether all nodes singled out by targets indeed conform to the accompanying constraints.

Definition 3.4. Let $\mathcal{I}$ be an interpretation and $(\mathcal{C}, \mathcal{G})$ a SHACL document, we say that $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ if for all $(a, s) \in \mathcal{G}$, we find that $a^{\mathcal{I}}$ validates φ , if for all $(A, s) \in \mathcal{G}$, we find that all $e \in A^{\mathcal{I}}$ validate φ , and for all (r, s) , if $e \in (\exists r. \top)^{\mathcal{I}}$, then e validates φ .

Since we are working in the context of DLs, we consider interpretations instead of data graphs. As interpretations and data graphs can naturally be translated into each other as described in the previous chapter, this is merely a presentation choice. Moreover, considering readability, we may also write $\mathcal{A}$ validates $(\mathcal{C}, \mathcal{G})$, for any set of atoms $\mathcal{A}$, instead of $\mathcal{I}_{\mathcal{A}}$ validates $(\mathcal{C}, \mathcal{G})$.

Example 3.5. Let $\mathcal{C}$ consist of the constraint

$$\text{VegetarianShape} \leftarrow \text{Human} \wedge \forall \text{eats}. \neg \text{Animal}$$

with target set $\mathcal{G} = \{(a_3, \text{VegetarianShape})\}$. Now consider the canonical model $\mathcal{I}_{\mathcal{A}}$ corresponding to the data graph presented in Example 2.7. We find that $\mathcal{I}_{\mathcal{A}}$ indeed validates $(\mathcal{C}, \mathcal{G})$, as both a_1 and a_6 are not in the extension of Animal . $\heartsuit$

The rest of this thesis has as main focus recursive SHACL. However, as the semantics for non-recursive SHACL as presented here coincides with all the semantics for recursive SHACL when applied to a non-recursive set of constraints, all discussions regarding semantics for combining SHACL and OWL, or rewriting techniques, naturally also apply to the non-recursive case.

3.2 Recursive SHACL

The main difference between non-recursive and recursive SHACL is that the latter has the possibility of circular definitions. This is made possible because shape names are now also allowed in the syntax of *shape expressions*, as is shown in the following definition.

$$\varphi ::= c \mid s \mid \neg s \mid A \mid \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists_{\geq n} E. \varphi \mid \forall E. \varphi \mid \text{eq}(E, r) \mid \text{disj}(E, r) \mid \text{closed}(R),$$

where $c \in N_I$, $A \in N_C$, $n \geq 1$, R a finite subset of $\overline{N}_R$ and E a regular path expression. Note that $\neg \varphi$ is replaced by $\neg s$. This does not influence the expressivity, but makes defining certain semantics later more straightforward.

Again, a *shape constraint* is an expression of the form $s \leftarrow \varphi$, for $s \in N_S$ and φ a shape expression. We call s the *head* of the constraint. Given a set of shape constraints $\mathcal{C}$, we assume each shape name s only appears as the head of one constraint; this does not influence expressibility as ‘ $\vee$ ’ may be used in φ . A *shape atom* has the form $s(e)$ with $s \in N_S$ and $e \in N_I \cup N_B$, and we call $\neg s(e)$ *negated shape atom*; a *shape literal* is a possibly negated shape atom.

Let $\text{sub}(\psi) := \{\psi\}$, for $\psi \in N_C \cup N_I \cup N_S$, or any ψ of the form $\text{eq}(E, r)$, $\text{disj}(E, r)$ or $\text{closed}(R)$, $\text{sub}(\neg s) := \{\neg s, s\}$, $\text{sub}(\varphi * \varphi') := \{\varphi * \varphi'\} \cup \text{sub}(\varphi) \cup \text{sub}(\varphi')$, for $*$ $\in \{\wedge, \vee\}$, $\text{sub}(\text{or}.\varphi) := \{\text{or}.\varphi\} \cup \text{sub}(\varphi)$, for $\text{or} \in \{\exists_{\geq n}, \forall\}$. For a constraint set $\mathcal{C}$, let $\text{sub}(\mathcal{C}) := \{\text{sub}(\varphi) \mid s \leftarrow \varphi \in \mathcal{C}\}$.

Recall that a *target* is a pair (ℓ, s) and that a SHACL document is a pair $(\mathcal{C}, \mathcal{G})$, where $\mathcal{C}$ is a set of shape constraints and $\mathcal{G}$ is a set of targets, just like in the non-recursive setting. With a slight abuse of notation, we sometimes use a shape atom $s(a)$ to specify a target (a, s) , for $a \in N_I$ and write $(\mathcal{C}, s(a))$ instead of $(\mathcal{C}, \{(a, s)\})$. We will not need abbreviations of the form $s(A)$ or $s(r)$.

The semantics of (recursive) SHACL is based on the notion of a *shape assignment*, defined as a set S of shape literals that are not contradictory, that is, there are no $\{s(e), \neg s(e)\} \subseteq S$. We view such shape assignments as three-valued: intuitively, $s(e) \in S$ means that s is validated at e , $\neg s(e) \in S$ means that s is not validated, and $\{s(e), \neg s(e)\} \cap S = \emptyset$ means that the validation of s at e is *undefined* in S . We say S is *total*, if for every shape s and node $e \in N_I \cup N_B$, we have $s(e) \in S$ or $\neg s(e) \in S$.

We say an interpretation $\mathcal{I}$ validates a set of targets $\mathcal{G}$ under a shape assignment S if the following are satisfied:

1. $s(a) \in S$, for every $(a, s) \in \mathcal{G}$,
2. $s(e) \in S$, for every $(A, s) \in \mathcal{G}$ and $e \in A^{\mathcal{I}}$, and
3. $s(e) \in S$, for every $(r, s) \in \mathcal{G}$ and $(e, e') \in r^{\mathcal{I}}$.

To define validation for a SHACL document $(\mathcal{C}, \mathcal{G})$, we still need to give semantics to constraints under a given shape assignment. For recursive SHACL, three different semantics have been advocated for in the literature: the well-founded [OŠ24], supported [CRS18], and stable semantics [ACO⁺20]. The well-founded semantics yields a unique assignment for each pair of an interpretation $\mathcal{I}$ and set of constraints $\mathcal{C}$. In contrast, the supported and the stable semantics both yield a set that may contain several *total* assignments (or none at all); in such cases, one adopts *brave* validation.

Definition 3.6. A semantics $\mathbb{S}$ for SHACL is a function that maps each pair of an interpretation and a set of constraints to a set of assignments. We say that $\mathcal{I}$ satisfies $(\mathcal{C}, \mathcal{G})$ under $\mathbb{S}$ if $\mathcal{I}$ validates $\mathcal{G}$ under some assignment in $\mathbb{S}(\mathcal{I}, \mathcal{C})$. We say that a node $e \in \Delta^{\mathcal{I}}$ satisfies a shape s w.r.t. $\mathcal{C}$ under $\mathbb{S}$ if $\mathcal{I}$ satisfies $(\mathcal{C}, s(e))$. If there exists some node in $\Delta^{\mathcal{I}}$ that satisfies s w.r.t. $\mathcal{C}$, we say that $\mathcal{I}$ satisfies s w.r.t. $\mathcal{C}$, of for simplicity, just $\mathcal{I}$ satisfies $(\mathcal{C}, s)$.

We define next the well-founded semantics, followed by the supported semantics. For the full definition of the stable semantics, we refer the reader to [ACO⁺20].

Recall that, based on the allowed syntax for SHACL documents, concrete individuals may explicitly appear inside shape expression or targets. To make presentation simpler, when we talk about checking if an interpretation $\mathcal{I}$ satisfies a document $(\mathcal{C}, \mathcal{G})$, we assume that $\mathcal{I}$ is *compatible* with $(\mathcal{C}, \mathcal{G})$. By this we mean that all individuals that appear in $(\mathcal{C}, \mathcal{G})$ also appear in $\Delta^{\mathcal{I}}$. Validation for incompatible $\mathcal{I}$ and $(\mathcal{C}, \mathcal{G})$ is considered undefined. When we quantify over possible interpretations $\mathcal{I}$ for validation or non-validation w.r.t. a document $(\mathcal{C}, \mathcal{G})$, the quantification is always limited to graphs that are compatible with $(\mathcal{C}, \mathcal{G})$.

$$\begin{aligned}
\lfloor s \rfloor_S^{\mathcal{I}} &:= \{e \mid s(e) \in S\} & \lceil s \rceil_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid \neg s(e) \notin S\} \\
\lfloor \neg s \rfloor_S^{\mathcal{I}} &:= \{e \mid \neg s(e) \in S\} & \lceil \neg s \rceil_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid s(e) \notin S\} \\
\lfloor e \rfloor_S^{\mathcal{I}} &:= \{e^{\mathcal{I}}\} & \lfloor A \rfloor_S^{\mathcal{I}} &:= A^{\mathcal{I}} & \lfloor \top \rfloor_S^{\mathcal{I}} &:= \Delta^{\mathcal{I}} & \lfloor \cdot \rfloor &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\} \\
\lfloor \varphi \wedge \varphi' \rfloor_S^{\mathcal{I}} &:= \lfloor \varphi \rfloor_S^{\mathcal{I}} \cap \lfloor \varphi' \rfloor_S^{\mathcal{I}} & \lfloor \varphi \vee \varphi' \rfloor_S^{\mathcal{I}} &:= \lfloor \varphi \rfloor_S^{\mathcal{I}} \cup \lfloor \varphi' \rfloor_S^{\mathcal{I}} & \lceil \cdot \rceil &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\} \\
\lfloor \exists_{\geq n} E. \varphi \rfloor_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid |\{e' \in \Delta^{\mathcal{I}} \mid (e, e') \in E^{\mathcal{I}} \wedge e' \in \lfloor \varphi \rfloor_S^{\mathcal{I}}\}| \geq n\} & \lceil \cdot \rceil &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\} \\
\lfloor \forall r. s \rfloor_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid \forall e' \in \Delta^{\mathcal{I}} : (e, e') \in r^{\mathcal{I}} \text{ implies } e' \in \lfloor s \rfloor_S^{\mathcal{I}}\} & \lceil \cdot \rceil &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\} \\
\lfloor \text{eq}(E, r) \rfloor_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid \{e' \mid (e, e') \in E^{\mathcal{I}}\} = \{e' \mid (e, e') \in r^{\mathcal{I}}\}\} & \lceil \cdot \rceil &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\} \\
\lfloor \text{disj}(E, r) \rfloor_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid \{e' \mid (e, e') \in E^{\mathcal{I}}\} \cap \{e' \mid (e, e') \in r^{\mathcal{I}}\} = \emptyset\} & \lceil \cdot \rceil &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\} \\
\lfloor \text{closed}(R) \rfloor_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid \text{there is no } r \in \overline{N}_R \setminus R \text{ s.t. } e \in (\exists r. \top)^{\mathcal{I}}\} & \lceil \cdot \rceil &\in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\}
\end{aligned}$$

Figure 3.2: Evaluating shape expressions: upper and lower bounds.

3.2.1 Well-founded Semantics.

For a given interpretation $\mathcal{I}$ and shape assignment S , we define two functions $\lfloor \cdot \rfloor_S^{\mathcal{I}}$ and $\lceil \cdot \rceil_S^{\mathcal{I}}$ that map shape expressions to sets of nodes as described in Figure 3.2. Let $\mathcal{C}(s) := \varphi$, where $s \leftarrow \varphi \in \mathcal{C}$ is the unique constraint in $\mathcal{C}$ with head s . Intuitively, assuming that the literals in S are true, $\lfloor \varphi \rfloor_S^{\mathcal{I}}$ is the set of nodes where φ is certainly validated, whereas $\lceil \varphi \rceil_S^{\mathcal{I}}$ is the set of nodes for which validation is possible, as it is not prevented by S . We can use $\lfloor \cdot \rfloor_S^{\mathcal{I}}$ to infer positive shape literals in the well-founded model: if $e \in \lfloor \varphi \rfloor_S^{\mathcal{I}}$, we can infer $s(e)$. In contrast, from $\lceil \cdot \rceil_S^{\mathcal{I}}$ we can infer negated shape literals: if $e \notin \lceil \varphi \rceil_S^{\mathcal{I}}$, then we can infer $\neg s(e)$. These inferred literals are added to the well-founded model incrementally, using the following one-step operators. For the positive inferences, the operator is

$$T_{\mathcal{I}, \mathcal{C}}(S) := \{s(e) \mid s \leftarrow \varphi \in \mathcal{C}, e \in \lfloor \varphi \rfloor_S^{\mathcal{I}}\}.$$

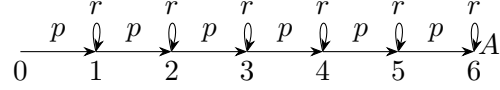
For inferring negative atoms, we rely on *unfounded sets*. For a set U of shape atoms, we let $\neg.U := \{\neg s(e) \mid s(e) \in U\}$. Given a shape assignment S , an interpretation $\mathcal{I}$ and a set of constraints $\mathcal{C}$, we call a set of shape atoms U *unfounded* w.r.t. $S, \mathcal{I}$ and $\mathcal{C}$ if $e \notin \lceil \mathcal{C}(s) \rceil_{S \cup \neg.U}^{\mathcal{I}}$ for all $s(e) \in U$. As being unfounded is preserved under union, there exists a unique *greatest unfounded set* w.r.t. $S, \mathcal{I}$ and $\mathcal{C}$ that is not subsumed by a strictly larger unfounded set w.r.t. $S, \mathcal{I}$ and $\mathcal{C}$. Let $U_{\mathcal{I}, \mathcal{C}}$ be a function that maps each shape assignment S to the greatest unfounded set w.r.t. $S, \mathcal{I}$ and $\mathcal{C}$.

Both positive and negative consequences are combined in the operator $W_{\mathcal{I}, \mathcal{C}}$. Like $T_{\mathcal{I}, \mathcal{C}}$ and $U_{\mathcal{I}, \mathcal{C}}$, it is a function that maps shape assignments to shape assignments.

$$W_{\mathcal{I}, \mathcal{C}}(S) := T_{\mathcal{I}, \mathcal{C}}(S) \cup \neg.U_{\mathcal{I}, \mathcal{C}}(S). \quad (3.1)$$

We say a function f is *monotone* if $x \subseteq y$ implies $f(x) \subseteq f(y)$. The above *immediate consequence operator* $W_{\mathcal{I},\mathcal{C}}$ is a monotone function, hence it has a unique least fixed point. We denote this fixed point $WF_{\mathcal{G},\mathcal{C}}$ and call it the *well-founded model* of $\mathcal{G}$ and $\mathcal{C}$.

Example 3.7. Consider the constraint set $\mathcal{C} = \{s \leftarrow A \vee \exists p.\neg s', s' \leftarrow \neg s \vee \exists r.s\}$. The SHACL document $(\mathcal{C}, \{s(0)\})$ is satisfiable in, for instance, the following, graphically represented interpretation $\mathcal{I}_1$.



To see this, observe the following step-by-step computation of the least fixed point $WF_{\mathcal{G}_1,\mathcal{C}}$.

$$\begin{aligned} W_{\mathcal{G}_1,\mathcal{C}}(\emptyset) &= \{s(6)\} \cup \emptyset \\ W_{\mathcal{G}_1,\mathcal{C}}^2(\emptyset) &= \{s(6)\} \cup \{\neg t(6)\} \\ W_{\mathcal{G}_1,\mathcal{C}}^3(\emptyset) &= \{s(6), s(5)\} \cup \{\neg t(6)\} \\ W_{\mathcal{G}_1,\mathcal{C}}^4(\emptyset) &= \{s(6), s(5)\} \cup \{\neg t(6), \neg t(5)\} \\ &\dots \end{aligned}$$

Note that if all p -paths from 0 to an A are infinitely long (and there exists one), this least fixed point will not terminate in a finite amount of steps. $\heartsuit$

3.2.2 Supported Model Semantics.

A supported model for constraint set $\mathcal{C}$ and a graph $\mathcal{I}$ is a total shape assignments S that satisfies $s^{\mathcal{I},S} = \lfloor \varphi \rfloor_S^{\mathcal{I}}$ for every $s \leftarrow \varphi \in \mathcal{C}$. We say $\mathcal{I}$ validates a SHACL document $(\mathcal{C}, \mathcal{G})$ under the *supported model semantics*, if there is some supported model S of $\mathcal{C}$ and a graph $\mathcal{I}$ that satisfies $\mathcal{G}$. Note that above we used $\lfloor \cdot \rfloor_S^{\mathcal{I}}$. However, both $\lfloor \cdot \rfloor_S^{\mathcal{I}}$ and $\lceil \cdot \rceil_S^{\mathcal{I}}$ can be used interchangeably here, since $\lfloor \varphi \rfloor_S^{\mathcal{I}} = \lceil \varphi \rceil_S^{\mathcal{I}}$, for any $\varphi, \mathcal{I}$, and total S .

Example 3.8. Consider the same constraint set as in Example 3.7, but now for the following, graphically represented interpretation $\mathcal{I}_2$.



As $WF_{\mathcal{G}_2,\mathcal{C}}(\emptyset) = \emptyset$, we find that $\mathcal{G}_2$ and $\mathcal{C}$ only have empty well-founded models, whereas there are two supported models S_1 and S_2 :

$$\begin{aligned} S_1 &= \{s(0), s(1), \neg t(0), \neg t(1)\} \\ S_2 &= \{\neg s(0), \neg s(1), t(0), t(1)\}. \end{aligned}$$

$\heartsuit$

3.3 Conclusion

The recent interest in SHACL stresses the importance of logic-based techniques to handle large amounts of data. Even more so given the recent changes in what large language models are capable of producing: texts and data sets that look convincing on the first glance, but that under close inspection are often full of inconsistencies or imprecisions.

Thus, in this chapter, we aimed to give an overview of SHACL as a logic, to stress its tight connections to Description Logics, and also its re-usage of the different approaches of handling negation and recursion that are well-known in the area of Logic Programming. More specifically, SHACL can be viewed as a DL where roles are *closed predicates* [Ort23, DSŠ24].

The different semantics for recursive SHACL all have their own properties: where the supported model semantics provides the most liberty in finding a satisfying shape assignment, the well-founded and stable model semantics both require some kind of minimality of positive shape atoms in their shape assignments, as a way of avoiding unwanted self justifications. Deciding validation in the supported model semantics is intractable in general, whereas validation for the stable model and well-founded semantics can be tractable in data complexity, although for the stable model semantics, this is restricted to *stratified* constraints. Intuitively, this means only positive recursive cycles are allowed. Formally, stratified SHACL is introduced in Section 4.3.

However, all these different semantics coincide in absence of negation: in that case, they all boil down to a least fixed point computation. In fact, the same holds for stratified constraints, as is discussed in more detail in the next chapter.

All chapters in the remainder of this thesis present recent findings concerning SHACL as a logic.

Validation in the Presence of Ontologies

In this chapter, we dive deeper into what would be an adequate semantics for SHACL validation in presence of constraints. That is, if we have a possibly incomplete graph and ontological knowledge that implies additional facts, can we validate given SHACL constraints over graphs containing the implied facts? Consider as an example a toy database of pet owners containing the facts $hasPetBird(linda, blu)$ and $hasPet(john, ace)$; the constraint $petOwnerShape \leftarrow \exists hasPet. \top$, which says that everyone that has a pet is a pet owner; and the target pet owners $linda$ and $john$. Clearly, we would like to leverage the knowledge that *all pet birds are pets*, written as $hasPetBird \sqsubseteq hasPet$ in description logics, which allows us to validate both targets. This type of validation is very natural, even more so in the light of the large amount of ontologies that are already being used for describing data on the web.

As shortly sketched in the introduction, a major challenge we must face is that we must simultaneously account for the open-world semantics of description logics and for the closed-world view of SHACL. The knowledge in the ontology implies additional facts, which can be added to the data graph so that it satisfies all the ontological axioms. All such possible completions are *models* that must be taken into account according to the traditional OWL/description logics semantics. However, SHACL allows for negation, which makes this *certain answer semantics* too weak, and quickly results in non validation. Update for example the toy database of pet owners we considered before to $hasPet(john, ace)$, $Hamster(ace)$, but with the constraint $petOwnerShape \leftarrow \exists hasPet. \top \wedge \forall hasPet. \neg Dangerous$. Like before, this expresses that pet owners own a pet. Moreover it is required that all pets are not dangerous. For $john$ to validate the $petOwnerShape$, it is not enough to make explicit in the ontology that hamsters are not dangerous animals; it should also be enforced that any possible pet $john$ might have cannot be dangerous.

In what follows, we will advocate for an *austere* universal model in which axioms are satisfied minimally. This is achieved by introducing as few successors as possible without losing universality, inspired by earlier work by Borgwardt and Forkel [BF19] for answering queries with negation over the description logic $\mathcal{ELH}_\perp$. We show that for ontologies in Horn- $\mathcal{ALCHIQ}$ such a model can be represented by a so-called *immediate successor function* that describes the minimal set of

facts that need to be added to satisfy the axioms at a given point of the model construction. The model itself can then be obtained in a deterministic, step-by-step fashion. Crucially, we show that the resulting model is a *core* in the traditional database sense, a result that was not yet known in the aforementioned work by Borgwardt and Forkel. We argue for evaluating SHACL over the austere universal model. This aligns with the result of the discussion by Ellmauthaler et al. [EKM22], who argue for evaluating queries with negation over core universal models.

With our semantics based on austere (i.e., core) models in place, we can tackle the problem of devising an algorithm for validation. Constructing the austere model may be infeasible in practice, since it is infinite in general. Instead, we use our finite representation of the model. Ideally, we would like to realise validation via *rewriting*. That is, we want to compile a given ontology and a set of SHACL constraints into a new set of SHACL constraints that incorporate the relevant knowledge of the ontology in such a way that the implicit facts are taken into account in validation, without having to explicitly add them to the graph. This is a well-known technique in ontology-mediated query answering, and we draw inspiration of such rewritings [EOŠ⁺12]. Such techniques are very desirable as they open the way to reuse standard SHACL validators to perform validation in the presence of ontologies. We use the finite representation of the austere universal model to construct a complex structure that stores so-called *2-types*, which intuitively represent an abstract copy of a possible object and its neighbours in the austere model of a given data graph—enriched with information about the shapes that are (not) satisfied in implied substructures. This structure is used to induce a modified set of SHACL constraints that validate over a given data graph exactly when the original constraints validate over the austere universal model of the input graph and the ontology. We first develop the technique for a positive fragment of SHACL (with minor restrictions) and then lift it to the case of *stratified SHACL* which allows both recursion and negation, but restricts their interaction.

4.1 Validation with Ontologies

We start by proposing a semantics for SHACL validation in the presence of a Horn- $\mathcal{ALCHIQ}$ ontology. More precisely, for a given TBox $\mathcal{T}$, an ABox $\mathcal{A}$, and a SHACL document $(\mathcal{C}, \mathcal{G})$, we aim to define when $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$. A natural first idea would be to follow the usual open-world semantics of Horn- $\mathcal{ALCHIQ}$ and check validation on *all models* of $\mathcal{A}$ and $\mathcal{T}$. While this works well for positive constraints, it does not yield a natural semantics in the presence of negation, as illustrated in the following simple example.

Example 4.1. Consider the ABox $\mathcal{A}$ consisting of the facts $\text{hasPet}(\text{linda}, \text{blu})$, $\text{Bird}(\text{blu})$, an empty TBox $\mathcal{T}$ and the target $s(\text{linda})$ such that s is given by

$$s \leftarrow \exists \text{hasPet}. \neg \text{Dog}.$$

Naturally, $\mathcal{A}$ validates $(\mathcal{C}, \mathcal{G})$, as *linda* indeed validates s , which corresponds to having a pet that is not a dog. Note that since we have an empty TBox, we would like to be in the usual setting of validation here. That is, one would expect $(\mathcal{T}, \mathcal{A})$ to validate $(\mathcal{C}, \mathcal{G})$. However, if we consider all possible models of $(\mathcal{T}, \mathcal{A})$, we find non-validation: the case in which *blu* is both a *Bird* and a

Dog is also a model of $(\mathcal{T}, \mathcal{A})$. This is an example that shows that the certain answer semantics, guaranteeing validation in all models, is not always meaningful in the presence of negation. ♡

The above example illustrates the problem of finding an intuitive semantics for shape expressions, or, on the same note, queries, with negation. Similar issues show up in shape expressions containing a more hidden form of negation, like equality, disjointness and the closed-operator. Also counting quantifiers have the capability of distinguishing between different universal models. Roughly speaking, adding facts to the data may cause a previously validated setting to become invalid. To this end, we aim at an intuitive semantics that coincides with the usual validation in case the TBox is empty, but also lets the TBox axioms influence the validation results in the relevant cases. As done in related settings (see e.g., [BF19, DNR08, MHS07b]) we rely on the *chase* procedure [AHV95] known from Knowledge Representation and Database Theory. The idea is that a chase procedure takes as input an ABox and TBox and iteratively applies the axioms of the TBox to the data by adding atoms over possibly fresh individuals until all the axioms in the TBox are satisfied. The result of the chase is a so-called *universal* model. Since there exists a homomorphism of all models of this type into every other model of the ABox and TBox, a universal model is often used as a representative of all models.

There are several chase variants producing universal models with different properties [DNR08]. While for positive shape expressions these differences do not lead to different validation results, shape expressions using negation can distinguish between them. Thus, the semantics we propose is based on a particular variant of the set of universal models, based on local minimality. The idea is to avoid redundant structures as much as we can: as illustrated in the next example, we do not wish to assume the existence of a second pet of *linda* if there is no need for this assumption. At the same time, we do not want to give up on the model being universal. This specific model, the *austere universal model*, will be constructed in the rest of this section.

In the following example, we show that shape expressions with negation can indeed distinguish between different universal models, and illustrate how the austere universal model does not have the redundant structures that may appear in other universal models.

Example 4.2. Consider $\mathcal{A} = \{PetOwner(linda), hasWingedPet(linda, blu), Bird(blu)\}$ and the following three axioms:

$$\begin{aligned} PetOwner &\sqsubseteq \exists hasPet, & hasWingedPet &\sqsubseteq hasPet, \\ PetOwner &\sqsubseteq \exists hasWingedPet. \end{aligned}$$

The austere universal model (right in Figure 4.1) will only add a *hasPet*-role from *linda* to *blue*, as we will see below. In contrast, the universal model obtained from the oblivious chase or Skolem chase (left in Figure 4.1) will introduce two fresh objects to satisfy the two axioms with existential restrictions.

Recall that when graphically representing interpretations, domain objects are written in rectangular boxes (individuals in red and blank nodes in green boxes), followed by a semicolon and

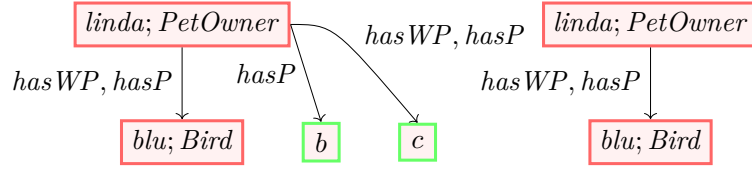


Figure 4.1: Result of oblivious or Skolem chase (left) and austere universal model (right). We use hasWP and hasP as a shorthand for hasWingedPet and hasPet.

the concept names in whose interpretations it participates, if any. Roles are depicted as labelled arrows.

Now let us consider the same shape expression and target as in the previous example: $\varphi = \exists \text{hasPet}. \neg \text{Bird}$ and $\mathcal{G} = \{\varphi(\text{linda})\}$. The target asks to validate whether *linda* has a pet that is not a bird. Clearly, the austere universal model provides the expected answer, as it does not validate $\mathcal{G}$. In contrast, the universal model on the left-hand-side of the figure, also the semantics of [MHS07b] adopted for SHACL in [SKL19], results in the unintended validation of $(\mathcal{C}, \mathcal{G})$. ♡

In the rest of this section, we will make the above precise.

4.1.1 Good Successor Configuration

To capture local minimality, we define the auxiliary notion of the *good successor configuration*. It determines, for each point e in a model, the set of fresh successor individuals and the roles connecting them to e in the austere universal model, together with the concepts that must hold at e . To describe the good successor configuration, we use *2-types*, *1½-types* and *1-types*. We say t is a 2-type if $t \in \mathcal{P}(N_C) \times \mathcal{P}(\overline{N}_R) \times \mathcal{P}(N_C)$, where we use $\mathcal{P}(X)$ to denote the power set of X . Similarly, we let u be a 1½-type in case $u \in \mathcal{P}(\overline{N}_R) \times \mathcal{P}(N_C)$. A 2-type describes a pair of nodes and the roles between them, while a 1½-type describes a set of roles leading to one node. A 1-type $c \in \mathcal{P}(N_C)$ is simply a set of concept names. For a tuple $\vec{x} = (x_1, \dots, x_n)$ and $1 \leq j \leq n$, we let $\pi_j(\vec{x}) = x_j$ be its *j-th projection*. Furthermore, we define the *inverse* function inv mapping a 2-type to a 2-type by setting $\text{inv}(t) := (\pi_3(t), (\pi_2(t))^\neg, \pi_1(t))$.

To understand the good successor configuration, assume we are building a model in a chase-like step-by-step manner. We want to introduce the children that a node c needs to satisfy the TBox. Let $c_1, \dots, c_n$ be the neighbours that c already has (either multiple neighbours already given in the ABox, or the one single parent c_1 has in the anonymous part, the part outside of the ABox: recall that a chase is building tree structures in steps). We use a set F of 2-types to describe c and its environment as follows. For each c_i , some type $t_i \in F$ describes c in relation to c_i : (i) $\pi_1(t_i)$ is the 1-type of c , that is, the concept names whose interpretations contain c , (ii) $\pi_3(t_i)$ is the 1-type of c_i , and (iii) $\pi_2(t_i)$ contains all role names connecting c and c_i . Note that, by item (i), $\pi_1(t_i) = \pi_1(t_j)$ for all $\{t_i, t_j\} \subseteq F$. The good successor configuration $\text{succ}_{\mathcal{T}}$, which is defined in the following definition, is a function that takes such an F , and returns the description of the children that c needs to satisfy all axioms in $\mathcal{T}$, as a set of 1½-types: we will add a new blank node d_i for each $u_i \in \text{succ}_{\mathcal{T}}(F)$; d_i will be connected to c via the roles in $\pi_1(u_i)$, and have 1-type $\pi_2(u_i)$.

However, the good successor configuration does not just take specific neighbourhoods of the nodes in some database as input; it is defined for all possible F 's. This makes the good successor configuration data independent. Thus, when determining the data complexity of any algorithm using the good successor configuration, its size does not have to be considered.

Moreover, the following definition defines a unique set of $I\frac{1}{2}$ -types; this is proven at the end of this subsection. Finally, note that we use expressions of the form $\exists(\prod R). \prod N$ below, for $R \subseteq \overline{N}_R$, $N \subseteq N_C$ with the following semantics:

$$(\exists(\prod R). \prod N)^{\mathcal{I}} := \{e \in \Delta^{\mathcal{I}} \mid \exists e' \in \Delta^{\mathcal{I}} \forall r \in R. (e, e') \in r^{\mathcal{I}} \wedge e' \in (\prod N)^{\mathcal{I}}\}.$$

Definition 4.3. *Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and set of 2-types F such that $\pi_1(t) = \pi_1(t')$ for all $\{t, t'\} \subseteq F$, the good successor configuration $\text{succ}_{\mathcal{T}}(F)$ is a possibly empty set of $I\frac{1}{2}$ -types u such that:*

- (R1) *If (i) $M \subseteq \pi_1(t)$ for some $t \in F$, (ii) $\mathcal{T} \models \prod M \sqsubseteq \exists(\prod R). \prod N$ for some $N \subseteq N_C$, $R \subseteq \overline{N}_R$, and (iii) for all $t \in F$, $R \not\subseteq \pi_2(t)$ or $N \not\subseteq \pi_3(t)$, then there exists $u \in \text{succ}_{\mathcal{T}}(F)$ such that $R \subseteq \pi_1(u)$ and $N \subseteq \pi_2(u)$;*
- (R2) *If $u \in \text{succ}_{\mathcal{T}}(F)$, then there exist $M \subseteq \pi_1(t)$ for some $t \in F$, such that $\mathcal{T} \models \prod M \sqsubseteq \exists(\prod \pi_1(u)). \prod \pi_2(u)$;*
- (R3) *There does not exist $\{u, u'\} \subseteq \text{succ}_{\mathcal{T}}(F)$ with $u \neq u'$ such that $\pi_1(u) \subseteq \pi_1(u')$ and $\pi_2(u) \subseteq \pi_2(u')$;*
- (R4) *If $u \in \text{succ}_{\mathcal{T}}(F)$, then $\pi_1(u) \not\subseteq \pi_2(t)$ or $\pi_2(u) \not\subseteq \pi_3(t)$, for all $t \in F$.*

A child of a 2-type t is a 2-type $t' \in \text{child}_{\mathcal{T}}(t)$ such that $\pi_1(t') = \pi_3(t)$ and $(\pi_2(t'), \pi_3(t')) \in \text{succ}_{\mathcal{T}}(\text{inv}(t))$.

Note that $\mathcal{T} \models C \sqsubseteq D$ can be checked by using the calculus developed by Eiter et al. [EOŠ⁺12].

For readability, we define the good successor configuration of a single type as the good successor configuration of the set containing only this type.

As mentioned, the good successor configuration focusses on checking which axioms are not yet satisfied in the context described by F . For each such unsatisfied axiom (R1) implies the existence of a child d_i represented by some u_i that ensures the satisfaction of the axiom. (R2) represents the other direction: for each child d_i , there must exist an axiom implying the information in the $I\frac{1}{2}$ -type u_i . Lastly, (R3) and (R4) check whether there are no superfluous children: they enforce we do not add two children d_i and d_j such that all information about d_i is subsumed by d_j , or a child d_i that is already subsumed by the environment described in F . We will use these properties extensively in Section 4.2 to prove that the austere universal model is in fact a core.

In the following example, we compute the good successor configuration for a concrete case.

Example 4.4. *Consider the axioms below, together forming the TBox $\mathcal{T}$.*

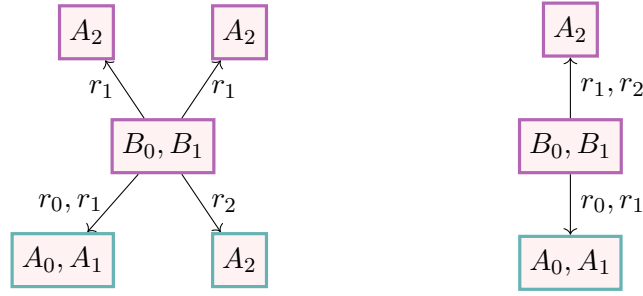


Figure 4.2: Given the TBox $\mathcal{T}$ as in Example 5.2, the good successor configurations for the sets of 2-types $F = \{(\{B_0, B_1\}, \{r_1\}, \{A_2\}), (\{B_0, B_1\}, \{r_1\}, \{A_2\})\}$ and $F' = \{(\{B_0, B_1\}, \{r_1, r_2\}, \{A_2\})\}$, are $\text{succ}_{\mathcal{T}}(F) = \{(\{r_0, r_1\}, \{A_0, A_1\}), (\{r_2\}, \{A_2\})\}$ and $\text{succ}_{\mathcal{T}}(F') = \{(\{r_0, r_1\}, \{A_0, A_1\})\}$.

$$\begin{array}{lll}
 (T1) B_0 \sqsubseteq \exists r_0. A_0 & (T4) A_0 \sqsubseteq A_1 & (T6) B_1 \sqsubseteq \exists r_2. \top \\
 (T2) B_0 \sqsubseteq \exists r_1. A_1 & (T5) r_0 \sqsubseteq r_1 & (T7) B_0 \sqsubseteq \forall r_2. A_2 \\
 (T3) B_1 \sqsubseteq \leq_1 r_1. A_1 & &
 \end{array}$$

We want to compute the good successor configuration for the following sets of 2-types: $F = \{(\{B_0, B_1\}, \{r_1\}, \{A_2\}), (\{B_0, B_1\}, \{r_1\}, \{A_2\})\}$ and $F' = \{(\{B_0, B_1\}, \{r_1, r_2\}, \{A_2\})\}$.

First, note that (T1–5) together lets us conclude $\mathcal{T} \models B_0 \sqcap B_1 \sqsubseteq \exists (r_0 \sqcap r_1). (A_0 \sqcap A_1)$. Moreover, combining (T6) and (T7) gives us $\mathcal{T} \models B_0 \sqcap B_1 \sqsubseteq \exists r_2. A_2$. Thus, by (R1), we find that there must exist $\{u, u'\} \subseteq \text{succ}_{\mathcal{T}}(F)$ such that $\{r_0, r_1\} \subseteq \pi_1(u)$, $\{A_0, A_1\} \subseteq \pi_2(u)$, $\{r_2\} \subseteq \pi_1(u')$ and $\{A_2\} \subseteq \pi_2(u')$. Note that considering $\pi_1(u) = \{r_0, r_1\}$, $\pi_2(u) = \{A_0, A_1\}$, $\pi_1(u') = \{r_2\}$ and $\pi_2(u') = \{A_2\}$ suffices, as we know that these are the axioms carrying most information that can be derived. It is easy to check that indeed $\text{succ}_{\mathcal{T}}(F) = \{u, u'\}$; see also Figure 4.2.

For computing the good successor configuration of F' , we can mostly follow the above strategy. However, u' cannot be added by (R1), as u' is already contained in the environment described by F' . This does not hold for u , so it is easy to check that $\text{succ}_{\mathcal{T}}(F') = \{u\}$. $\heartsuit$

Essential to the rest of this article is the uniqueness and existence of the good successor configuration. This implies that the construction described in the next chapter creates a unique structure.

Proposition 4.5. *Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, for each set of 2-types F , such that $\pi_1(t) = \pi_1(t')$ for all $\{t, t'\} \subseteq F$, there exists a unique good successor configuration $\text{succ}_{\mathcal{T}}(F)$.*

Proof. To show uniqueness, suppose for a given set of 2-types F , such that $\pi_1(t) = \pi_1(t')$ for all $\{t, t'\} \subseteq F$, there exist two good successor configurations $U \neq U'$. W.l.o.g., assume that there exists $u \in U$, such that $u \notin U'$. By (R2), we know that there exists $M \subseteq \pi_1(t)$ for some $t \in F$ such that $\mathcal{T} \models \bigwedge M \sqsubseteq \exists (\bigwedge \pi_1(u)). \bigwedge \pi_2(u)$. Moreover, by (R4), we can conclude that $\pi_1(u) \not\subseteq \pi_2(t)$ or $\pi_2(u) \not\subseteq \pi_3(t)$ for all $t \in F$, so we can apply rule (R1) with respect to U' : there must exist $u' \in U'$ such that $\pi_1(u) \subseteq \pi_1(u')$ and $\pi_2(u) \subseteq \pi_2(u')$. By similar reasoning, we

find that there must exist $u'' \in U$ such that $\pi_1(u') \subseteq \pi_1(u'')$ and $\pi_2(u') \subseteq \pi_2(u'')$. Combining this with (R4), we find that $\pi_1(u) = \pi_1(u'')$ and $\pi_2(u) = \pi_2(u'')$, from which we derive that $u \in U'$, which is a contradiction. Thus, $U = U'$. For existence, note that we can first add all possible $I^{1/2}$ -types that are not eliminated by (R2). It follows that (R1) is clearly satisfied. Then, remove all $I^{1/2}$ -types that are eliminated because of (R3) or (R4). $\clubsuit$

Note that it is possible to decide whether $\mathcal{T} \models C \sqsubseteq D$ for normalised Horn- $\mathcal{ALCHIQ}$ by using the inference rules from Table 2 in [EOŠ⁺12]. Thus, combining this with the algorithm described in this proof, it becomes clear how the good successor configuration can actually be computed.

4.1.2 Austere Universal Model

The good successor configuration locally describes how to satisfy the TBox axioms based on some incentives; minimality and universality. Before we use it as a building block in the austere universal model, we first want to complete $\mathcal{A}$ under all but the axioms with existential restrictions. We will use the notation $\mathcal{A}_{\mathcal{T}}$ for this completed ABox, which is in fact the least fixed point of an immediate consequence operator, a monotone function. Note that a function f is *monotone* if $x \subseteq y$ implies $f(x) \subseteq f(y)$. The point of this operator is to mimic firing the axioms without existential restrictions as soon as they become applicable. We perform the least fixed point starting from a non-empty set: the ABox. This is made precise in the following definitions.

Definition 4.6. Let X be any set and $T : X \rightarrow X$ be any monotone function, which we will call an immediate consequence operator, define for any $S \subseteq X$, $T \uparrow^\omega(S)$ as follows:

- $T \uparrow^0(S) := S$,
- $T \uparrow^{n+1}(S) := T(T \uparrow^n(S))$ for $n \geq 0$, and
- $T \uparrow^\omega(S) := \bigcup_{n=0}^{\infty} T \uparrow^n(S)$.

Definition 4.7 (Completion $\mathcal{A}_{\mathcal{T}}$). Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, define an immediate consequence operator $T_{\mathcal{T}}$ that maps a set of atoms X to a set of atoms as follows:

$$\begin{aligned}
T_{\mathcal{T}}(X) := & X \cup \{B(a) \mid \{A_0(a), \dots, A_n(a)\} \subseteq X, \mathcal{T} \models A_0 \sqcap \dots \sqcap A_n \sqsubseteq B\} \\
& \cup \{B(a) \mid \{r(b, a), A(b)\} \subseteq X, A \sqsubseteq \forall r. B \in \mathcal{T}\} \\
& \cup \{r(a, b) \mid r'(a, b) \in X, \mathcal{T} \models r' \sqsubseteq r\} \\
& \cup \{r(b, a) \mid r'(a, b) \in X, \mathcal{T} \models r' \sqsubseteq r^-\} \\
& \cup \{B_i(b) \mid \{A(a), A_1(a), \dots, A_n(a), r(a, b), B(b)\} \subseteq X, A \sqsubseteq \leq_1 r. B \in \mathcal{T}, \\
& \mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq \exists(r_1 \sqcap \dots \sqcap r_m). B_1 \sqcap \dots \sqcap B_m, A = A_j, r = r_k \text{ for some } j, k\} \\
& \cup \{r_i(a, b) \mid \{A(a), A_1(a), \dots, A_n(a), r(a, b), B(b)\} \subseteq X, A \sqsubseteq \leq_1 r. B \in \mathcal{T}, \\
& \mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq \exists(r_1 \sqcap \dots \sqcap r_m). B_1 \sqcap \dots \sqcap B_m, A = A_j, r = r_k \text{ for some } j, k\}.
\end{aligned}$$

Given any ABox $\mathcal{A}$, let $N_I(\mathcal{A})$ be the set of individuals occurring in $\mathcal{A}$. We set $A(a) \in \mathcal{A}_{\mathcal{T}}$ and $r(a, b) \in \mathcal{A}_{\mathcal{T}}$, iff $A(a) \in T_{\mathcal{T}} \uparrow^\omega(\mathcal{A})$, respectively $r(a, b) \in T_{\mathcal{T}} \uparrow^\omega(\mathcal{A})$, for all $A \in N_C$, $r \in N_R$, $\{a, b\} \subseteq N_I(\mathcal{A})$.

It can be verified how the rules in this definition mimic the completion rules for ABoxes under Horn- $\mathcal{SHIQ}$ ontologies in [EOŠ⁺12].

Example 4.8. Let $\mathcal{A} = \{B_0(a), r_0(a, b), r_2(a, b), A_0(b)\}$ and $\mathcal{T}$ as in Example 5.2. Then $\mathcal{A}_{\mathcal{T}} = \mathcal{A} \cup \{r_1(a, b), A_1(b), A_2(b)\}$. $\heartsuit$

Now we are ready to define the austere universal model. In this model, we use words to represent the elements in the anonymous parts of the model: the first letter corresponds to an individual in the data graph, followed by a letter for each node connecting the named node to the data graph. Thus, the length of the word naming an anonymous node indicates its ‘distance’ from the data graph. We mainly use this distance in the next section in which we discuss the relation to core models.

Definition 4.9. Let $(\mathcal{T}, \mathcal{A})$ be a normalised Horn- $\mathcal{ALCHIQ}$ knowledge base, and let $N_{\mathcal{K}} \subseteq N_B$ be the set of finite words of the form $ak_1 \dots k_n$, with $a \in N_I(\mathcal{A})$ such that for all $1 \leq i \leq n$, k_i is a 2-type such that the following hold:

1. $\pi_1(k_1) = \{A \mid A(a) \in \mathcal{A}_{\mathcal{T}}\}$ and $(\pi_2(k_1), \pi_3(k_1)) \in \text{succ}_{\mathcal{T}}(T_a)$, for

$$T_a = \bigcup_{b \in N_I, r(a, b) \in \mathcal{A}_{\mathcal{T}}} \{(\{A \mid A(a) \in \mathcal{A}_{\mathcal{T}}\}, \{r \mid r(a, b) \in \mathcal{A}_{\mathcal{T}}\}, \{A \mid A(b) \in \mathcal{A}_{\mathcal{T}}\})\} \\ \cup \{(\{A \mid A(a) \in \mathcal{A}_{\mathcal{T}}\}, \emptyset, \emptyset)\};$$

2. for every $1 \leq i < n$, $k_{i+1} \in \text{child}_{\mathcal{T}}(k_i)$.

We use $\text{tail}(w)$ to denote the last 2-type k_n in, and $|w| = n + 1$ as the length of a word $w \in N_{\mathcal{K}}$ of the form $ak_1 \dots k_n$.

The austere universal model $\text{can}(\mathcal{T}, \mathcal{A})$ of a normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$ is the interpretation $\text{can}(\mathcal{T}, \mathcal{A})$ with domain $\Delta^{\text{can}(\mathcal{T}, \mathcal{A})} := N_I(\mathcal{A}) \cup N_{\mathcal{K}}$ such that for all $a \in N_I(\mathcal{A})$, concept names A and roles r , the following hold:

$$a^{\text{can}(\mathcal{T}, \mathcal{A})} := a \\ A^{\text{can}(\mathcal{T}, \mathcal{A})} := \{a \in N_I \mid A(a) \in \mathcal{A}_{\mathcal{T}}\} \cup \{w \in N_{\mathcal{K}} \mid A \in \pi_3(\text{tail}(w))\} \\ r^{\text{can}(\mathcal{T}, \mathcal{A})} := \{(a, b) \in N_I \times N_I \mid r(a, b) \in \mathcal{A}_{\mathcal{T}}\} \cup \\ \{(w_1, w_2) \in (N_I \cup N_{\mathcal{K}}) \times N_{\mathcal{K}} \mid w_2 = w_1 k, r \in \pi_2(k)\} \cup \\ \{(w_2, w_1) \in (N_I \cup N_{\mathcal{K}}) \times N_{\mathcal{K}} \mid w_2 = w_1 k, r^- \in \pi_2(k)\}.$$

This is illustrated in the following example.

Example 4.10. Let $\mathcal{A} = \{B_0(a), B_1(a), r_1(a, b), r_1(a, c), A_2(b), A_2(c)\}$ and suppose $\mathcal{T}$ contains the same axioms as in Example 5.2. We find that $\mathcal{A}_{\mathcal{T}} = \mathcal{A}$, as none of the axioms there can derive any new information about ABox individuals.

Now consider the set $F_a = \{(\{B_0, B_1\}, \{r_1\}, \{A_2\}), (\{B_0, B_1\}, \{r_1\}, \{A_2\})\}$, based on the neighbours of a . The good successor configuration of this set has already been computed in 5.2: $\text{succ}_{\mathcal{T}}(F_a) = \{u, u'\}$, where $u = (\{r_0, r_1\}, \{A_0, A_1\})$ and $u' = (\{r_2\}, \{A_2\})$. Following the definition of the austere universal model, we find $\{at, at'\} \subseteq N_K$, where $t = (\{B_0, B_1\}, \pi_1(u), \pi_2(u))$ and $t' = (\{B_0, B_1\}, \pi_1(u'), \pi_2(u'))$. The structure of the anonymous part of the constructed model is now already encoded in the names of the nodes: we find for instance $(a, at) \in r_0^{\text{can}(\mathcal{T}, \mathcal{A})}$ and $at \in A_0^{\text{can}(\mathcal{T}, \mathcal{A})}$. $\heartsuit$

For a consistent knowledge base $(\mathcal{T}, \mathcal{A})$, the austere universal model $\text{can}(\mathcal{T}, \mathcal{A})$ exists and is unique, which follows directly from the uniqueness and existence of the good successor configuration and $\mathcal{A}_{\mathcal{T}}$.

Note that the austere universal model coincides with the result of the model construction presented in [EOŠ⁺12]. In that construction, $\mathcal{A}$ is first closed¹ under all implied axioms, except for the ones with existential restrictions, followed by applying all so-called ‘maximal’ axioms with existential restrictions in a chase like manner whenever applicable. To see the correspondence with our construction, it suffices to check that Definition 4.7 formalises the first step, whilst the good successor configuration captures exactly the ‘maximality’ of axioms in (R3) and the applicability in (R3) and (R4).

Corollary 4.11. *For each normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$, $\text{can}(\mathcal{T}, \mathcal{A})$ is (i) a model of $(\mathcal{T}, \mathcal{A})$ that is (ii) universal.*

This result follows directly from Proposition 2 in [EOŠ⁺12] and justifies the name ‘universal’ we gave to our construction. Finally, we define SHACL validation with Horn- $\mathcal{ALCHIQ}$ as validation of SHACL over the austere universal model.

Definition 4.12 (Validation with Horn- $\mathcal{ALCHIQ}$). *Given a normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$ and a set of targets $\mathcal{G}$, we say $(\mathcal{T}, \mathcal{A})$ validates $\mathcal{G}$ if $\text{can}(\mathcal{T}, \mathcal{A})$ validates $\mathcal{G}$.*

In general, given any semantics for SHACL validation, we define the semantics of SHACL validation with Horn- $\mathcal{ALCHIQ}$ ontologies, as validation of SHACL over the austere universal model. In particular, this approach will be used for stratified SHACL, a fragment of recursive SHACL. We refer the reader to Section 4.3 for more details.

4.2 Finite and Infinite Cores

In database theory, the property of an interpretation or structure being a core is well studied. It is a property that ultimately represents the lack of redundant structures. Therefore, this property is not only nice to have, but particularly relevant in our setting. In fact, we will show that the austere universal model is a core.

¹The conclusion of rule $\mathbf{R}_{\leq}$ should be updated to $M \sqcap M' \sqcap A \sqsubseteq \exists(S \sqcap S' \sqcap r).(N \sqcap N' \sqcap B)$, as confirmed by the authors of [EOŠ⁺12].

To start, recall that an interpretation $\mathcal{I}$ is a *core* if each endomorphism of $\mathcal{I}$ is an embedding. That is, each homomorphism of $\mathcal{I}$ into itself is both strong and injective.

Example 4.13. Recall Figure 4.1, which illustrates the results of the oblivious chase and the austere universal model of some given knowledge base. The oblivious chase model is the following set of facts:

$$X = \{ \text{PetOwner}(\text{linda}), \text{Bird}(\text{blu}), \text{hasWingedPet}(\text{linda}, \text{blu}), \text{hasPet}(\text{linda}, \text{blu}), \\ \text{hasWingedPet}(\text{linda}, b), \text{hasPet}(\text{linda}, b), \text{hasPet}(\text{linda}, c) \}$$

where b and c are unknown individuals, i.e., elements of the set N_B , whereas $\{\text{linda}, \text{blu}\} \subseteq N_I$. Thus, in each homomorphism from X to X , that is, each endomorphism of X , ‘linda’, and ‘blu’ must be mapped to themselves, which is not true for b and c . It is easy to check that $h : X \rightarrow X$ given by $h(\text{linda}) = \text{linda}$, $h(\text{blu}) = \text{blu}$, $h(b) = \text{blu}$ and $h(c) = \text{blu}$ is an homomorphism, but not injective or strong. Thus, h is not an embedding and X is not a core. Nevertheless, the austere universal model described in the same Figure 4.1, is a core, and also the core of X . $\heartsuit$

In the rest of this section, we first show that the austere universal model is the unique universal core model of a given normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$. After that, we discuss the tight connection between the austere universal model and the core chase.

Furthermore, in what follows, we also allow sets of atoms to be models. Recall the transformations presented in Chapter 2 to see that this is solely a matter of presentation: a set of atoms represents the canonical interpretation of that set.

4.2.1 Universal Core Model

To show that $\text{can}(\mathcal{T}, \mathcal{A})$ is a core, we use the notion *core cover*: an interpretation $\mathcal{I}$ has a *core cover* whenever there exist a series of finite interpretations $\mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \dots$ with $\mathcal{I} = \bigcup_{i>0} \mathcal{I}_i$, such that for all $\mathcal{I}_i$, each homomorphism $h : \mathcal{I}_i \rightarrow \mathcal{I}$ is an embedding [CKM⁺18]. Showing there exists a core cover is enough to show our structure is a core.

Definition 4.14. Set $\text{can}_n(\mathcal{T}, \mathcal{A})$ for each positive natural number n as a finite approximation of $\text{can}(\mathcal{T}, \mathcal{A})$, as follows:

1. $\Delta^{\text{can}_n(\mathcal{T}, \mathcal{A})} := \{x \in \Delta^{\text{can}(\mathcal{T}, \mathcal{A})} \mid |x| \leq n\};$
2. $a^{\text{can}_n(\mathcal{T}, \mathcal{A})} := a^{\text{can}(\mathcal{T}, \mathcal{A})}$, for all $a \in N_I$;
3. $C^{\text{can}_n(\mathcal{T}, \mathcal{A})} := \{x \in C^{\text{can}(\mathcal{T}, \mathcal{A})} \mid |x| \leq n\}$, for all $C \in N_C$;
4. $r^{\text{can}_n(\mathcal{T}, \mathcal{A})} := \{(x, y) \in r^{\text{can}(\mathcal{T}, \mathcal{A})} \mid |x| \leq n, |y| \leq n\}$, for all $r \in N_R^+$.

Note that $\text{can}_n(\mathcal{T}, \mathcal{A})$ is not per se a model of $(\mathcal{T}, \mathcal{A})$. However, if the n th approximation is a model, then $\text{can}_n(\mathcal{T}, \mathcal{A}) = \text{can}(\mathcal{T}, \mathcal{A})$.

Example 4.15. Suppose $\mathcal{A} = \{A(a)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$. Let $t = (\{A\}, \{r\}, \{A\})$ be a 2-type. First, notice that $t \in \text{child}_{\mathcal{T}}(t)$. Then the n th approximation is a chain of length n :

$$\begin{aligned}\Delta^{\text{can}_n(\mathcal{T}, \mathcal{A})} &:= \{at^i \mid i < n\} \\ A^{\text{can}_n(\mathcal{T}, \mathcal{A})} &:= \{at^i \mid i < n\} \\ r^{\text{can}_n(\mathcal{T}, \mathcal{A})} &:= \{(at^i, at^{i+1}) \mid i < n-1\},\end{aligned}$$

where t^i denotes the concatenation of i times t . ♥

Theorem 4.16. For each normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$, $\text{can}(\mathcal{T}, \mathcal{A})$ is a core.

Proof. From [CKM⁺18], Theorem 16, we learn that each interpretation that has a core cover is a core. Thus, it suffices to show that $\mathcal{I}_i = \text{can}_i(\mathcal{T}, \mathcal{A})$ is a core cover for

$$\mathcal{I} = \bigcup_{i>0} \text{can}_i(\mathcal{T}, \mathcal{A}) = \text{can}(\mathcal{T}, \mathcal{A}).$$

It is immediate that for each i , the identity mapping $h_i : \Delta^{\mathcal{I}_i} \rightarrow \Delta^{\text{can}(\mathcal{T}, \mathcal{A})}$ given by $x \mapsto x$ is an embedding.

Consider a fixed i . By induction on $|x|$ it is shown that for all homomorphisms $h : \Delta^{\mathcal{I}_i} \rightarrow \Delta^{\text{can}(\mathcal{T}, \mathcal{A})}$, $h(x) = h_i(x) = x$ holds.

- $|x| = 1$. As by definition, $\{x \in \Delta^{\mathcal{I}_i} \mid |x| = 1\} = \Delta^{\mathcal{I}_i} \cap N_C$, it follows directly that for each homomorphism h , $h(x) = x$.
- $|x| \leq n+1 \leq i$. Let us consider some $ak_1 \dots k_n \in \Delta^{\mathcal{I}_i}$. By induction hypothesis, we know that for all homomorphisms h , $h(ak_1 \dots k_{n-1}) = ak_1 \dots k_{n-1}$. As $(ak_1 \dots k_{n-1}, ak_1 \dots k_n) \in r^{\mathcal{I}_i}$ for all roles r in k_n , we also have to ensure

$$(ak_1 \dots k_{n-1}, h(ak_1 \dots k_n)) \in r^{\text{can}(\mathcal{T}, \mathcal{A})},$$

which means that we can already restrict

$$h(ak_1 \dots k_n) \in \{ak_1 \dots k_{n-2}\} \cup \{ak_1 \dots k_{n-1}k'_n \mid k'_n \in \text{child}_{\mathcal{T}}(k_{n-1})\}$$

for each possible homomorphism h . By (R4) of Definition 5.1, we find that $h(ak_1 \dots k_n) \neq ak_1 \dots k_{n-2}$. Similarly, (R3) eliminates $h(ak_1 \dots k_n) = ak_1 \dots k'_n$ for all $k'_n \neq k_n$. So, the only option is $h(ak_1 \dots k_n) = ak_1 \dots k_n$, which concludes the induction step.

Thus, h_i is the only homomorphism $h : \mathcal{I}_i \rightarrow \text{can}(\mathcal{T}, \mathcal{A})$ and the identity mapping is trivially an embedding, which concludes the proof. ♣

The following corollary can be shown with a similar proof.

Corollary 4.17. *Every finite approximation $\text{can}_n(\mathcal{T}, \mathcal{A})$ of the austere universal model is a core.*

Next to being a core, the austere universal model is also the unique core universal model, up to isomorphism, for each normalised Horn- $\mathcal{ALCHIQ}$ knowledge base. This is proven in the following theorem.

Theorem 4.18. *Each consistent normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$ has a unique (up to isomorphism) universal core model, namely the austere universal model $\text{can}(\mathcal{T}, \mathcal{A})$.*

Before we get into the actual proof, note that the definition of a core was originally solely intended for finite structures. There are multiple ways to define a core for infinite structures that all coincide for finite cases [Bau95, Bau96]. One of these approaches, requiring that each endomorphism is an embedding, is the one used up till now in this section. A stronger considered version is to also require each endomorphism to be surjective, that is, requiring that each endomorphism is an isomorphism. To distinguish this case from the core definitions discussed before, we say such an interpretation is a *strong core*. According to Bauslaugh [Bau95], these stronger cores have the nicest behaviour in the infinite case. However, they might not exist: an infinite chain of blank nodes can be a core and at the same time not a strong core.

In the following proof, we prove that the austere universal model is also a strong core. Some properties of this stronger definition then help to show that there exists indeed a unique universal core model, up to isomorphism. In line with the definition above, we say an interpretation $\mathcal{I}$ has a *strong core cover* if there exist finite interpretations $\mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \dots$ with $\mathcal{I} = \bigcup_{i>0} \mathcal{I}_i$, such that for all $\mathcal{I}_i$, each homomorphism $h : \mathcal{I}_i \rightarrow \mathcal{I}$ is an isomorphism.

Proof. First, note that the proof of Theorem 16 in [CKM⁺18] can easily be extended to also hold in this case: if an instance has a strong core cover then it is a strong core.

From the proof of Theorem 4.16, it is clear that $\mathcal{I}_i = \text{can}_i(\mathcal{T}, \mathcal{A})$ is a strong core cover for $\text{can}(\mathcal{T}, \mathcal{A})$, and thus that $\text{can}(\mathcal{T}, \mathcal{A})$ is a strong core. Now suppose there exists another universal core model $\mathcal{J}$ of the knowledge base $(\mathcal{T}, \mathcal{A})$. Because both models are universal, there exists homomorphisms in both directions. Since each endomorphism of $\text{can}(\mathcal{T}, \mathcal{A})$ must be bijective and strong, and each one of $\mathcal{J}$ injective and strong, it is straightforward to conclude that the homomorphism mapping $\mathcal{J}$ to $\text{can}(\mathcal{T}, \mathcal{A})$ must be bijective and strong too: $\mathcal{J}$ and $\text{can}(\mathcal{T}, \mathcal{A})$ are indeed isomorphic, as required. ♣

Corollary 4.19. *Each consistent normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$ has a unique universal strong core model, namely $\text{can}(\mathcal{T}, \mathcal{A})$.*

4.2.2 Core Chase

The core chase is a method that is complete for finding the unique (up to isomorphism) *finite* universal core model, whenever it exists [DNR08]. We specifically note that the core chase never produces an *infinite* structure, that is, it is only relevant to compare the austere universal model to the result of the core chase in case the core chase terminates and produces a finite universal core

model. In fact, we will see that in that case, the results coincide. Following this same paper, we will now briefly introduce the core chase.

The core chase is constructed by alternating the application of two operations: firing all not-yet satisfied axioms and taking the core of the resulting structure. This procedure is repeated until it terminates, producing a series of sets of atoms where the last set is defined to be the result of the core chase. If the series does not terminate, the core chase is undefined.

We start by formalising the first operation. For this, we consider a function that fires the right-hand side of axioms: the function f_e , defined for each $e \in N_I \cup N_B$. This function translates a concept into a set of atoms, and is inductively defined in the following way: $f_e(\top) := \emptyset$, $f_e(A) := \{A(e)\}$, $f_e(\exists(r_0 \sqcap \dots \sqcap r_n).C) := \{r_0(e, e'), r_0^-(e', e), \dots, r_n(e, e'), r_n^-(e', e)\} \cup f_{e'}(C)$ for some fresh variable e' and $f_e(C \sqcap C') := f_e(C) \cup f_e(C')$.

To determine which axioms we want to fire, we define a set of matches $m(\mathcal{T}, \mathcal{A})$ containing pairs of one or two nodes in $N_I \cup N_B$ and a Horn- $\mathcal{ALCHIQ}$ axiom. We say $(e, C \sqsubseteq D) \in m(\mathcal{T}, \mathcal{A})$ iff there exists an axiom $C \sqsubseteq D \in \mathcal{T}$ and a homomorphism h from $f_x(C)$ to $\mathcal{A}$, such that $h(x) = e$ and there is no homomorphism from $f_x(D)$ to $\mathcal{A}$ such that $h(x) = e$. Similarly, we say $((e, e'), r \sqsubseteq r') \in m(\mathcal{T}, \mathcal{A})$ iff there exists an axiom $r \sqsubseteq r' \in \mathcal{T}$ such that $r(e, e') \in \mathcal{A}$ and $r'(e, e') \notin \mathcal{A}$. No other elements are contained in $m(\mathcal{T}, \mathcal{A})$.

Combining the above, we can fully define the first operation: for an normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and ABoxes $\mathcal{A}, \mathcal{A}'$, we let $\mathcal{A} \xrightarrow{\mathcal{T}} \mathcal{A}'$ iff

$$\mathcal{A}' = \mathcal{A} \cup \bigcup_{(e, C \sqsubseteq D) \in m(\mathcal{T}, \mathcal{A})} f_e(D) \cup \bigcup_{((e, e'), r \sqsubseteq r') \in m(\mathcal{T}, \mathcal{A})} r'(e, e').$$

If there is an axiom of the form $A \sqsubseteq_{\geq 1} r.B$ in $\mathcal{T}$, such that $\{A(x), r(x, y), B(y), r(x, z), B(z)\}$ would be contained in $\mathcal{A}'$ for some $\{x, y, z\} \subseteq N_I \cup N_B$, a simple substitution of all y 's by z 's is performed to get the final $\mathcal{A}'$, assuming $y \notin N_I$.

Furthermore, for any binary relation $\rightarrow$, we use the notation $\mathcal{A} (\rightarrow)^\omega \mathcal{A}'$ to denote that there exists a natural number n such that $\mathcal{A} (\rightarrow)^n \mathcal{A}'$ and for all $\mathcal{A}''$ such that $\mathcal{A}' \rightarrow \mathcal{A}''$ we find $\mathcal{A}' = \mathcal{A}''$. With $\circ$ we denote the concatenation of two binary relations, that is, $\mathcal{A} \rightarrow \circ \rightarrow' \mathcal{B}$ iff there exists $\mathcal{A}'$ such that $\mathcal{A} \rightarrow \mathcal{A}'$ and $\mathcal{A}' \rightarrow' \mathcal{B}$, for each pair of binary relations $\rightarrow$ and $\rightarrow'$. Recall we write $\mathcal{A} \xrightarrow{\text{core}} \mathcal{B}$ in case $\mathcal{B}$ is the core of $\mathcal{A}$.

Definition 4.20 ([DNR08]). *Given a normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$, the core chase is the unique, up to isomorphism, set of atoms $\mathcal{B}$ such that $\mathcal{A} (\xrightarrow{\mathcal{T}} \circ \xrightarrow{\text{core}})^\omega \mathcal{B}$.*

Clearly, this series is not guaranteed to be finite, see for instance the following example.

Example 4.21. *Suppose $\mathcal{A} = \{A(a)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$. Then*

$$\mathcal{A} (\xrightarrow{\mathcal{T}} \circ \xrightarrow{\text{core}})^n \{A(a), r(a, a_1), A(a_1), r(a_1, a_2), A(a_2), \dots, r(a_{n-1}, a_n), A(a_n)\}$$

for each natural number n .

♡

As mentioned before, in non-terminating cases, the core chase does not produce a result. Simply taking the union of the sets in the core chase construction is in any case not a good idea. More on this topic is discussed in [CKM⁺18], combined with a proposal on how to generalise the core chase construction to an infinite setting: the stable chase. A drawback of this approach is that the result of the stable chase in general misses the universality property. It is unclear whether the stable chase for description logics might result in a non-universal model too. In case the result of the stable chase for normalised Horn- $\mathcal{ALCHIQ}$ is indeed universal, this result coincides with the austere universal model, up to isomorphism, which can be shown in a similar fashion as Theorem 4.18.

A drawback of the core chase construction is that it is unclear whether the core chase terminates and produces a result. Furthermore, the process of performing the core chase construction itself is expensive, as taking the core of a structure is expensive and happens regularly. The good news is that for normalised Horn- $\mathcal{ALCHIQ}$, the austere universal model coincides with the result of the core chase, if existent. This means the above troubles are resolved: the austere universal model is always defined and is constructed without having to take cores of structures.

Theorem 4.22. *Given a normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$ such that $\text{can}(\mathcal{T}, \mathcal{A})$ is finite, and let $\mathcal{B}$ be the unique, up to isomorphism, structure such that $\mathcal{A}(\xrightarrow{\mathcal{T}} \circ \xrightarrow{\text{core}})^\omega \mathcal{B}$, then*

$$\mathcal{B} \cong \text{can}(\mathcal{T}, \mathcal{A}).$$

Proof. We note that $\mathcal{B}$ is a core by definition, and the same holds for $\text{can}(\mathcal{T}, \mathcal{A})$ by Theorem 4.16. Since both are cores, it suffices to show that there exists homomorphisms in both directions.

Note that $\text{can}(\mathcal{T}, \mathcal{A})$ is a model by Proposition 4.11. As $\mathcal{B}$ is a universal model [DNR08], it follows that there exists a homomorphism from $\mathcal{B}$ into each model, including $\text{can}(\mathcal{T}, \mathcal{A})$. By Proposition 4.11 we also have that $\text{can}(\mathcal{T}, \mathcal{A})$ is universal, which means also the other required homomorphism must exist. ♣

4.3 Stratified SHACL

In Section 4.1, we described non-recursive SHACL validation in presence of ontologies, however, there is no reason that the same reasoning does not apply to argue for the same semantics of recursive SHACL in presence of ontologies. For this reason, we will continue the rest of the chapter introducing a fragment of SHACL that allows recursion: *stratified SHACL*. As discussed in Chapter 3, there is no unique way to extend the simple semantics of non-recursive SHACL to the recursive setting. Different alternatives have been explored in the literature, like the stable model semantics [ACO⁺20], the supported model semantics [CRS18] and the well-founded semantics [OŠ24]. The semantics we will discuss here is the *least fixed point* semantics, for stratified constraints. The main advantage of considering stratified constraints is that the stable and well-founded semantics coincide with the least fixed point semantics, which is computable in a more straightforward way. In this case, this assignment, also captured by $WF_{\mathcal{G}, \mathcal{C}}$, is sometimes called the *least* or *perfect* model of $\mathcal{G}$ and $\mathcal{C}$. In particular, it is also a total model, a supported model, and the unique stable model of $\mathcal{G}$ and $\mathcal{C}$.

$$[\exists(\prod R).\varphi]_S^{\mathcal{I}} := \{e \in \Delta^{\mathcal{I}} \mid \exists e' \in \Delta^{\mathcal{I}} \forall r \in R : (e, e') \in r^{\mathcal{I}} \wedge e' \in [\varphi]_S^{\mathcal{I}}\}$$

Figure 4.3: Evaluating shape expressions

In the rest of this chapter, we impose some additional restrictions on the definition of a shape expression and consider only a fragment of recursive SHACL.

That is, let φ be defined in the following way

$$\varphi ::= a \mid s \mid \neg s \mid A \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists(\prod R).\varphi \mid \exists E.\varphi \mid a \wedge \text{eq}(E, E') \mid a \wedge \text{disj}(E, E'),$$

where $a \in N_I$, $s \in N_S$, $A \in \overline{N}_C$, $R \subseteq \overline{N}_R$, and E and E' regular path expressions. Here, we use ‘ $\exists$ ’ as a shorthand for ‘ $\exists_{\geq 1}$ ’. The rest of the definitions of recursive SHACL follow the lines set out in Chapter 3, except that in this chapter, we allow multiple constraints with the same shape name in the head in a set of constraints. The intended reading of this is as followed: $\{s \leftarrow \varphi_1, \dots, s \leftarrow \varphi_n\}$ can be replaced by one big disjunction: $\{s \leftarrow \bigvee_{1 \leq i \leq n} \varphi_i\}$, to recover the original setting.

First, note that our versions of path equality and disjointness, $a \wedge \text{eq}(E, E')$ and $a \wedge \text{disj}(E, E')$, distinguish themselves from the version in Chapter 3 and the SHACL standard in two ways. First, the standard requires $E' \in \overline{N}_R$. We generalise this and allow E' to be any regular expression. Secondly, for technical reasons, we add a ‘guard’ in the form of an individual $a \in N_I$. Furthermore, this fragment lacks counting over regular path expressions and closure constraints. We note that our results do not simply extend to the setting without these restrictions; especially counting over regular paths is a feature that requires a lot of attention in presence of ontologies, see for instance Chapter 6. We refer the reader to Section 4.8 for a more extensive discussion. Lastly, we increase the expressivity of SHACL slightly by adding conjunction of roles in the form $\exists R.\varphi$, for R a conjunction of roles in $\overline{N}_R$, which we use to reduce Horn-*ALCHIQ*-reasoning combined with SHACL to plain SHACL reasoning. For the extension of $[\cdot]_S^{\mathcal{I}}$ to this conjunction, we refer to Figure 4.3, where $[\cdot] \in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\}$.

Finally, we assume that all targets to be of the shape (a, s) , for $a \in N_I$, or similarly, $s(a)$.

4.3.1 Least Fixed Point Semantics

In this part of the thesis, we are particularly interested in shape assignments where each shape atom has a proper justification. That is, we do not wish to consider any set of shape atoms S , but put some constraints on the type of shape assignments we accept. One option would be to resort to the full stable model semantics, but only if we want to give up on polynomial time data complexity. Thus, we focus on a more straight-forward semantics, which is admitted by *stratified* sets of constraints. This is a fragment of recursive SHACL which supports negation to a certain extent, but restricts the full combination of recursion and negation. To this end and following the logic programming literature [ABW88], a partition of constraints (stratification) is defined such that a justified shape assignment can be constructed by processing each partition

individually. What makes this semantics particularly suitable is that it coincides with the stable model and well-founded semantics on stratified sets of constraints [ACO⁺20, OŠ24], while being more straightforward to compute than those semantics.

Definition 4.23. *We say a shape name s occurs negatively in a shape constraint $s' \leftarrow \varphi$ if $\neg s$ occurs in φ . We say a shape name s is defined in a set $\mathcal{C}$ of constraints if $s \leftarrow \varphi \in \mathcal{C}$ for some φ .*

A set $\mathcal{C}$ of constraints is stratified if it can be partitioned into sets $\mathcal{C}_0, \dots, \mathcal{C}_k$, called strata, such that, for all $0 \leq i \leq k$, the following hold.

1. *If $i < k$ and s' occurs in φ for some $s \leftarrow \varphi \in \mathcal{C}_i$, then s' is not defined in $\mathcal{C}_{i+1} \cup \dots \cup \mathcal{C}_k$.*
2. *If s' occurs negatively in φ for some $s \leftarrow \varphi \in \mathcal{C}_i$, then s' is not defined in $\mathcal{C}_i \cup \dots \cup \mathcal{C}_k$.*

A set of constraints is stratified if it admits a stratification.

Without loss of generality, we can assume that all constraints with the same head are defined in the same stratum. An easy way to see this is noticing that replacing a set of constraints with the same head by a single constraint of which the body is formed by taking a disjunction of all bodies cannot influence which interpretations are satisfied.

A standard way to obtain a stratification is to define a dependency graph (V, E, E^*) with the set of shape names used as the nodes V . In this graph, there are two types of edges: marked edges E^* and standard edges E , such that $E^* \subseteq E$. We use the standard edges to mark that s occurs in a shape constraint with head s' . In that case, we set $(s, s') \in E$. If this is not just any occurrence, but s occurs *negatively* in a shape constraint with head s' , then we also set $(s, s') \in E^*$. The lowest stratum is the biggest set of nodes X such that if $s \in X$ and $(s', s) \in E$, then $s' \in X$ and for all $s \in X$, there does not exist any $s' \in V$ such that $(s', s) \in E^*$. To find the next lowest stratum, simply remove all shape names in X from the dependency graph and repeat the above. This may be repeated until all shape names are assigned a stratum.

Given any stratification of a set of constraints, we compute the shape assignment of stratified SHACL with a least fixed point operator over each stratum, with negation as failure to compute the opposite in an earlier stratum. To do so, recall the definition of the immediate consequence operator $T_{\mathcal{I}, \mathcal{C}}$ for the well-founded semantics as defined in Chapter 3, that, given a shape assignment S , adds new shape atoms to satisfy the constraints that are fired by the constraints in $\mathcal{C}$ based on S and $\mathcal{I}$.

The following two propositions are a direct consequence of the characterisations from [ABW88] in the context of stratified logic programs. Here, we will again use the definition of the least fixed point starting from a given set. We refer the reader back to Definition 4.6 for the precise definition.

Proposition 4.24. *If $\mathcal{C}$ is a constraint set that does not define any shape names that occur negatively in $\mathcal{C}$, then the following hold:*

1. $T_{\mathcal{I},\mathcal{C}}$ is monotonic, i.e. if $S \subseteq S'$, then $T_{\mathcal{I},\mathcal{C}}(S) \subseteq T_{\mathcal{I},\mathcal{C}}(S')$;
2. $T_{\mathcal{I},\mathcal{C}}$ is finitary, i.e. $T_{\mathcal{I},\mathcal{C}}(\bigcup_{n=0}^{\infty} S_n) \subseteq \bigcup_{n=0}^{\infty} T_{\mathcal{I},\mathcal{C}}(S_n)$ for all infinite sequences $S_0 \subseteq S_1 \subseteq \dots$;
3. $T_{\mathcal{I},\mathcal{C}}$ is growing, i.e. $T_{\mathcal{I},\mathcal{C}}(S_2) \subseteq T_{\mathcal{I},\mathcal{C}}(S_3)$ for all S_1, S_2, S_3 such that $S_1 \subseteq S_2 \subseteq S_3 \subseteq T_{\mathcal{I},\mathcal{C}} \uparrow^{\omega}(S_1)$.

Proposition 4.25. *If $\mathcal{I}$ is an interpretation and $\mathcal{C}_0, \dots, \mathcal{C}_k$ is a stratification of $\mathcal{C}$, then each $T_{\mathcal{I},\mathcal{C}_0}, \dots, T_{\mathcal{I},\mathcal{C}_k}$ is monotone, finitary, and growing. Thus, for any shape assignment S and each $0 \leq j \leq k$, $T_{\mathcal{I},\mathcal{C}_j} \uparrow^{\omega}(S)$ is a fixpoint of $T_{\mathcal{I},\mathcal{C}_j}$.*

Based on the above, we can now define the computation of the desired shape assignment along a stratification $\mathcal{C}_0, \dots, \mathcal{C}_k$ of $\mathcal{C}$.

Definition 4.26. *Assume $\mathcal{I}$ is an interpretation, $\mathcal{C}$ is a stratified set of constraints, and let $\mathcal{C}_0, \dots, \mathcal{C}_k$ be a stratification of $\mathcal{C}$. Then let*

$$\begin{aligned} M_0 &= T_{\mathcal{I},\mathcal{C}_0} \uparrow^{\omega}(\emptyset) \\ M_i &= T_{\mathcal{I},\mathcal{C}_i} \uparrow^{\omega}(M_{i-1}) \quad \text{for each } 1 \leq i < k. \end{aligned}$$

We call M_k be the perfect assignment for $\mathcal{C}$ and $\mathcal{I}$, and use the notation $PA(\mathcal{C}, \mathcal{I}) := M_k$. An interpretation $\mathcal{I}$ (resp., ABox $\mathcal{A}$) validates a SHACL document $(\mathcal{C}, \mathcal{G})$ if $\mathcal{G} \subseteq PA(\mathcal{C}, \mathcal{I})$ (resp., $\mathcal{G} \subseteq PA(\mathcal{C}, \mathcal{I}_{\mathcal{A}})$).

A well-known result is that the particular stratification chosen does not matter: any stratification will give the same perfect assignment [ABW88]. Given this semantics, it is now straightforward to extend Definition 4.12 to recursive SHACL documents.

Definition 4.27. *Given a normalised Horn-ALCHIQ TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and a stratified SHACL document $(\mathcal{C}, \mathcal{G})$, we say $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ if $\text{can}(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$.*

4.3.2 Normal Form

A nice feature of the SHACL fragment introduced at the start of this section, is that it has a rather succinct normal form with the same expressivity. In the rest of the paper, we will assume that all the given sets of constraints are in normal form. This will be particularly useful in the following two sections, restricting the shapes of constraints we need to consider.

Definition 4.28. *A SHACL constraint is in normal form if it has one of the following forms*

$$\begin{aligned} (NC1) \quad s &\leftarrow a & (NC2) \quad s &\leftarrow A & (NC3) \quad s &\leftarrow s' \\ (NC4) \quad s &\leftarrow s' \wedge s'' & (NC5) \quad s &\leftarrow \exists(\bigcap R).s' & (NC6) \quad s &\leftarrow \neg s', \end{aligned}$$

where $a \in N_I$, $R \subseteq \overline{N}_R$, $\{s, s', s''\} \subseteq N_S$ and $A \in \overline{N}_C$.

We use the term *negative constraints* to refer to constraints of the form (NC6) and *positive constraints* for constraints of the form (NC1)–(NC5). Clearly, each set of positive constraints is automatically stratified. Furthermore, for simplicity, if $R = \{r\}$, we may write $s \leftarrow \exists r.s'$ instead of $s \leftarrow \exists(\bigcap R).s'$.

We can indeed rewrite parts of SHACL into this restricted, normalised, form of SHACL, as formalised below.

Proposition 4.29. *Let $\mathcal{C}$ be a stratified set of constraints expressed in the fragment of SHACL introduced in Section 4.3. Then $\mathcal{C}$ can be translated in time polynomial in $|\mathcal{C}|$ into a stratified set of constraints $\mathcal{C}'$ in normal form such that, for every $\mathcal{G}$ and every interpretation $\mathcal{I}$ we have that $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{I}$ validates $(\mathcal{C}', \mathcal{G})$. This holds under the least fixed point, well-founded and stable model semantics.*

In particular, the above statement holds for SHACL validation in presence of ontologies: for each normalised Horn- $\mathcal{ALCHIQ}$ knowledge base $(\mathcal{T}, \mathcal{A})$, we have $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}', \mathcal{G})$.

Proof. First, note that the least fixed point, well-founded and stable model semantics coincide for stratified sets of constraints [ACO⁺20, OŠ24]. Thus, it will be sufficient to consider the least fixed point semantics. Note that this is only defined for stratified constraints.

We first recursively introduce fresh shapes for sub-expressions that appear in constraints, as happens in the construction presented by Andresel et al. [ACO⁺20]. Now, what is left to show is that also constraints of the form $s \leftarrow a \wedge \text{eq}(E, E')$, $s \leftarrow a \wedge \text{disj}(E, E')$ and $s \leftarrow \exists E.s'$ can be translated into constraints in normal form in polynomial time.

- $s \leftarrow \exists E.s'$. Suppose $\mathcal{M} = (Q, \Sigma, q_I, \Delta, Q_F)$ is the automaton recognising E . Take fresh shape names s_q for each $q \in Q$ and add the following constraints

$$\begin{aligned} s &\leftarrow s_{q_I} \\ s_q &\leftarrow \exists r.s'_q \quad \text{if } (q, r, q') \in \Delta \\ s_{q_F} &\leftarrow s' \quad \text{for all } q_F \in Q_F. \end{aligned}$$

Here, the idea is to encode the states of the automaton using shape names. We let every state that is an s' be a final state, and walk through the automaton backwards to figure out which states can be labelled with an initial state. Then those states are exactly the states that can reach one of the nodes labelled with a final state via E . Thus, a node is assigned s_{q_I} iff that node has an E -path to a node that is an s' .

- $s \leftarrow a \wedge \text{eq}(E, E')$. Suppose $\mathcal{M} = (Q, \Sigma, q_I, \Delta, Q_F)$ and $\mathcal{M}' = (Q', \Sigma, q'_I, \Delta', Q'_F)$ are the automata recognising E and E' respectively, such that $Q \cap Q' = \emptyset$. Take fresh shape names s_{error} , s_{noerror} , s_{pos} , s_{neg} , s_F , $s_{F'}$ and s_q for each $q \in Q \cup Q'$ and add the following constraints

$$\begin{array}{ll}
s_q \leftarrow a & \text{for each } s_q \in \{s_{q_I}, s_{q'_I}\} \\
s_{q'} \leftarrow \exists r^-.s_q & \text{for each } (q, r, q') \in \Delta \cup \Delta' \\
s_F \leftarrow s_q & \text{for each } q \in Q_F \\
s_{F'} \leftarrow s_q & \text{for each } q \in Q_{F'} \\
s_{pos} \leftarrow s_F \\
s_{pos} \leftarrow s_{F'} \\
s_{neg} \leftarrow \neg s_F \\
s_{neg} \leftarrow \neg s_{F'} \\
s_{error} \leftarrow s_{pos} \wedge s_{neg} \\
s_{error} \leftarrow \exists r.s_{error} & \text{for each } r \in \Sigma \\
s_{noerror} \leftarrow \neg s_{error} \\
s \leftarrow a \wedge s_{noerror}.
\end{array}$$

In this case, because of the ‘guard’ a , there is only one node considered to be the initial state of both automata. From there, we assign the states using their encoding in shape names. This time, we consider the states of the automaton from initial state towards, possibly, a final state. After finishing assigning states, it is stored which nodes are labelled with (at least) one of the final states of any of the two automata, by labelling them with s_F or $s_{F'}$. We can use negation to detect the nodes that are reachable at some final state of one automaton, but at no final state of the other, and an ‘error’ shape name to propagate the existence of such a node witnessing that $\text{eq}(E, E')$ is not satisfied at a .

- $s \leftarrow a \wedge \text{disj}(E, E')$. We use a similar solution as for $s \leftarrow a \wedge \text{eq}(E, E')$: all shapes remain except for the ones with s_{pos} or s_{neg} in the head, and the first error shape s_{error} is replaced by $s_{error} \leftarrow s_F \wedge s_{F'}$

Note that the automata can be constructed in polynomial time, which is a standard result in the literature, see for instance [HSW01].

Correctness of this construction follows from an induction on the construction of shape expressions. More specifically, for each (non-normalised) constraint $s \leftarrow \varphi$ and set of constraints $\mathcal{C}$ and each interpretation $\mathcal{I}$, we prove that the shape assignment S of $\mathcal{C} \cup \{s \leftarrow \varphi\}$ on the interpretation $\mathcal{I}$ under the least fixed point semantics coincides with the shape assignment S' of $\mathcal{C} \cup \mathcal{C}_s$ on the interpretation $\mathcal{I}$ under the well-founded assignment, after restricting it to the shape names that are defined in $\mathcal{C} \cup \{s \leftarrow \varphi\}$, where $\mathcal{C}_s$ is the set of normalised constraints obtained by applying the above construction. For the constructs discussed in the proof by Andresel et al. the induction steps are straightforward. The other steps we discuss below.

- $s \leftarrow \exists E.s'$. First, note that by the induction hypothesis, we know that $s'(e) \in S$ iff $s'(e) \in S'$, for all $e \in \Delta^{\mathcal{I}}$. Thus, $s'(e) \in S$ iff $s_{q_F}(e) \in S'$ for all $q_F \in Q_F$. First, assume there is no $s \leftarrow \varphi' \in \mathcal{C}$ used to derive $s(d)$ in S or S' (in that case, in that case, it is immediate that $s(d) \in S$ iff $s(d) \in S'$). So $s(d) \in S$ iff there exists $s'(e) \in S$ such that there exists a path $r_1 \cdot \dots \cdot r_n \in L_E$ connecting d to e in $\mathcal{I}$. This holds iff there is a series of states $q_I, q_1, \dots, q_n$ of the automaton $\mathcal{M}$ recognising L_E such that

$q_n \in Q_F$ and $\{(q_I, r_1, q_1), \dots, (q_{n-1}, r_n, q_n)\} \subseteq \Delta$. As $s_{q_n}(e) \in S'$, a simple induction argument suffices (using the least fixed point semantics) to conclude that this holds iff also $s_{q_I}(d) \in S'$, and thus $s(d) \in S'$.

- $s \leftarrow a \wedge \text{eq}(E, E')$. First, note that a simple induction (using the least fixed point semantics) suffices to conclude that with the first two constraint types, we propagate to every individual the set of states the two automata $\mathcal{M}$ and $\mathcal{M}'$ can reach in that state from a . Thus, we find $s_q(e) \in S'$ for $q \in Q_F$, and thus $s_F(e) \in S'$ iff e can be reached by some path $r_1 \cdot \dots \cdot r_n \in L_E$ from a , and similarly for $\mathcal{M}'$. Thus, the last thing to do is to see whether there exists a node that is the final state of one of the two automata (thus finding $s_{pos}(e) \in S'$ for some e), but not any final state of the other automaton (thus also finding $s_{neg}(e) \in S'$ for the same e). Given that a node is assigned s_F as soon as one of the final states of $\mathcal{M}$ is reached, this means that we find $s_{pos}(e) \in S'$ iff this node is assigned at least one final state of at least one of the automata, and $s_{neg}(e) \in S'$ iff this node does not see any final state of at least one of the automata. Note that these constraints correspond to $s_{error} \leftarrow (s_F \wedge \neg s_{F'}) \vee (\neg s_F \wedge s_{F'})$, but this constraint is not in normal form. To express this in the normal form, we use the fact that a node cannot be assigned s_F and $\neg s_F$ at the same time, thus $s_{error} \leftarrow s_{neg} \wedge \neg s_{pos}$ suffices. The last thing to do is to bring this information back towards a , with the constraints $s_{error} \leftarrow \exists r. s_{error}$ for each r appearing in the automata, and we find $s(a) \in S$ iff $s(a) \in S'$.
- $s \leftarrow a \wedge \text{disj}(E, E')$. A similar argument applies.

This concludes the induction argument. ♣

4.4 Rewriting for Positive SHACL

Now we will return to problem of SHACL validation in presence of ontologies. In Section 4.1, we defined the semantics of SHACL in presence of ontologies as SHACL validation over the core universal model. This was independent of the chosen semantics for SHACL. As we decided on the least fixed point semantics for SHACL in the previous section, it is now clear what the intended meaning is of a knowledge base validating a SHACL document.

However, we also care about computability. To this end, we do not wish to materialise the full, possibly infinite, core universal model. Instead, we will bring SHACL validation over the core universal model back to SHACL validation over the enriched ABox $\mathcal{A}_{\mathcal{T}}$, as defined in Definition 4.7, by rewriting the set of constraints. More precisely, given a TBox $\mathcal{T}$ and a set of stratified constraints $\mathcal{C}$, we want to compile $\mathcal{T}$ and $\mathcal{C}$ into a new set $\mathcal{C}_{\mathcal{T}}$ of stratified constraints so that for every ABox $\mathcal{A}$ consistent with $\mathcal{T}$, and every target $\mathcal{G}$, we have

$$(\mathcal{T}, \mathcal{A}) \text{ validates } (\mathcal{C}, \mathcal{G}) \text{ iff } \mathcal{A}_{\mathcal{T}} \text{ validates } (\mathcal{C}_{\mathcal{T}}, \mathcal{G}).$$

This will be achieved by means of an inference procedure that uses a collection of inference rules to capture the possible “propagation” of shape names in the anonymous part of the austere universal model. In the following, the rewriting procedure for SHACL without any occurrence of negation is discussed, followed by the general case of recursive SHACL.

4.4.1 Rewriting Algorithm

The idea of the rewriting is that we can encode all reasoning of axioms in the constraints. The core of our technique is an inference procedure that derives a set of quadruples (t, P, Q, H) , where t a 2-type, as introduced at the start of Subsection 4.1.1. Intuitively, it is possible to derive (t, P, Q, H) iff an object that satisfies all concept names in $\pi_1(t)$, that satisfies all expressions (assumptions) in P , and does not satisfy any single expression in Q , must validate all shape names in H . Note that there may be some overlap in the information stored in t and P .

Definition 4.30. Let $\top, a, \exists r.s$, be basic shape expressions, for $r \in \overline{N}_R$, $s \in N_S$, and $a \in N_I$. Moreover, let $\exists(\prod R). \prod N$ be called a basic concept expression, for each $R \subseteq \overline{N}_R$ and $N \subseteq N_C$.

Definition 4.31. A 2-type t is called locally consistent if the following are satisfied.

- If $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq B$ and $\{A_1, \dots, A_n\} \subseteq \pi_i(t)$, then $B \in \pi_i(t)$, for $i \in \{1, 3\}$.
- If $\mathcal{T} \models \prod R \sqsubseteq r'$ and $R \subseteq \pi_2(t)$, then $r' \in \pi_2(t)$.
- If $A \sqsubseteq \forall r.B \in \mathcal{T}$, $A \in \pi_1(t)$ and $r \in \pi_2(t)$, then $B \in \pi_3(t)$.
- If $A \sqsubseteq \forall r.B \in \mathcal{T}$, $A \in \pi_3(t)$ and $r^- \in \pi_2(t)$, then $B \in \pi_1(t)$.

Recall that we are only considering consistent knowledge bases. As discussed in the preliminaries, this means we may ignore all axioms containing ' $\perp$ ' in the universal model construction. On the same note, there is also no need to consider those axioms in the following rewriting procedure.

Definition 4.32. Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of positive, normalised constraints, let $N_C^{\mathcal{T}} \subseteq N_C$, $N_I^{\mathcal{T}} \subseteq N_I$ and $N_R^{\mathcal{T}} \subseteq N_R$ be the (finite) sets of concept, individual and role names occurring in $\mathcal{T}$ or $\mathcal{C}$. We let $\text{psat}_{\mathcal{C}, \mathcal{T}}$ be the smallest set of quadruples (t, P, Q, H) that are closed under the following rules and such that t is a locally consistent 2-type, such that $\pi_1(t) \cup \pi_3(t) \subseteq N_C^{\mathcal{T}}$, $\pi_2(t) \subseteq N_R^{\mathcal{T}} \cup (N_R^{\mathcal{T}})^-$ and P and Q are sets of basic shape and basic concept expressions, limited to the concept, individual and role names occurring in $N_C^{\mathcal{T}}$, $N_I^{\mathcal{T}}$ and $N_R^{\mathcal{T}} \cup (N_R^{\mathcal{T}})^-$, and H is a set of shape names.

1. If t is a locally consistent 2-type and Q is a set of basic shape expressions such that for all $\exists(\prod R).(\prod N) \in Q$, we have $(R, N) \in \text{succ}_{\mathcal{T}}(t)$ and not both $R \subseteq \pi_2(t)$ and $N \subseteq \pi_3(t)$, then $(t, \bar{Q}, Q, \emptyset)$ belongs to $\text{psat}_{\mathcal{C}, \mathcal{T}}$, where

$$\bar{Q} := \{\top\} \cup \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset)), \exists(\prod R).(\prod N) \notin Q\}.$$

2. If $\{(t, P, Q, H), (t, P', Q', H')\} \subseteq \text{psat}_{\mathcal{C}, \mathcal{T}}$ such that $\exists(\prod R).(\prod N) \in Q$ holds iff $\exists(\prod R).(\prod N) \in Q'$, then $(t, P \cup P', Q \cup Q', H \cup H')$ belongs to $\text{psat}_{\mathcal{C}, \mathcal{T}}$.
3. If $s \leftarrow S \in \mathcal{C}$ for some basic shape expression S and $(t, P, Q, H) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$, then $(t, P \cup (\{S\} \setminus N_C), Q, H \cup \{s\})$ belongs to $\text{psat}_{\mathcal{C}, \mathcal{T}}$ if either

- $S = a$, for $a \in N_I$ and $a \notin Q$; or
 - $S = A$, for $A \in \pi_1(t)$; or
 - $S = \exists r.s'$ and $r \in \pi_2(t)$ or there exists $\exists(\prod R).(\prod N) \in P$ such that $r \in R$ and $\exists r.s' \notin Q$.
4. If $s \leftarrow s' \in \mathcal{C}$, $(t, P, Q, H) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$ and $\{s'\} \subseteq H$, then $(t, P, Q, H \cup \{s\})$ belongs to $\text{psat}_{\mathcal{C}, \mathcal{T}}$.
5. If $s \leftarrow s_1 \wedge s_2 \in \mathcal{C}$, $(t, P, Q, H) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$ and $\{s_1, s_2\} \subseteq H$, then $(t, P, Q, H \cup \{s\})$ belongs to $\text{psat}_{\mathcal{C}, \mathcal{T}}$.
6. If $s \leftarrow \exists(\prod R).s' \in \mathcal{C}$, $\{(t, P, Q, H), (t', P', Q', H')\} \subseteq \text{psat}_{\mathcal{C}, \mathcal{T}}$, $R^- \subseteq \pi_2(t')$, $P' \cap N_I = \emptyset$, $s' \in H'$ and $\text{inv}(t') \in \text{child}_{\mathcal{T}}(\text{inv}(t))$ such that
- $\exists(\prod R').(\prod N') \in P'$ iff $(R', N') \in \text{succ}_{\mathcal{T}}((\pi_1(t'), \emptyset, \emptyset)) \setminus \text{succ}_{\mathcal{T}}(t')$; and
 - $\{s'' \mid \exists r''.s'' \in P'\} \subseteq H$,
- then $(t, P, Q, H \cup \{s\})$ belongs to $\text{psat}_{\mathcal{C}, \mathcal{T}}$.

In this definition, Rule 1 ensures that for each possible node in a data graph, there is a quadruple describing the I -type of that node in $\pi_1(t)$, restricted to the concept names occurring in $\mathcal{T}$. Furthermore, the set of neighbours this I -type needs to have in all models is partitioned into a set of neighbours we observe (P), and which will be introduced as children of this I -type (Q). Because of this, we may consider the first three entries of the quadruple as a description of a small part of a model, containing all information about its type and neighbours that might be relevant. In this way, it contains enough information to mimic the least fixed point operator for shape names step by step by Rules 3–6. Note that because we are considering the 2-type from the ‘perspective’ of the node, that is, ‘looking back towards the data’, we have to consider their inverses when checking the child-relationship in Rule 6. Lastly, Rule 2 is not necessary, but eases the completeness proof. We note that the same set of rules will be reused for the rewriting of stratified SHACL. Because of this, certain conditions do not have a purpose yet, like $c \notin Q$ in Rule 3, which will always be met in the current positive setting.

Moreover, note that even though the sets of basic shape and concept expressions are infinite, this does not imply that the set $\text{psat}_{\mathcal{C}, \mathcal{T}}$ contains an infinite amount of quadruples: in Rule 1 the type of quadruples with empty heads H is limited by the set of pairs occurring in the good successor configuration of a finite set of types. The other rules only add finite amount of quadruples and

Definition 4.33 (Rewriting Procedure). *Consider a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, $\mathcal{C}$ a set of positive, normalised constraints and K a set of quadruples (t, P, Q, H) . Let $\mathcal{C}_{\mathcal{T}, K}$ be the set of constraints that contains $\mathcal{C}$, and moreover, for each $(t, P, Q, H) \in K$ and each $s \in H$, the constraint*

$$s \leftarrow \bigwedge_{A \in \pi_1(t)} A \wedge \bigwedge_{A \in N_{\mathcal{T}} \setminus \pi_1(t)} \neg A \wedge \bigwedge_{X \in P} X \wedge \bigwedge_{Y \in Q} \neg Y. \quad (4.1)$$

Note that only $\pi_1(t)$ and not the full 2-type is taken into account when transforming a triple into a constraint. This suffices as we are considering axioms in normal form. Thus, knowing exactly which concept names are true in a node implies knowing exactly which structures may appear in the anonymous tree structure with this node as its root. The next step is eliminating the superfluous substructures of this anonymous tree, similar to what can be expected in any core universal model construction. This boils down to knowing exactly which of the substructures of depth at most 1 are already present in the enriched ABox $\mathcal{A}_{\mathcal{T}}$. Dividing all relevant substructures in the positive or negative set P and Q happens in Rule 1. As P is subsuming the information stemming from $\pi_2(t)$ and $\pi_3(t)$, it does not have to be repeated in (4.1).

Example 4.34. Suppose $\mathcal{T} = \{A \sqsubseteq \exists p.B, B \sqsubseteq \exists q.C\}$ and consider the following constraints C :

$$\begin{array}{lll} s \leftarrow \exists p.s & s \leftarrow \exists q.s & s \leftarrow s' \wedge s'' \\ s' \leftarrow \exists p^-.s' & s' \leftarrow \exists q^-.s' & s' \leftarrow A \\ s'' \leftarrow C. & & \end{array}$$

Let $t = (\{A\}, \{p\}, \{B\})$ and $t' = (\{B\}, \{q\}, \{C\})$ and recall the definition of the inverse of a 2-type: $\text{inv}(t) = (\{B\}, \{p^-\}, \{A\})$. Note that both $\text{inv}(t)$, $\text{inv}(t')$ and $t_A := (\{A\}, \emptyset, \emptyset)$ are locally consistent 2-types. Thus, according to the Rule 1 of Definition 4.32, the following quadruples belong to $\text{psat}_{C, \mathcal{T}}$:

$$(t_A, \{\top\}, \{\exists p.B\}, \emptyset), \quad (\text{inv}(t), \{\top\}, \{\exists q.C\}, \emptyset), \quad (\text{inv}(t'), \{\top\}, \emptyset, \emptyset).$$

Given these quadruples and following Rule 3 (of Definition 4.32) multiple times, we can also derive the following quadruples:

$$(t_A, \{\top\}, \{\exists p.B\}, \{s'\}), \quad (\text{inv}(t), \{\top, \exists p^-.s'\}, \{\exists q.C\}, \{s'\}), \\ (\text{inv}(t'), \{\top, \exists q^-.s'\}, \emptyset, \{s', s''\}).$$

That is, in the first quadruple, s' is added as a conclusion as A is contained in $\pi_1(t_A)$. For the second quadruple, we add $\exists p^-.s'$ to the set of positive assumptions, thus also describing an environment in which s' can be concluded. To see that the third quadruple is contained in $\text{psat}_{C, \mathcal{T}}$, Rule 3 is applied twice: once because C is contained in $\pi_1(\text{inv}(t'))$, and once because of the addition $\exists q^-.s'$ to the positive assumptions. The last quadruple may be updated to $(\text{inv}(t'), \{\top, \exists q^-.s'\}, \emptyset, \{s, s', s''\})$ because of $s \leftarrow s' \wedge s''$. Now Rule 6 can be used based on $s \leftarrow \exists q.s$, $(\text{inv}(t), \{\top, \exists p^-.s'\}, \{\exists q.C\}, \{s'\})$ and $(\text{inv}(t'), \{\top, \exists q^-.s'\}, \emptyset, \{s, s', s''\})$ to infer that $(\text{inv}(t), \{\top, \exists p^-.s'\}, \{\exists q.C\}, \{s, s'\})$ must belong to $\text{psat}_{C, \mathcal{T}}$ too. That is, we compare the environments described by these two quadruples: the main idea is that we want to derive s in the first quadruple because of a q connecting the first environment to the second one (reflected by the check $\{q^-\} \subseteq \pi_2(\text{inv}(t'))$), where s will be derived ($s \in H'$). The existence of this second environment must be supported by its type being a child of the first type ($\text{inv}(\text{inv}(t')) \in \text{child}_{\mathcal{T}}(\text{inv}(\text{inv}(t)))$) and thus cannot have an individual name in the set of positive assumptions ($P' \cap N_I = \emptyset$). Lastly, every assumption of the form $\exists r.s$ assumes, in case of an anonymous node, that its parent must be an s , and in case of a data node, that there exists an r -neighbour

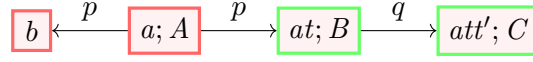


Figure 4.4: Austere Universal Model $\text{can}(\mathcal{T}, \mathcal{A})$, for $\mathcal{A} = \{A(a), p(a, b)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists p.B, B \sqsubseteq \exists q.C\}$

that is an s . As the second environment is assumed to describe an anonymous node, with as ‘parent environment’ the first quadruple, we have to check whether $s' \in H$. Similarly, positive assumptions of the form $\exists(\prod R').(\prod N')$ must be evaluated ‘upwards towards the parent environment’ as well, that is $R' \subseteq \pi_2(t')$ and $N' \subseteq \pi_3(t')$. Note that this exactly corresponds to $(R', N') \in \text{succ}_{\mathcal{T}}((\pi_1(t'), \emptyset, \emptyset)) \setminus \text{succ}_{\mathcal{T}}(t')$ for all $\exists(\prod R').(\prod N')$ that may appear in P' . Using this derived quadruple and again Rule 6, but now together with the quadruple $(t_A, \{\top\}, \{\exists p.B\}, \{s'\})$ and constraint $s \leftarrow \exists p.s$, we also find $(t_A, \{\top\}, \{\exists p.B\}, \{s, s'\}) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$. Thus, the following constraint is contained in $\mathcal{C}_{\mathcal{T}, \emptyset}$:

$$s \leftarrow A \wedge \neg B \wedge \neg C \wedge \top \wedge \neg \exists p.B.$$

Now let $\mathcal{A} = \{A(a), p(a, b)\}$. The target we wish to validate is $\mathcal{G} = \{s(a)\}$. It is now simple to check that this target, $s(a)$, is validated by considering the above constraint on $\mathcal{A}_{\mathcal{T}}$, which coincides with $\mathcal{A}$ in this example.

The other approach to test validation is to build $\text{can}(\mathcal{T}, \mathcal{A})$, graphically depicted in Figure 4.4, and then check validation of $(\mathcal{C}, \mathcal{G})$. In this figure, the same notation is used as described in Example 4.2. It is straightforward to check that $\text{can}(\mathcal{T}, \mathcal{A})$ indeed validates $(\mathcal{C}, \mathcal{G})$, as required. $\heartsuit$

We note that, even in the best-case scenario, Definition 4.32 creates an amount of quadruples exponential in size of $\mathcal{T}$ and $\mathcal{C}$. This expressive power is used to describe all possible relevant settings that may appear at the border between the data and anonymous part of the austere universal model. However, what is considered relevant, is only dependent on the TBox $\mathcal{T}$; concept names that do not appear in any axiom cannot be introduced or introduce anything new and may thus be ignored. This means that the exponential blow-up observed in this definition does not influence the data complexity of determining SHACL validation in presence of ontologies. Moreover, as the described SHACL constraint rewriting happens completely independent of the data, the same rewritten shape constraint sets may be reused for different data graphs.

4.4.2 Completeness and Correctness

In the rest of this section, we will show completeness and correctness of the presented rewriting technique to translate (positive) SHACL validation in presence of ontologies to plain SHACL validation over the enriched ABox $\mathcal{A}_{\mathcal{T}}$. After that we aim further: in the next subsection, we discuss how to handle stratified negation in the rewriting, followed by a section on how to achieve a pure rewriting, that is, to plain SHACL validation over the original ABox $\mathcal{A}$.

For a quadruple $\rho = (t, P, Q, H)$ a shape name $s \in H$ and a fresh shape name s_ρ , we define the constraint s_ρ as follows:

$$s_\rho \leftarrow \bigwedge_{A \in \pi_1(t)} A \wedge \bigwedge_{A \in N_{\mathcal{C}}^T \setminus \pi_1(t)} \neg A \wedge \bigwedge_{X \in P} X \wedge \bigwedge_{Y \in Q} \neg Y.$$

Note that this definition is very close to Definition 4.33. However, it has a different purpose. In the following proofs, we will focus on validating s_ρ instead of s . This makes the connection to a certain quadruple precise. And clearly, if $s_\rho(a)$ is contained in some perfect assignment, also $s(a)$ is contained in that same assignment. We will use the notation $\mathcal{C} \cup s_\rho$ to denote that the constraint defining s_ρ as above is added to the constraints in $\mathcal{C}$.

Furthermore, we note that quadruples are designed to model the direct environment of a node that is on the ‘border’ of the model under construction. More precisely, they model the environment of e in $\text{can}_{|e|}(\mathcal{T}, \mathcal{A})$ (sometimes shortened to $\mathcal{I}_{|e|}$), the approximation of the austere universal model, as defined in Definition 4.14. In this interpretation, e is missing its successors. For this reason, we will evaluate $s_\rho(e)$ specifically on $\text{can}_{|e|}(\mathcal{T}, \mathcal{A})$. When proving this in general, it definitely also holds for all e such that $|e| = 1$, or put differently, all $e \in N_I$. We will prove this via an induction on which new shape atoms are derived in the perfect assignment for all sizes of e at the same time. The latter is necessary because of constraints of the form $s \leftarrow \exists r.s'$, which may change the distance of the considered nodes to the ABox. For the other direction, we will step by step decrease $|e|$ until we reach $|e| = 1$.

To clarify which shape atoms depend on others, we consider the immediate consequence operator $T_{\mathcal{I}, \mathcal{C}}$. Here ‘depending’ is intended in the sense that $s(e)$ depends on $s'(e)$ and $s''(e)$ in case of the constraint $s \leftarrow s' \wedge s''$, and similar for other constraints mentioning s' or s'' in its body. Recall that the perfect assignment for positive constraints consists simply of the least fixed point of this operator. With all this machinery, we can demonstrate how the rules in Definition 4.32 can mimic all these types of dependencies. In this way, we show first the completeness, and then, in Proposition 4.37, the correctness of the rewriting for positive constraints.

Proposition 4.35. *Given a normalised Horn-ALCHIQ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of positive, normalised constraints, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that if $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$, then $\text{can}_1(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}_{\mathcal{T}}, \mathcal{G})$.*

Proof. In the following claim we will show that for each approximation $\text{can}_i(\mathcal{T}, \mathcal{A})$, there exist a quadruple for each node of size i such that all shape names that are observed in the perfect assignment of the original set of constraints over $\text{can}(\mathcal{T}, \mathcal{A})$ are found in the head of the quadruple. And that in such a way that t, P and Q describe the environment of that node. In particular, this will also hold for $i = 1$.

The proof of the following claim consists of an induction on n , the amount of times the direct consequence operator of the perfect assignment has been applied.

Claim 4.36. *Let $\mathcal{I}_i = \text{can}_i(\mathcal{T}, \mathcal{A})$ and $S_n = T_{\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}} \uparrow^n (\emptyset)$. If $s(e) \in S_{n+1}$, there exists a quadruple $\rho = (t, P, Q, H) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$ such that $s \in H$, $s_\rho(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}} \cup s_\rho}(S_n)$ and furthermore:*

1. if $e \in N_K \setminus N_I$, then

- a) $t = \text{inv}(\text{tail}(e))$;
- b) $P \supseteq \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset)) \setminus \text{succ}_{\mathcal{T}}(t))\}$ and $P \cap N_I = \emptyset$;
- c) $Q \supseteq \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset))\}$;

2. for all other $e \in N_I$, we have

- a) $t = (\{A \in N_C \mid A(e) \in \mathcal{A}_{\mathcal{T}}\}, \emptyset, \emptyset)$;
- b) $P \supseteq \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}(t), e \in (\exists(\prod R).(\prod N))^{\mathcal{A}_{\mathcal{T}}}\}$ and $(P \cap N_I) \setminus \{e\} = \emptyset$;
- c) $Q \supseteq \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}(t), e \notin (\exists(\prod R).(\prod N))^{\mathcal{A}_{\mathcal{T}}}\}$;

For $(n = 0)$, note that either $s \leftarrow e \in \mathcal{C}$ or $s \leftarrow A \in \mathcal{C}$. In the first case, $\rho = (t, P \cup \{e\}, Q, \{s\})$ can be constructed by performing Rule 1, followed by Rule 3, where $t = (\{A \in N_C \mid A(e) \in \mathcal{A}_{\mathcal{T}}\}, \emptyset, \emptyset)$, and P and Q are the smallest sets that satisfy the criteria. Removing $\{e\}$ from the second entry in the quadruple, and possibly using $t = \text{inv}(\text{tail}(e))$ instead of $t = (\{A \in N_C \mid A(e) \in \mathcal{A}_{\mathcal{T}}\}, \emptyset, \emptyset)$ in case $e \notin N_I$, suffices for the second case. It is then easy to check that indeed $s_{\rho}(e) \in T_{\mathcal{I}_0, \mathcal{C}_{\mathcal{T}} \cup s_{\rho}}(\emptyset)$.

Now suppose $s(e) \in S_{n+1} \setminus S_n$. In this induction step, we will only focus on the case $s(e)$ got derived from $s \leftarrow \exists r.s' \in \mathcal{C}$ and $s'(d) \in S_n$, for some r -neighbour d of e , as the other derivation options are reflected in a straightforward way in the rules of Definition 4.32. We distinguish two cases: $|d| \leq |e|$, which we call ‘upwards’, and $|d| > |e|$, which we call ‘downwards’.

For the upwards case, if $|d| \leq |e|$ then either there exists a 2-type t such that $dt = e$ and $r^- \in \pi_2(t)$, or $\{e, d\} \subseteq N_I$.

- $dt = e$. By Rule 1, we find $(\text{inv}(t), P, Q, \emptyset) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$, such that 1a, 1b and 1c of the claim are satisfied. With Rule 3, recalling that we assumed $s \leftarrow \exists r.s' \in \mathcal{C}$, we can update this quadruple to find $\rho = (\text{inv}(t), P \cup \{\exists r.s'\}, Q, \{s\}) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$. As $s'(d) \in S_n$, it is straightforward to see that $s_{\rho}(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}} \cup s_{\rho}}(S_n)$, which concludes this case.
- $\{e, d\} \subseteq N_I$. Again, by Rule 1, we find $(t, P, Q, \emptyset) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$, where $t = (\{A \in N_C \mid A(e) \in \mathcal{I}_1\}, \emptyset, \emptyset)$ and such that 1a, 1b and 1c of the claim are satisfied. Applying Rule 3, means that also $\rho = (t, P \cup \{\exists r.s'\}, Q, \{s\}) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$. With similar reasoning as in the previous case, it is clear that indeed $s_{\rho}(e) \in T_{\mathcal{I}_1, \mathcal{C}_{\mathcal{T}} \cup s_{\rho}}(S_n)$.

In the second case, downwards, we have $|d| > |e|$, that is, $et' = d$, for some 2-type t' . In the following argument, we assume $e \in N_K \setminus N_I$, but a similar argument works for $e \in N_I$. Because we assumed that $s'(d) \in S_n$, we can apply the induction hypothesis to find $\rho' = (\text{inv}(t'), P', Q', H') \in \text{psat}_{\mathcal{C}, \mathcal{T}}$ such that (i) $r^- \in \pi_2(\text{inv}(t'))$, (ii) $P' \cap N_I = \emptyset$, (iii) $s' \in H'$, (iv) $\exists(\prod R).(\prod N) \in P'$ iff $(R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t'), \emptyset, \emptyset)) \setminus \text{succ}_{\mathcal{T}}(t')$. Furthermore, by construction of elements in N_K , we find $t' \in \text{child}_{\mathcal{T}}(\text{tail}(e))$. This means that to perform

Rule 6, we only need a second quadruple (t, P, Q, H) for which we have to ensure that $\{s'' \mid \exists r''.s'' \in P'\} \in H$.

So, suppose $\exists r''.s'' \in P'$. Since, by induction hypothesis, $s_{\rho'}(d) \in T_{\mathcal{I}_{|d|}, \mathcal{C}_{\mathcal{T}} \cup s_{\rho'}}(S_n)$, we know $s''(e) \in S_n$. Thus, we can also apply the induction hypothesis to $s''(e)$, to find that $\rho_{s''} = (\text{inv}(\text{tail}(e)), P_{s''}, Q_{s''}, H_{s''}) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$ such that $P_{s''}$ and $Q_{s''}$ satisfy the constraints in 1b and 1c of the claim we are currently proving, and $s'' \in H_{s''}$. Furthermore, we find that $s_{\rho_{s''}}(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}}}(S_n)$.

For each $\exists r''.s'' \in P'$, we can construct such a quadruple $\rho_{s''}$ and combine them with Rule 2 to conclude

$$\rho'' = (\text{inv}(\text{tail}(e)), \bigcup_{\exists r''.s'' \in P'} P_{s''}, \bigcup_{\exists r''.s'' \in P'} Q_{s''}, \bigcup_{\exists r''.s'' \in P'} H_{s''}) \in \text{psat}_{\mathcal{C}, \mathcal{T}},$$

which can serve exactly as the quadruple we were looking for to apply Rule 6, which we also do. Thus, we find that

$$\rho = (\text{inv}(\text{tail}(c)), \bigcup_{\exists r''.s'' \in P'} P_{s''}, \bigcup_{\exists r''.s'' \in P'} Q_{s''}, \bigcup_{\exists r''.s'' \in P'} H_{s''} \cup \{s\}) \in \text{psat}_{\mathcal{C}, \mathcal{T}},$$

too. Note that this quadruple satisfies all properties requested in the claim, which naturally follows from the point that all $\rho_{s''}$ satisfy these properties. Furthermore, note that $s_{\rho''}(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}} \cup s_{\rho''}}(S_n)$ holds, because for each $\exists r''.s'' \in P'$, we have $s_{\rho_{s''}}(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}} \cup s_{\rho_{s''}}}(S_n)$. Thus, we can conclude that $s_{\rho}(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}} \cup s_{\rho}}(S_n)$, as required.

In case $\{s'' \mid \exists r''.s'' \in P'\} = \emptyset$, the above does not work. Instead, we can simply apply Rule 1 to construct the required $\rho = (t, P, Q, H)$, such that $P = \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset)) \setminus \text{succ}_{\mathcal{T}}(t))\}$ and $Q = \{\exists(\prod R).(\prod N) \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset))\}$. Then a simple argument suffices to show that $s_{\rho}(e) \in T_{\mathcal{I}_{|e|}, \mathcal{C}_{\mathcal{T}} \cup s_{\rho}}(S_n)$, which concludes the induction step. $\clubsuit$

Proposition 4.37. *Given a normalised Horn-ALCHIQ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of positive, normalised constraints, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that if $\text{can}_1(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}_{\mathcal{T}}, \mathcal{G})$, then $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$.*

Proof. We consider the following claim. Taking $i = 1$ suffices to conclude the above proposition.

Claim 4.38. *Let $\mathcal{I}_i = \text{can}_i(\mathcal{T}, \mathcal{A})$. If $\rho = (t, P, Q, H) \in \text{psat}_{\mathcal{C}, \mathcal{T}}$ with $s \in H$ and $s_{\rho}(e) \in \text{PA}(\mathcal{C}_{\mathcal{T}} \cup \{s_{\rho}\}, \mathcal{I}_i)$ for some $e \in N_{\mathcal{K}}$ such that $|e| = i$, then $s(c) \in \text{PA}(\mathcal{C}, \text{can}(\mathcal{T}, \mathcal{A}))$.*

We show this by letting i decrease step-by-step until reaching 1. Note that because we are considering a least fixed point computation and $\mathcal{C} \subseteq \mathcal{C}_{\mathcal{T}}$, there must exist an $i = n$ such that the above naturally holds.

Assume $i = n - 1$. Suppose that $s_{\rho}(e) \in \text{PA}(\mathcal{C}_{\mathcal{T}} \cup \{s_{\rho}\}, \mathcal{I}_{n-1})$, such that $|e| = n - 1$. The proof is based on unwinding how ρ got constructed, by an induction on the construction. Without loss of generality, we can assume s got added to H in the last rule applied. Now suppose the

last rule applied was Rule 6 (the other cases are rather straightforward). Thus, there exists some $s \leftarrow \exists r.s' \in \mathcal{C}$ and moreover we find that $\{(t, P, Q, H \setminus \{s\}), (t', P', Q', H')\} \in \text{psat}_{\mathcal{C}, \mathcal{T}}$, such that $s' \in H'$. Let $\rho' = (t', P', Q', H')$ and note that because we were able to perform Rule 6, we can conclude that $s'_{\rho'}(et') \in PA(\mathcal{C}_{\mathcal{T}} \cup \{s'_{\rho'}\}, \mathcal{I}_n)$. By construction of the austere core model, it is clear that in $\mathcal{I}_n$ the node et' satisfies exactly all concept names mentioned in $\pi_1(t')$; if $\exists(\prod R). \prod N \in P'$, it follows that $R \subseteq \pi_2(t')$ and $N \subseteq \pi_3(t')$; and if $\exists r'.s'' \in P$, then s'' must be contained in H , which means we can first perform this same proof on s'' and the quadruple $(t, P, Q, H \setminus \{s\})$ to conclude that $s''(e) \in PA(\mathcal{C}, \text{can}(\mathcal{T}, \mathcal{A}))$, and thus $s''(e) \in PA(\mathcal{C}_{\mathcal{T}}, \mathcal{I}_n)$ by Proposition 4.35. This means that indeed $et' \in (\exists r'.s'')^{\mathcal{I}_n, \emptyset}$, as required. As et' will not have any successor nodes in $\mathcal{I}_n$, the conditions posed by Q' are enforced too.

Combining $s'_{\rho'}(et') \in PA(\mathcal{C}_{\mathcal{T}} \cup \{s'_{\rho'}\}, \mathcal{I}_n)$ with assuming the claim holds for $i = n$, we derive that $s'(et') \in PA(\mathcal{C}, \text{can}(\mathcal{T}, \mathcal{A}))$. Since $s \leftarrow \exists r.s' \in \mathcal{C}$, and $r^- \in \pi_2(t')$ we find $s(e) \in PA(\mathcal{C}, \text{can}(\mathcal{T}, \mathcal{A}))$ too, as required. ♣

These results can be summarised in the following way.

Theorem 4.39. *Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of positive, normalised constraints, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}_{\mathcal{T}}$ validates $(\mathcal{C}_{\mathcal{T}}, \mathcal{G})$, under the least fixed point, well-founded and stable model semantics*

4.5 Rewriting for Stratified SHACL

We now extend the rewriting to constraint sets $\mathcal{C}$ with stratified negation. Intuitively, this is done by running a saturation procedure in the rewriting for each stratum of $\mathcal{C}$, starting with the lowermost. For the transition to the next stratum in the rewriting procedure, we need to ensure that the outcome of the saturation at a non-topmost stratum is completed with negative information. To this end, we w.l.o.g. assume that all constraints from $\mathcal{C}$ with the same shape name on the left-hand-side occur together in the same stratum.

We will operate again on quadruples (t, P, Q, H) , which are similar to the ones in the previous section, except that H might additionally contain expressions of the form $\neg s$ for a shape name s . For a set K of such quadruples, we say (t, P, Q, H) is *maximal* in K if $(t, P, Q, H) \in K$ and there is no $H' \supset H$ with $(t, P, Q, H') \in K$. Then the notion of *completion* is defined as follows:

Definition 4.40. *The completion $\text{comp}_{\mathcal{C}, \mathcal{T}}(K)$ of a set K of quadruples w.r.t. a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and a set of normalised and stratified constraints $\mathcal{C}$ is a set, defined as follows:*

$$\text{comp}_{\mathcal{C}, \mathcal{T}}(K) = \{(t, P, Q \cup Q', H \cup \bar{H}) \mid (t, P, Q, H) \text{ is maximal in } K\},$$

where

$$\begin{aligned} \bar{H} &:= \{\neg s \mid s \text{ occurs in } \mathcal{C}, s \notin H\} \\ Q' &:= \{\exists r.s' \mid s \leftarrow \exists r.s' \in \mathcal{C}, s \notin H\} \cup \{a \mid s \leftarrow a \in \mathcal{C}, s \notin H\}. \end{aligned}$$

We need to augment the inference rules of the rewriting procedure for positive constraints with an additional rule to handle constraints of the form $s \leftarrow \neg s'$. Furthermore, Rule 6, is updated to also consider the newly added information to Q . Note that using the following updated definition of $\text{sat}_{\mathcal{C}, \mathcal{T}}(K)$ will give the same results as the previously defined $\text{psat}_{\mathcal{C}, \mathcal{T}}(K)$ in Definition 4.32 on the first stratum, as the proposed changes do not have any effect if no $\neg \exists r.s$'s or $\neg s$'s are present in the quadruples.

Definition 4.41. *Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of normalised and stratified constraints, we let $\text{sat}_{\mathcal{C}, \mathcal{T}}(K)$ be the smallest set of pairs containing K closed under Rules 2–5 defined of Definition 4.32, where ‘ $\text{psat}_{\mathcal{C}, \mathcal{T}}$ ’ is replaced by ‘ $\text{sat}_{\mathcal{C}, \mathcal{T}}(K)$ ’, and additionally under the following Rule 6’ and 7:*

- 6'. If $s \leftarrow \exists r.s' \in \mathcal{C}$, $\{(t, P, Q, H), (t', P', Q', H')\} \subseteq \text{sat}_{\mathcal{C}, \mathcal{T}}(K)$, $r^- \in \pi_2(t')$, $P' \cap N_I = \emptyset$, $s' \in H'$ and $\text{inv}(t') \in \text{child}_{\mathcal{T}}(\text{inv}(t))$ such that
- $\exists(\prod R). \prod N \in P'$ iff $(R, N) \in \text{succ}_{\mathcal{T}}(\pi_1(t'), \emptyset, \emptyset) \setminus \text{succ}_{\mathcal{T}}(t')$;
 - $\{s'' \mid \exists r''. s'' \in P'\} \cup \{\neg s'' \mid \exists r''. s'' \in Q', r'' \in \pi_2(t')\} \subseteq H$,
- then $(t, P, Q, H \cup \{s\})$ belongs to $\text{sat}_{\mathcal{C}, \mathcal{T}}(K)$.
7. If $s \leftarrow \neg s' \in \mathcal{C}$ and $(t, P, Q, H) \in \text{sat}_{\mathcal{C}, \mathcal{T}}(K)$ such that $\neg s' \in H$, then $(t, P, Q, H \cup \{s\})$ belongs to $\text{sat}_{\mathcal{C}, \mathcal{T}}(K)$.

Now can we define the inference procedure that processes strata from the lowest to the top-most, performing saturation using the updated set of rules at every stratum, interleaved with a computation of the completion in between.

Definition 4.42. *For a TBox $\mathcal{T}$ and a constraint set $\mathcal{C}$ with stratification $\mathcal{C}_0, \dots, \mathcal{C}_n$, we let $K_0 = \text{psat}_{\mathcal{C}_0, \mathcal{T}}$ and for each $0 < i \leq n$*

$$K_i = \text{sat}_{\mathcal{C}_i, \mathcal{T}}(\text{comp}_{\mathcal{C}_{i-1}, \mathcal{T}}(K_{i-1})).$$

We let $\mathcal{C}_{\mathcal{T}} = \mathcal{C}_{\mathcal{T}, K_n}$, where $\mathcal{C}_{\mathcal{T}, K_n}$ is defined as in Definition 4.33.

Example 4.43. *Suppose $\mathcal{T} = \{A \sqsubseteq \exists p.B\}$ and consider the following set of constraints $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1$:*

$$\begin{aligned} \mathcal{C}_0 &= \{s_C \leftarrow C, & s' \leftarrow \exists p.s_C\} \\ \mathcal{C}_1 &= \{s'' \leftarrow \exists p.\bar{s}_C, & \bar{s}_C \leftarrow \neg s_C, & s \leftarrow s' \wedge s''\}. \end{aligned}$$

We consider the following two 2-types in the rest of this example: $t_A := (\{A\}, \emptyset, \emptyset)$ and $t := (\{A\}, \{p\}, \{B\})$. Because of Rule 1, we find that the following two quadruples are contained in K_0 :

$$(t_A, \{\top\}, \{\exists p.B\}, \emptyset), \quad (\text{inv}(t), \{\top\}, \emptyset, \emptyset).$$

Here, the first one can be updated to $(t_A, \{\top, \exists p.s_C\}, \{\exists p.B\}, \{s'\})$. Note that all three quadruples are maximal in K_0 , which means we find the following quadruples in $\text{comp}_{C_0, \mathcal{T}}$:

$$\begin{aligned} & (t_A, \{\top\}, \{\exists p.B, \exists p.s_C\}, \{\neg s' \neg s_C\}), \\ & (t_A, \{\top, \exists p.s_C\}, \{\exists p.B\}, \{s', \neg s_C\}), \\ & (\text{inv}(t), \{\top\}, \{\exists p.s_C\}, \{\neg s', \neg s_C\}). \end{aligned}$$

By definition, these quadruples are also contained in K_1 , and thus, because of Rule 7, also the versions in which $\bar{s}_C$ is added to the fourth set in each quadruple. Then, we can apply Rule 6' to $(t_A, \{\top, \exists p.s_C\}, \{\exists p.B\}, \{s', \neg s_C, \bar{s}_C\})$, $(\text{inv}(t), \{\top\}, \{\exists p.s_C\}, \{\neg s', \neg s_C, \bar{s}_C\})$ and the constraint $s'' \leftarrow \exists p.\neg s_C$, to find

$$(t_A, \{\top, \exists p.s_C\}, \{\exists p.B\}, \{s', \neg s_C, \bar{s}_C, s''\}) \in K_1$$

too. After applying Rule 5 based on the constraint $s \leftarrow s' \wedge s'' \in \mathcal{C}_1$, we can extract the following rewritten constraint:

$$s \leftarrow A \wedge \neg B \wedge \neg C \wedge \exists p.s_C \wedge \neg \exists p.B \in \mathcal{C}_{\mathcal{T}},$$

Now let $\mathcal{A} = \{A(a), p(a, b), C(b)\}$ and set the target to $\mathcal{G} = \{s(a)\}$. As $\mathcal{A} = \mathcal{A}_{\mathcal{T}}$, we indeed find a positive answer when validating $(\mathcal{C}_{\mathcal{T}}, \mathcal{G})$ over $\mathcal{A}_{\mathcal{T}}$. Notice $\text{can}(\mathcal{T}, \mathcal{A})$ corresponds to the canonical interpretation of $\{A(a), p(a, b), C(b), p(a, at), B(at)\}$. Thus, we can conclude that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ too. $\heartsuit$

Given a stratified set of constraints $\mathcal{C}$ with stratification $\mathcal{C}_0, \dots, \mathcal{C}_n$, let $\mathcal{C}_{\leq i} := \mathcal{C}_0 \cup \dots \cup \mathcal{C}_i$. Similarly, we will use $\mathcal{C}_{\mathcal{T}, \leq i}$ to denote $\mathcal{C}_{\mathcal{T}, K_0} \cup \dots \cup \mathcal{C}_{\mathcal{T}, K_i}$.

Theorem 4.44. *Given a normalised Horn-ALCHIQ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of normalised constraints with stratification $\mathcal{C}_0, \dots, \mathcal{C}_n$, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}_{\mathcal{T}}$ validates $(\mathcal{C}_{\mathcal{T}}, \mathcal{G})$ under the least fixed point, well-founded and stable model semantics.*

Proof. ($\Rightarrow$). We will prove completeness by proving the following claim by induction on i

Claim 4.45. *Let $\mathcal{I}_i = \text{can}_i(\mathcal{T}, \mathcal{A})$. For each $e \in N_{\mathcal{K}}$ and for each $0 \leq i \leq n$, there exists $\rho = (t, P, Q, H) \in K_i$ that is maximal in K_i , and moreover:*

1. if $e \in N_{\mathcal{K}} \setminus N_I$, then
 - a) $t = \text{inv}(\text{tail}(e))$;
 - b) $P \supseteq \{\exists(\prod R). \prod N \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset)) \setminus \text{succ}_{\mathcal{T}}(t)\}$ and $P \cap N_I = \emptyset$;
 - c) $Q \supseteq \{\exists(\prod R). \prod N \mid (R, N) \in \text{succ}_{\mathcal{T}}((\pi_1(t), \emptyset, \emptyset))\}$;
2. for all other $e \in N_I$, we have

- a) $t = (\{A \in N_C \mid A(e) \in \mathcal{A}_T\}, \emptyset, \emptyset);$
- b) $P \supseteq \{\exists(\prod R). \prod N \mid (R, N) \in \text{succ}_T(t), e \in (\exists(\prod R). \prod N)^{\mathcal{A}_T}\} \text{ and } (P \cap N_I) \setminus \{e\} = \emptyset;$
- c) $Q \supseteq \{\exists(\prod R). \prod N \mid (R, N) \in \text{succ}_T(t), e \notin (\exists(\prod R). \prod N)^{\mathcal{A}_T}\};$
- 3. $H = \{s \mid s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})\} \cup \{\neg s \mid s(e) \notin PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i}), s \text{ defined in } \mathcal{C}_{i-1}\};$
- 4. and $s_\rho(e) \in PA(\mathcal{I}_{|e|}, \mathcal{C}_{T, \leq i} \cup s_\rho).$

Analogous to Proposition 4.35, it follows that for each s such that $s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq 0})$, there exists a quadruple $(t, P_s, Q_s, H_s) \in \text{sat}_{\mathcal{C}_0, \mathcal{T}}(\emptyset) = K_0$ such that $s \in H_s$ and such that requirements 1, 2 and 4 of the claim hold. Now let $X = \{s \mid s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq 0})\}$. Then, following Rule 2 of the rewriting algorithm in Definition 4.32, we must conclude that also $(t, \bigcup_{s \in X} P_s, \bigcup_{s \in X} Q_s, \bigcup_{s \in X} H_s) \in K_0$. This quadruple is exactly the quadruple we were looking for: requirements 1, 2 and 4 have to be satisfied, as they are satisfied for each (t, P_s, Q_s, H_s) separately; requirement 3 follows, as $s \in H_s$ implies $X \subseteq \bigcup_{s \in X} H_s$, and at the same time the correctness of the rewriting forbids that $H_s \setminus X \neq \emptyset$. In the same way, maximality of the quadruple is implied.

For the induction step, $i = j + 1$, note that for each $e \in N_I \cup N_K$, we are given some $(t, P, Q, H) \in K_j$ that is maximal and satisfies requirements 1 to 4 for $i = j$. As this quadruple is assumed to be maximal, completion $\text{comp}_{\mathcal{C}_j, \mathcal{T}}$ can be applied to (t, P, Q, H) . Thus, we find $(t, P, Q \cup Q', H \cup \bar{H}) \in K_{j+1}$. Since $s \in H$ iff $s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq j})$, it must follow that $\neg s \in \bar{H}$ iff $s(e) \notin PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq j})$ and s occurs in $\mathcal{C}_{\leq j}$. Now suppose $s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq j+1}) \setminus PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq j})$, then there must exist a constraint $s \leftarrow C \in \mathcal{C}$ that caused this addition.

If $C = \neg s'$, then s' is defined in $\mathcal{C}_{\leq j}$ by the stratification rules, and $s'(e) \notin PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq j})$, which means $\neg s' \in \bar{H}$, thus Rule 7 can be applied to conclude that $(t, P, Q \cup Q', H \cup \bar{H} \cup \{s\}) \in K_{j+1}$. If C is any of the other options, the same reasoning as in the proof of Proposition 4.35 can be used.

Now all that is left to show, is that each $Y \in Q'$ is not blocking the satisfaction of restriction 4. By taking a closer look at the completion procedure, we see that $Y = \exists r. s'$ for some $s \leftarrow \exists r. s' \in \mathcal{C}_{\leq j}$ such that $s \notin H$, or $Y = e$, for some $s \leftarrow e \in \mathcal{C}_{\leq j}$ and $s \notin H$. For both cases, note that $s \notin H$, which means $s(e) \notin PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq j})$. Assuming that any of these Y is indeed blocking restriction 4, directly leads to a contradiction with the previous observation. Thus, we have indeed found the required quadruple.

($\Leftarrow$). Again, let $\mathcal{I}_i = \text{can}_i(\mathcal{T}, \mathcal{A})$. To show correctness, we will show by induction on i that for each $e \in N_K$ and for each $\rho = (t, P, Q, H) \in K_i$ such that (a) $s_\rho(e) \in PA(\mathcal{I}_{|e|}, \mathcal{C}_{T, \leq i} \cup s_\rho)$, we find for each $s \in H$ that $s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$.

First, note that Proposition 4.37 provides the induction basis. So, for the induction step, suppose we are given some $e \in N_K$ and $(t, P, Q, H) \in K_{i+1}$ such that (a) holds. We will focus on the

case that there exists a $(t, P, Q', H') \in K_i$ that is maximal in K_i , such that (t, P, Q, H) is the result of first applying completion to (t, P, Q', H') to find $(t, P, Q' \cup Q'', H' \cup \bar{H}') \in \text{comp}_{\mathcal{C}_i, \mathcal{T}}$, followed by application(s) of rules 3-5, 6' or 7. Here, we will only consider the result of applying Rule 7, as the treatment of the application of the other rules is very similar to what is discussed in Proposition 4.37. Note that Rules 1 and 2 are not really having any relevant effects after finishing the first stratum.

So, assume $s \leftarrow \neg s' \in \mathcal{C}_{i+1}$ and $\neg s' \in \bar{H}'$. We argue that each $s' \leftarrow C \in \mathcal{C}$ cannot be used to deduce $s'(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$.

- $C \in N_C$. If $C \in \pi_1(t)$, we would have found $s' \in H'$ by maximality of (t, P, Q', H') . Thus, $C \notin \pi_1(t)$, which means $\neg C$ appears in $s_{(t, P, Q, H)}$. Since (a) holds, this means that $e \notin C^{\text{can}(\mathcal{T}, \mathcal{A})}$, which suffices.
- $C \in N_I$. As $\neg s' \in \bar{H}'$, we find $C \in Q$. Since (a) holds, this means $C^{\text{can}(\mathcal{T}, \mathcal{A})} \neq e^{\text{can}(\mathcal{T}, \mathcal{A})}$, which suffices.
- $C \in \{s'', s'' \wedge s''' \mid \{s'', s'''\} \subseteq N_S\}$. Perform this same argument for s'' or s''' . As we are considering a least fixed point semantics, this regression must terminate.
- $C = \exists r.s''$, for some $r \in N_R$, $s \in N_S$. As $\neg s' \in \bar{H}'$, we find $C \in Q$. Note we derived completeness and correctness of all strata up till i , that is for all s , for all natural numbers n , and all e such that $|e| \leq n$, we have $s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$ iff $s(e) \in PA(\mathcal{I}_n, \mathcal{C}_{\mathcal{T}, \leq i})$. Thus, combining this with (a), we find that in any case $s''(d) \notin PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$, for $dt = e$, such that $r^- \in \pi_2(t)$.

The other option we have to exclude is $d = et$, for some $r \in \pi_2(t)$. For contradiction, assume $s''(d) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$. So we know there must exist a quadruple $\rho_d = (inv(t), P_d, Q_d, H_d) \in K_i$ such that $s'' \in H_d$ and $s_{\rho_d}(d) \in PA(\mathcal{I}_{|d|}, \mathcal{C}_{\leq i, \mathcal{T}} \cup s_{\rho_d})$, and such that there does not exist $(inv(t), P'_d, Q'_d, H'_d) \in K_i$ with the same properties, but $|\{\exists r.s \in P'_d\}| < |\{\exists r.s \in P_d\}|$. First of all, note that $\{\neg s \mid \exists r.s \in Q_d\} \subseteq H$; for all such s , we have $\neg s \notin H$, implies $s \in H$. Thus, $s(e) \in PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$ by induction hypothesis. Using completeness and correctness, this implies $s_{\rho_d}(d) \notin PA(\text{can}_{|d|}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i, \mathcal{T}} \cup s_{\rho_d})$, which is a contradiction.

Now there are two options: if $\{s \mid \exists r.s \in P_d\} \subseteq H$, then Rule 6' could have been applied on $s' \leftarrow \exists r.s''$ at a certain point, which is in contradiction with the maximality of (t, P, Q', H') . So, we are left with the second option: $\{s \mid \exists r.s \in P_d\} \not\subseteq H$. Now we can repeat this same argument for each member of $\{s \mid \exists r.s \in P_d\} \setminus H$. Because we are solely considering the least amount of $\exists r.s$'s in P_d , we know they all are essential in deriving $s''(d)$. Thus, $s''(d) \notin PA(\text{can}(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$ for all d such that $(e, d) \in (r)^{\text{can}(\mathcal{T}, \mathcal{A})}$, which automatically implies the required result.

- $C = \neg s''$. If $s \leftarrow \neg s' \in \mathcal{C}_{i+1}$, then $s' \leftarrow \neg s'' \in \mathcal{C}_{\leq i}$, by definition of the strata. This means that $\neg s''$ cannot be part of $\bar{H}'$. So, to apply Rule 7, it must be that $\neg s'' \in H'$. However, this is in contradiction with the maximality of (t, P, Q', H') that would imply $s' \in H'$, which concludes this case.

Thus, we must conclude that $s'(e) \notin PA(can(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i})$, from which follows that indeed $s(e) \in PA(can(\mathcal{T}, \mathcal{A}), \mathcal{C}_{\leq i+1})$, as required.



4.6 Pure Rewriting

Theorem 4.44 lets us reduce SHACL validation in the presence of Horn- $\mathcal{ALCH}\mathcal{I}$ TBoxes to plain SHACL validation over the completed ABox $\mathcal{A}_{\mathcal{T}}$. In this section we discuss how to update this rewriting to a pure rewriting. That is, how to produce a set of constraints $\mathcal{C}'$, in a data-independent way, such that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}', \mathcal{G})$.

To make the rewriting pure, we have to encode the information added to $\mathcal{A}$ by the immediate consequence operator $T_{\mathcal{T}}$ defined in Definition 4.7 in new constraints. However, SHACL as introduced in [W3C17] can only propagate information on nodes, and information about the properties/roles connecting the nodes cannot be directly represented.

We discuss two ways around this. First we present a solution for TBoxes that do not contain counting axioms ($A \sqsubseteq \leq_1 r.B$). The role information added to $\mathcal{A}_{\mathcal{T}}$ in this case is simple, and it can be directly derived from locally satisfying axioms of the form $r_0 \sqcap \dots \sqcap r_n \sqsubseteq r$. For the general case, we extend SHACL and enable it to describe edges, rather than just nodes.

Without loss of generality, we assume in this section that there are no cycles over role inclusions. That is, no TBox contains a set of role inclusions of the form $\{r \sqsubseteq r_1, r_1 \sqsubseteq r_2, \dots, r_n \sqsubseteq r\}$. If such a cycle exists, then all roles names appearing in the cycle are replaced by one unique role name.

4.6.1 Rewriting for normalised Horn- $\mathcal{ALCH}\mathcal{I}$

Let us first focus on how to define a pure rewriting for normalised Horn- $\mathcal{ALCH}\mathcal{I}$.

Definition 4.46. A Horn- $\mathcal{ALCH}\mathcal{I}$ TBox $\mathcal{T}$ is in normal form if each of the concept inclusions in $\mathcal{T}$ are of one of the following forms:

$$\begin{array}{ll} (F1) & A_0 \sqcap \dots \sqcap A_n \sqsubseteq B \\ (F3) & A \sqsubseteq \forall r.B \\ (F4) & A \sqsubseteq \exists r.B, \end{array}$$

for $\{A, A_0, \dots, A_n, B\} \subseteq \overline{N}_C$ and $r \in \overline{N}_R$. Furthermore, $\mathcal{T}$ may contain role inclusions of the form $r \sqsubseteq r'$, for $\{r, r'\} \subseteq \overline{N}_R$.

For each Horn- $\mathcal{ALCH}\mathcal{I}$ TBox $\mathcal{T}$, we define the following set of constraints, capturing most of the information derived by the immediate consequence operator $T_{\mathcal{T}}$, defined in Definition 4.7.

Definition 4.47. For each $A \in N_C$, we let $s_A \in N_S$ be a fresh shape name. Given a normalised Horn- $\mathcal{ALCH}\mathcal{I}$ TBox $\mathcal{T}$, let $\mathcal{T}_s$ be the smallest set of constraints containing for each $\mathcal{T} \models$

$A_0 \sqcap \dots \sqcap A_n \sqsubseteq B$, the constraint

$$s_B \leftarrow \bigwedge_{0 \leq i \leq n} s_{A_i} \in \mathcal{T}_s, \quad (4.2)$$

furthermore, for each $A \sqsubseteq \forall r.B \in \mathcal{T}$ and $\mathcal{T} \models S \sqsubseteq r$ the constraint

$$s_B \leftarrow \exists S^-.s_A \in \mathcal{T}_s, \quad (4.3)$$

and for each $A \in \overline{N}_C$, the constraint

$$s_A \leftarrow A \in \mathcal{T}_s. \quad (4.4)$$

Definition 4.48. Given a normalised Horn- $\mathcal{ALCH}\mathcal{I}$ TBox $\mathcal{T}$, let $\mathcal{C}_{\mathcal{T}}^+$ be the set of constraints consisting of the constraints in $\mathcal{C}_{\mathcal{T}}$, taking into account the following alterations:

- each concept name $A \in N_C$ appearing in an axiom in $\mathcal{C}_{\mathcal{T}}$ is replaced by s_A ; and
- for each $\mathcal{T} \models \sqcap R \sqsubseteq r$, such that $R \cup \{r\} \subseteq R'$, and $\sqcap R'$ appears in some constraint, $\sqcap R'$ is replaced by $\sqcap (R' \setminus \{r\})$.

Note that when considering $\mathcal{C}_{\mathcal{T}}^+ \cup \mathcal{T}_s$ instead of $\mathcal{C}_{\mathcal{T}}$, we are introducing a new lowest stratum: the set of freshly introduced constraints, based on existing concept names. However, the only point of this stratum is to mimic exactly which concept names hold at which points in the enriched ABox $\mathcal{A}_{\mathcal{T}}$, defined in Definition 4.7.

Proposition 4.49. Given a normalised Horn- $\mathcal{ALCH}\mathcal{I}$ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of normalised, positive constraints, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $\text{can}_0(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}_{\mathcal{T}}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}_{\mathcal{T}}^+ \cup \mathcal{T}_s, \mathcal{G})$.

Proof. Note that $\text{can}_0(\mathcal{T}, \mathcal{A})$ corresponds to $\mathcal{A}_{\mathcal{T}}$, the first layer in building the austere universal model, defined in Definition 4.7. In this definition, the ABox is completed in a least fixed point manner under certain axioms ($A_0 \sqcap \dots \sqcap A_n \sqsubseteq B$ and $A \sqsubseteq \forall r.B$) that are precisely encoded in constraints of the form (4.2) and (4.3). The only difference is that we cannot translate axioms of the form $\mathcal{T} \models S \sqsubseteq r'$ directly. A simple fix suffices: as role atoms in $\mathcal{A}_{\mathcal{T}} \setminus \mathcal{A}$ can only be the result of an axiom of the form $r \sqsubseteq r'$, the replacement in Definition 4.48 in combination with the implied roles in constraints of the form (4.3) is sufficient. As we are also considering a least fixed point semantics for SHACL validation, the above result directly follows. $\clubsuit$

Thus, considering $\mathcal{C}_{\mathcal{T}}^+ \cup \mathcal{T}_s$ as rewritten set of constraints indeed provides us with a pure rewriting. The following theorem follows from combining the results of Theorem 4.44 and Proposition 4.49.

Theorem 4.50. Given a normalised Horn- $\mathcal{ALCH}\mathcal{I}$ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of normalised, stratified constraints, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}_{\mathcal{T}}^+, \mathcal{G})$.

$$\begin{aligned}
[\exists\psi.\varphi]_S^{\mathcal{I}} &:= \{e \in \Delta^{\mathcal{I}} \mid \exists e' \in \Delta^{\mathcal{I}} : (e, e') \in [\psi]_S^{\mathcal{I}} \wedge e' \in [\varphi]_S^{\mathcal{I}}\} \\
[s?]_S^{\mathcal{I}} &:= \{(e, e) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid e \in [s]_S^{\mathcal{I}}\} \\
[b]_S^{\mathcal{I}} &:= \{(e, e') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid b(e, e') \in S\} \\
[r]_S^{\mathcal{I}} &:= r^{\mathcal{I}} \\
[\psi \cup \psi']_S^{\mathcal{I}} &:= [\psi]_S^{\mathcal{I}} \cup [\psi']_S^{\mathcal{I}} \\
[\psi \cap \psi']_S^{\mathcal{I}} &:= [\psi]_S^{\mathcal{I}} \cap [\psi']_S^{\mathcal{I}} \\
[\psi \cdot \psi']_S^{\mathcal{I}} &:= \{(e, e') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists e'' \in \Delta^{\mathcal{I}} : (e, e'') \in [\psi]_S^{\mathcal{I}} \wedge (e'', e') \in [\psi']_S^{\mathcal{I}}\} \\
[\psi^*]_S^{\mathcal{I}} &:= \{(e, e') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists \{e_0, \dots, e_n\} \subseteq \Delta^{\mathcal{I}} : \\
&\quad \forall i \in \{0, \dots, n-1\} : (e_i, e_{i+1}) \in [\psi]_S^{\mathcal{I}} \wedge e_0 = e \wedge e_n = e'\} \\
[\psi^-]_S^{\mathcal{I}} &:= \{(e, e') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (e', e) \in [\psi]_S^{\mathcal{I}}\} \\
[\psi \setminus \psi']_S^{\mathcal{I}} &:= [\psi]_S^{\mathcal{I}} \setminus [\psi']_S^{\mathcal{I}}
\end{aligned}$$

Figure 4.5: Evaluation of SHACL^b shape expressions.

4.6.2 SHACL^b

For a pure rewriting of normalised Horn- $\mathcal{ALCHIQ}$, we propose a different solution: we define an extension of SHACL that we call SHACL^b. Here we also allow ‘binary’ shape constraints b : which may express constraints on pairs of nodes. We note that not all operators defined in SHACL^b are necessary to define a pure rewriting for normalised Horn- $\mathcal{ALCHIQ}$. However, we do resort to the following version to demonstrate how the imbalance between unary and binary (shape) atoms in SHACL can be fully reduced.

Definition 4.51. Let SHACL^b consist of shape constraints of the form $s \leftarrow \varphi$, or $b \leftarrow \psi$, for $\{s, b\} \subseteq N_S$, such that

$$\begin{aligned}
\varphi &::= c \mid s \mid A \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists\psi.\varphi \\
\psi &::= s? \mid b \mid r \mid \psi \cup \psi \mid \psi \cap \psi \mid \psi \cdot \psi \mid \psi^* \mid \psi^- \mid \psi \setminus \psi,
\end{aligned}$$

where $c \in N_I$, $\{s, b\} \subseteq N_S$, $A \in N_C$ and $r \in \overline{N}_R$.

We evaluate SHACL^b as defined in Figures 3.1 and 4.3, extended with $[\cdot]_S^{\mathcal{I}}$ as defined in Figure 4.5, where $[\cdot] \in \{\lfloor \cdot \rfloor, \lceil \cdot \rceil\}$. Note that these same functions can directly be used to define the well-founded semantics for SHACL^b, in the exact same way as the well-founded semantics for SHACL makes use of $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$.

The following shape expressions, although not explicitly present in the fragment we consider in

this chapter, are also expressible in SHACL^b.

$$\begin{aligned}\exists R.\varphi &= \exists(r_0 \sqcap \dots \sqcap r_n).\varphi \\ \text{eq}(E, E') &= \neg\exists.(E \setminus E').\top \wedge \neg\exists.(E' \setminus E).\top \\ \text{disj}(E, E') &= \neg\exists(E \sqcap E').\top\end{aligned}$$

where $R = r_0 \sqcap \dots \sqcap r_n$.

Definition 4.52. Define $s_A \in N_S$ and $b_r \in N_S$ to be a fresh shape names for each $A \in N_C$, respectively $r \in \overline{N}_R$. Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, let $\mathcal{T}_s$ be the smallest set of binary SHACL constraints containing for each $\mathcal{T} \models A_0 \sqcap \dots \sqcap A_n \sqsubseteq B$, the constraint

$$s_B \leftarrow \bigwedge_{0 \leq i \leq n} s_{A_i} \in \mathcal{T}_s, \quad (4.5)$$

for each $A \sqsubseteq \forall r'.B \in \mathcal{T}$ the constraint

$$s_B \leftarrow \exists b_{r^-}.s_A \in \mathcal{T}_s, \quad (4.6)$$

and, for each $\mathcal{T} \models A_0 \sqcap \dots \sqcap A_n \sqsubseteq \exists(r_0 \sqcap \dots \sqcap r_m).(B_0 \sqcap \dots \sqcap B_k)$ and $A \sqsubseteq \leq_1 r.B \in \mathcal{T}$ such that $r \in \{r_0, \dots, r_m\}$ and $B \in \{B_0, \dots, B_k\}$, the constraints

$$\{b_{r_i} \leftarrow \hat{s}^? \cdot b_r \cdot s_B^?, \quad \hat{s} \leftarrow s_A \wedge s_{A_0} \wedge \dots \wedge s_{A_n}, \quad (4.7)$$

$$s_{B_j} \leftarrow s_B \wedge \exists b_{r^-}.\hat{s}\} \subseteq \mathcal{T}_s, \quad (4.8)$$

for each $i \in \{0, \dots, m\}$ and $j \in \{0, \dots, k\}$, and a fresh shape name $\hat{s}$, furthermore, for each $r \sqsubseteq r' \in \mathcal{T}$, the constraint

$$b_{r'} \leftarrow b_r \in \mathcal{T}_s, \quad (4.9)$$

and for each $r \in \overline{N}_R$, $p \in N_R$ the constraints

$$\{b_r \leftarrow r, \quad b_{(p^-)} \leftarrow (b_p)^-, \quad b_p \leftarrow (b_{p^-})^-\} \subseteq \mathcal{T}_s, \quad (4.10)$$

and, lastly, for each $A \in \overline{N}_C$, the constraint

$$s_A \leftarrow A \in \mathcal{T}_s. \quad (4.11)$$

Proposition 4.53. Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, for each $\mathcal{A}$, we find that for all $A \in N_C$, all $r \in \overline{N}_R$, and all $\{a, b\} \subseteq \Delta^{\mathcal{I}_A}$,

- $A(a) \in \mathcal{A}_{\mathcal{T}}$ iff $s_A(a) \in PA(\mathcal{T}_s, \mathcal{A})$;
- $r(a, a') \in \mathcal{A}_{\mathcal{T}}$ iff $b_r(a, a') \in PA(\mathcal{T}_s, \mathcal{A})$.

Proof. Let $\mathcal{T}_s^*$ consists of the constraints in $\mathcal{T}_s$ defined by (4.10) and (4.11). Then, it is clear that $A(a) \in \mathcal{A}$ iff $s_A(a) \in PA(\mathcal{T}_s^*, \mathcal{A})$ and $r(a, a') \in \mathcal{A}$ iff $b_r(a, a') \in PA(\mathcal{T}_s^*, \mathcal{A})$. This means that all information of the ABox can be precisely captured in shape names. For the rest, note how the different lines in Definition 4.7 defining the immediate consequence operator used to build $\mathcal{A}_{\mathcal{T}}$ from $\mathcal{A}$ correspond one-to-one to the constraints defined in (4.5) to (4.9), with the exception of using the shorthand $\hat{s}$. As both the perfect assignment PA as the construction of $\mathcal{A}_{\mathcal{T}}$ are based on a least fixed point semantics, the result follows. ♣

The completeness and correctness of the full rewriting now directly follow. Clearly, we concluded that all information in $\mathcal{A}_{\mathcal{T}}$, albeit in the form s_A and b_r , can be derived by SHACL^b constraints constructed to this end. Therefore, there is not really a difference between evaluating $\mathcal{C}_{\mathcal{T}}$ over $\mathcal{A}_{\mathcal{T}}$, or $\mathcal{C}_{\mathcal{T}} \cup \mathcal{T}_s$ over $\mathcal{A}$, except that we should replace concept names and roles by their shape names referring to them.

Definition 4.54. *Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$, let $\mathcal{C}_{\mathcal{T}}^+$ be the set of constraints consisting of the axioms in $\mathcal{C}_{\mathcal{T}}$, such that each concept name $A \in N_C$ appearing in an axiom in $\mathcal{C}_{\mathcal{T}}$ is replaced by s_A , and similarly, each $r \in \overline{N}_R$ by b_r .*

Now that the concept and role names are replaced by their shape equivalents, we have indeed produced a pure rewriting.

Theorem 4.55. *Given a normalised Horn- $\mathcal{ALCHIQ}$ TBox $\mathcal{T}$ and $\mathcal{C}$ a set of stratified constraints, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}_{\mathcal{T}}^+ \cup \mathcal{T}_s, \mathcal{G})$.*

4.7 Complexity Results

We now discuss the computational complexity of SHACL validation in the presence of Horn- $\mathcal{ALCHIQ}$ TBoxes. Specifically, we discuss the *combined complexity* and the *data complexity* of the problem [Var82]. The former is measured in terms of the combined size of all input components, while the latter is measured assuming all components except the ABox are of fixed size.

Theorem 4.56. *The problem of SHACL validation for stratified set of constraints expressible in the fragment of SHACL introduced in Section 4.3 in the presence of normalised Horn- $\mathcal{ALCHIQ}$ TBoxes is EXPTIME-complete in combined complexity and PTIME-complete in data complexity.*

Proof. For data complexity, the PTIME lower bound was shown in [ACO⁺20], which already applies in the absence of ontologies. The matching PTIME upper bound follows from Theorem 4.44, combined with the fact that $\mathcal{A}_{\mathcal{T}}$ has size polynomial in $\mathcal{A}$'s size itself, and that validation under stratified constraints without ontologies is feasible in polynomial time in data complexity [ACO⁺20]. We note that checking whether the input graph is consistent with a TBox can also be done in polynomial time.

For the upper bound in combined complexity, we rely on the rewriting algorithm discussed in Sections 4.4 and 4.5. Let $(\mathcal{T}, \mathcal{A})$ be a normalised Horn- $\mathcal{ALCHIQ}$ knowledge base and $\mathcal{C}$ any set of (stratified) constraints. Observe that the number of different quadruples (t, P, Q, H) over the signature of $\mathcal{T}$ and $\mathcal{C}$ that can be added to any K_i during the rewriting is bounded by an exponential in the size of $\mathcal{T}$ and $\mathcal{C}$. An application of any of the rules takes polynomial time in the size of $\mathcal{T}$, $\mathcal{A}$, $\mathcal{C}$ and $\bigcup_{i>0} K_i$, which means the application of all rules in the procedure is bounded by an exponential in the size of $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{C}$. Computing $\mathcal{C}_{\mathcal{T}, K}$ is polynomial in the size of K , for all possible K . Thus, overall we get a procedure that runs in exponential time in the size of $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{C}$.

The hardness result in combined complexity follows from the hardness of reasoning in the ontology alone: checking whether an atom of the form $r(a, b)$ or $A(a)$ is implied by a knowledge base is already EXPTIME-complete in combined complexity [EOŠ⁺12]. ♣

The hardness inherited from the ontology alone is not the only source of complexity. EXPTIME-hardness holds even for very weak description logics like $DL-Lite_{\mathcal{R}}$ and a very simple fragment of SHACL.

Define a *simple* shape expression φ in the following way

$$\varphi ::= s \mid A \mid \varphi \wedge \varphi \mid \exists r. \varphi,$$

for $s \in N_S$, $A \in N_C$ and $r \in \overline{N}_R$. Shape constraints in *simple*-SHACL are then build in the same way as for regular SHACL by using simple shape expressions instead of regular ones. All other definitions extend in a straightforward way to simple-SHACL.

Theorem 4.57. *Let $\mathcal{L}$ be a DL that support axioms of the forms $A \sqsubseteq \exists r. \top$ and $\exists r_1. \top \sqsubseteq \exists r_2. \top$, for $\{r, r_1, r_2\} \subseteq \overline{N}_R$. In the presence of $\mathcal{L}$ TBoxes, simple-SHACL validation is EXPTIME-complete in combined complexity.*

Proof. Membership follows directly from Theorem 4.56. To prove EXPTIME-hardness in combined complexity we reduce the word problem of polynomially space-bounded *Alternating Turing Machines* (ATMs) to validating shapes under an $\mathcal{L}$ ontology.

An *ATM* is defined as a tuple of the form

$$\mathcal{M} = (\Sigma, Q_{\exists}, Q_{\forall}, q_0, q_{acc}, q_{rej}, \delta)$$

where Σ is an *alphabet*, $Q_{\exists}$ is a set of *existential states*, $Q_{\forall}$ is a set of *universal states*, disjoint from $Q_{\exists}$, $q_0 \in Q_{\exists}$ is an *initial state*, $q_{acc} \in Q_{\exists} \cup Q_{\forall}$ is an *accepting state*, $q_{rej} \in Q_{\exists} \cup Q_{\forall}$ is a *rejecting state*, and δ is a *transition relation* of the form

$$\delta \subseteq Q \times (\Sigma \cup \{B\}) \times Q \times (\Sigma \cup \{B\}) \times \{-1, 0, +1\}$$

with $Q = Q_{\exists} \cup Q_{\forall}$. Here B is the *blank symbol*. We let $\delta(q, a) = \{(q', b, D) \mid (q, a, q', b, D) \in \delta\}$. W.l.o.g., we assume that in $\mathcal{M}$ universal and existential states are strictly alternating: if

$(q, a, q', b, m) \in \delta$ and $q \in Q_{\exists}$ (resp., $q \in Q_{\forall}$), then $q' \in Q_{\forall}$ (resp., $q' \in Q_{\exists}$). We further assume that $|\delta(q, a)| = 2$ for all combinations of states $q \in Q$ and symbols $a \in \Sigma \cup \{B\}$. If $\delta(q, a) = \{(q_1, a_1, D_1), (q_2, a_2, D_2)\}$, we let $\delta_\ell(q, a) = (q_1, a_1, D_1)$ and $\delta_r(q, a) = (q_2, a_2, D_2)$.

A run of an ATM $\mathcal{M}$ on an input word w is defined as usual. A run is accepting if we can find a subtree such that all leaves are accepting states and such that on existential states, at least one of the successors is contained in the subtree, whereas in universal states, both successors are contained. We assume a word $w = d_1 \cdots d_n \in \Sigma^*$ with $n > 0$ together with an ATM $\mathcal{M}$ that only uses the tape cells where the input word was written, i.e., it only uses the first n cells, which can be shown by padding arguments (see for instance [CGK13] for similar assumptions). Checking if such $\mathcal{M}$ accepts w is an EXPTIME-hard problem.

We show how to construct a knowledge base $(\mathcal{T}, \mathcal{A})$ and SHACL document $(\mathcal{C}, \{s_{acc}(a)\})$ such that $\mathcal{M}$ accepts w iff $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \{s_{acc}(a)\})$. The reduction takes polynomial time in the size of $\mathcal{M}$ and w . It uses the following symbols:

- a concept name $Init$ and a shape name s_{acc} ;
- role names $succ_\ell, succ_r$;
- shape names s_q for all states $q \in Q_{\exists} \cup Q_{\forall}$;
- shape names h_i for all $1 \leq i \leq n$;
- shape names $c_b^{(i)}$ for all $b \in \Sigma \cup \{B\}$ and $1 \leq i \leq n$.

Set $\mathcal{A} = \{Init(a)\}$ and let $\mathcal{T}$ contain the following inclusions, for each $\gamma \in \{\ell, r\}$

$$Init \sqsubseteq \exists succ_\gamma. \top \quad \exists succ_\ell. \top \sqsubseteq \exists succ_\gamma. \top \quad \exists succ_r. \top \sqsubseteq \exists succ_\gamma. \top.$$

The interpretation $can(\mathcal{T}, \mathcal{A})$ provides us with an infinite binary tree. In that tree, the root is representing the starting configuration of $\mathcal{M}$ and each child of a node represents a next step in the run of the ATM $\mathcal{M}$.

To mimic the start configuration, we define the following shapes:

$$h_1 \leftarrow Init \quad s_{q_0} \leftarrow Init \quad c_{d_i}^{(i)} \leftarrow Init \text{ for all } 1 \leq i \leq n.$$

Intuitively, this is setting the starting state to q_0 (denoted by the shape name s_{q_0}), putting the head in the starting position (h_1), and stating the input word written on each tape cell ($c_{d_i}^{(i)}$).

The next step is to encode the transition relation of the $\mathcal{M}$. For each $1 \leq i \leq n$, each $(q, a) \in Q \times (\Sigma \cup \{B\})$, and $\gamma \in \{\ell, r\}$ we add the following shapes, where $(q', b, D) = \delta_\gamma(q, a)$:

$$\begin{aligned} s_{q'} &\leftarrow \exists succ_\gamma. (s_q \wedge h_i \wedge c_a^{(i)}) \\ c_b^{(i)} &\leftarrow \exists succ_\gamma. (s_q \wedge h_i \wedge c_a^{(i)}) \\ h_{i+D} &\leftarrow \exists succ_\gamma. (s_q \wedge h_i \wedge c_a^{(i)}). \end{aligned}$$

Furthermore, the tape cells that are not under the read-write head have their content preserved. Thus, for each $\gamma \in \{\ell, r\}$, $1 \leq i \leq n$, and $1 \leq j \leq n$ with $i \neq j$, add

$$c_a^{(i)} \leftarrow \exists succ_{\gamma}^-. (c_a^{(i)} \wedge h_j).$$

We now identify subtrees that represent accepting computations. For all $q \in Q_{\exists}$ and all $q' \in Q_{\forall}$ we add the following:

$$\begin{aligned} s_{acc} &\leftarrow s_{q_{acc}} \\ s_{acc} &\leftarrow s_q \wedge \exists succ_{\ell}. s_{acc} \\ s_{acc} &\leftarrow s_q \wedge \exists succ_r. s_{acc} \\ s_{acc} &\leftarrow s_{q'} \wedge \exists succ_{\ell}. s_{acc} \wedge \exists succ_r. s_{acc}. \end{aligned}$$

This concludes the reduction.

As we assumed $|\delta(q, a)| = 2$ for all $q \in Q$, $a \in \Sigma \cup \{B\}$, the run of $\mathcal{M}$ on an input word w can be thought of as a binary tree with nodes in $\{0, 1\}^*$ and edges $succ_{\ell} := \{(x, x0) \mid x \in \{0, 1\}^*\}$ and $succ_r := \{(x, x1) \mid x \in \{0, 1\}^*\}$, such that the root, ϵ , represents the input word, combined with the starting position of the head and the name of the starting state; and each other node of this tree represents the next configurations of the run, such that if some node x represents the word $e_1 \cdots e_n \in \Sigma^*$, with the head in position i and state name q , such that $\delta_{\ell}(q, e_i) = (q', a', D')$, then the left-child of this node, $x0$, represents the word $e_1 \cdots e_{i-1} a' e_{i+1} \cdots e_n$, head position $i + D$ and state name q' , and similarly for the child on the right, $x1$, and $\delta_r(q, e_i)$.

Let h be an homomorphism of $\{0, 1\}^*$ into $can(\mathcal{T}, \mathcal{A})$. A simple check suffices to see that x represents the word $e_1 \cdots e_n$, head position i and state name q iff $h(x)$ is decorated with the shape names $c_{e_i}^{(i)}$ for each $1 \leq i \leq n$, h_i and s_q in the perfect assignment.

An accepting run of $\mathcal{M}$ on the input word w is a binary tree as described above such that there exists a (finite) subtree in which, if x represents a state name $q \in Q_{\exists} \setminus \{q_{acc}\}$, then either $x0$ or $x1$ is part of the subtree, and if x represents a state name $q \in Q_{\forall} \setminus \{q_{acc}\}$, then both $x0$ and $x1$ are part of the subtree. As we already concluded that x represents a state q iff $h(x)$ is decorated with s_q in the perfect assignment, and a decoration with s_{acc} requires a state to be an accepting state, or one successor to be decorated with s_{acc} in case of x representing a $Q_{\exists}$ state, resp. two in case of a $Q_{\forall}$ state, it is immediate that there exists such a subtree with all leaves representing the state name q_{acc} iff we can decorate the root node a in $can(\mathcal{T}, \mathcal{A})$ with the shape name s_{acc} . Thus, $\mathcal{M}$ accepts w iff $can(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \{s_{acc}(a)\})$.

♣

4.8 Discussion and Conclusion

Beyond normalised Horn- $\mathcal{ALCHIQ}$. In this chapter we considered Horn- $\mathcal{ALCHIQ}$ TBoxes in normal form. Unfortunately, we cannot easily lift the results to general Horn- $\mathcal{ALCHIQ}$, as our techniques are not immune to the usual normalisation procedures. For instance, the

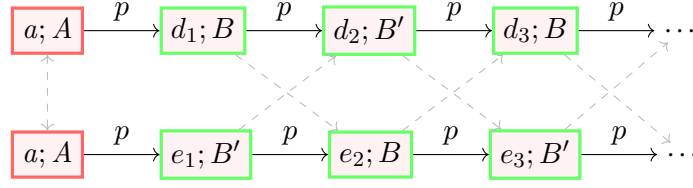


Figure 4.6: Two core universal models (after taking the transitive closure of p) of one knowledge base. Two of the non-surjective homomorphisms are denoted by the dashed, grey arrows.

procedures in [Kaz09] and [KRH07] may significantly change the form of the universal core model; the fresh concepts introduced during normalisation may force us to use different objects to satisfy axioms that would otherwise be satisfied by the same object. Take for instance the axiom $A \sqsubseteq \geq 2 r.B$ and the ABox $\mathcal{A} = \{A(a), p(a, b), B(b), p(a, c), B(c)\}$. In this case, $\mathcal{A}$ itself is a core universal model already. However, if we follow the normalisation as proposed in Lemma 2 of [Kaz09], we would find the following set of axioms $\mathcal{T} = \{A \sqsubseteq \exists p.B_1, A \sqsubseteq \exists p.B_2, B_1 \sqsubseteq B, B_2 \sqsubseteq B, B_1 \sqcap B_2 \sqsubseteq \perp\}$. It is easy to check that in the core universal model $\text{can}(\mathcal{T}, \mathcal{A})$, a has four outgoing p -edges to a B , instead of two. We believe it is possible to obtain similar results for full Horn- $\mathcal{ALCHIQ}$, albeit with a more cautious normalisation and a weaker notion of homomorphism. We leave the details for further work.

It would also be desirable to add transitivity axioms and thus cover the well-known Horn- $\mathcal{SHIQ}$ description logic. But unfortunately, in the presence of transitive roles, the uniqueness of the core universal model can no longer be guaranteed: take for instance $\mathcal{A} = \{A(a)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists p.B, A \sqsubseteq \exists p.B', B \sqsubseteq \exists p.B', B' \sqsubseteq \exists p.B\}$ where p is transitive. In this case, there are two (not strong) core universal models: a chain of p 's such that the concept names along this chain are, in the first node A , and then B and B' , in an alternating fashion, closed under transitive roles, and the version in which B and B' swap places, as is illustrated in Figure 4.6. Note that there exist (non-surjective) homomorphisms between both models. It is unclear what would be a good semantics if multiple core universal models exist.

Towards full SHACL. Extending the considered SHACL fragment is also a promising avenue. However, we first point that the ‘guard’ a we introduced for constraints of the forms $\text{eq}(E, r)$ and $\text{disj}(E, r)$, cannot easily be dropped. Without it, the normalisation as described in the proof of Proposition 4.29 does not work well. There is no straightforward way to distinguish from which starting point it was possible to reach a final state, as becomes clear in the following example.

Example 4.58. Let $\mathcal{A} = \{r(a, c), t(b, c)\}$. Over this graph, we wish to check validation for the constraint $s \leftarrow \text{eq}(r, t)$ with targets $s(a)$ and $s(c)$. Clearly, we expect non-validation. However, this cannot easily be achieved by a similar rewriting technique.

To see this, let us consider the following constraints as, loosely following the construction above to rewrite $s \leftarrow \text{eq}(r, t)$ and based on the automata $\mathcal{M} = (\{q, q'\}, \{r, t\}, q, \{(q, r, q')\}, q')$ and $\mathcal{M}' = (\{q'', q'''\}, \{r, t\}, q'', \{(q'', t, q''')\}, q''')$ we find the following constraints:

$$\begin{array}{lllll}
s_q \leftarrow a & s_q \leftarrow c & s_{q'} \leftarrow \exists r^-.s_q & s_{pos} \leftarrow s_{q'} & s_{neg} \leftarrow \neg s_{q'} \\
s_{q''} \leftarrow a & s_{q''} \leftarrow c & s_{q'''} \leftarrow \exists t^-.s_{q''} & s_{pos} \leftarrow s_{q'''} & s_{neg} \leftarrow \neg s_{q'''} \\
s_e \leftarrow s_{pos} \wedge s_{neg} & s_e \leftarrow \exists r.s_e & s_e \leftarrow \exists t.s_e & s_{noe} \leftarrow \neg s_e & s \leftarrow s_q \wedge s_{noe}
\end{array}$$

where we shortened the names of the error-shapes to s_e and s_{noe} . As b is reachable from starting states a and c by both an r and a t , b will only be assigned s_{pos} , the shape name denoting being the final state of at least one of the automata, not causing any error-shape s_e to fire, which breaks our expectation of non-validation. The problem is that we cannot distinguish which initial state caused the positive outcome. $\heartsuit$

For non-recursive SHACL, it is known that adding path equality or disjointness increases expressivity [BJVdB24], but we do not know whether this also holds for recursive SHACL. In any case, it seems hard to include full path equality and disjointness due their ‘non-local’ nature.

On the other hand, it might be possible to encode local counting constraints of the form $\exists_{\leq n} r.\varphi$, after a careful examination of our techniques. Extending counting over roles to counting over regular paths is even more interesting, and will be the topic of Chapter 6. It will be quite a challenge to also define a rewriting in which the quadruples store information on counting over regular paths. However, as the anonymous parts of the core universal model are tree-like, it might be possible to also achieve such a rewriting.

Richer Targets. In parallel to increasing the expressivity of the considered shape expressions, we may also consider richer targets, as introduced in Chapter 3. That is, additionally to shape atoms, the SHACL standard considers *target classes*: if a concept name A is given as target, all nodes in the interpretation of A must validate the given shape. For plain validation, without ontologies, class targets reduce trivially to sheer shape atoms: the set of domain elements is known and can replace the class or concept name. But if ontology axioms are given, this is a whole different story: the set of domain elements that forms the interpretation of the target concept name is not known a priori. Nevertheless, under the assumption that the input data graph is connected, it is possible to model this for $s \leftarrow \varphi$ with target class A , by creating new constraints of the form “there does not exists a path (the Kleene star of the union of all roles appearing) to an A that does not satisfy φ ”. Now any domain element in the graph can be considered as the target to get a logically equivalent constraint. Clearly, a similar trick works if it is known which parts of the graph are connected.

Finally, we would like to emphasise that allowing full negation would be a very interesting, but complicated challenge.

Conclusion. We have considered the validation of SHACL constraints in the presence of normalised Horn- $\mathcal{ALCHIQ}$ ontologies. To this end, we defined the semantics over a carefully constructed notion of a universal model that minimises the number of fresh successors introduced to satisfy the ontology axioms at each chase step and which happens to be the unique core universal model. Moreover, we have argued that this semantics is natural and intuitive. We proposed a normal form and a rewriting algorithm for recursive SHACL constraints with *stratified*

negation. It takes as an input a SHACL document and an ontology, and constructs a new SHACL document (also with stratified negation) that can be used for sound and complete validation over a slightly extended version of the data graph alone, without needing to reason about the ontology at validation time. We also discussed some approaches to even get to validation over the pure data graph. We showed that, under our semantics, validation in the presence of normalised Horn- $\mathcal{ALCHIQ}$ is complete for EXPTIME, but it remains PTIME complete in data complexity, and hence it is not harder than validation of stratified SHACL alone, without the ontology axioms.

4.9 Related Work

There is a whole line of works on the use cases of universal models, e.g. [BO15, CCK⁺17, CGL⁺11, EOŠ⁺12, ORŠ11, PMH10] and their references. In particular, it is shown how they can be used for evaluating conjunctive, navigational and graph queries in the presence of ontologies. However, not all nice properties are retained for queries that involve negation as failure. Those queries are highly desirable and have been advocated for in several papers [BF19, AGP18, EKM22]. Apparently innocuous differences in the chase procedure may affect the query answers, and the certain answer semantics becomes inadequate; see, e.g., the discussion in [EKM22].

In this chapter we consider the austere universal model, of which the starting idea of locally minimising the amount of successors is inspired by a related setting [BF19]. However, there are more types of minimal universal models considered in the literature on combining open- and closed-world reasoning. One of these options is using the *minimal models* of the Skolemisation of the DL ontology, which has been advocated for in the case of integrity constraints [MHS07b, SKL19].

However, such models give very weak semantics in formalisms with negation such as SHACL. Let us for example consider another version of the toy database containing pet owners as discussed in the introduction of this chapter: assume it consists of $hasPet(john, ace)$, $PetOwner(john)$ and $Hamster(ace)$. It is conceivable to find an axiom like $PetOwner \sqsubseteq \exists hasPet.\top$ in the accompanying ontology. Now let the *onlyHamsterShape* be given by the constraint enforcing all pets to be hamsters, $onlyHamsterShape \leftarrow \forall hasPet.Hamster$, which we want to validate for *john*. This is clearly the case for the original database. It is also true that the given database is already satisfying all given ontology axioms, so it seems there is no reason to change the validation result. However, the minimal model under the Skolem semantics adds a fresh node as a *hasPet*-child of *john*, without the label *Hamster*, changing the validation result to negative.

The austere universal model being a core provides strong evidence in favour of our chosen semantics, since cores are often advocated as the adequate choice for languages that are not closed under homomorphisms, but satisfy the weaker property of being closed under isomorphisms [AGP18, BF19, EKM22]. In the light of this relationship, our austere model construction provides a novel technique for building core models without the expensive core-checking step of traditional core chase procedures. As we point out, the same applies to some previous model constructions from the DL literature [BF19, EOŠ⁺12]. Moreover, the austere universal model coincides with the results of both the core as the stable chase whenever these terminate [DNR08, CKM⁺18].

Cores were originally only defined for finite graphs, for which there are multiple, equivalent

definitions. However, on infinite structures, these definitions no longer coincide. That is, there are different types of cores considered over infinite structures [Bau95, Bau96]. For Horn- $\mathcal{ALCHIQ}$, we find that in all core universal models we have that each endomorphism is an isomorphism, which corresponds to the stronger variant of being a core. However, for Horn- $\mathcal{SHIQ}$, as becomes clear from the example in Figure 4.6, there are endomorphisms (for instance the concatenation of the two homomorphisms depicted in the figure) that are not surjective. That is, in this specific example, we have a Horn- $\mathcal{SHIQ}$ knowledge base that has two core universal models, and zero strong core universal models.

Rewriting techniques as presented in this chapter were believed to be non-existent: the impossibility of such a rewriting for SHACL with negation was claimed in Theorem 1 of [SKL19]. However, their proof and theorem do not hold, neither for our semantics nor for the minimal-model semantics adopted in that work, as acknowledged by the authors in personal communication, and shown by the results of this work. This is positive news: developing rewriting techniques is a promising approach towards practical integrations of SHACL and OWL.

Alternatively, when considering SHACL without negation, the rewriting presented in this chapter is related to rewritings of queries over knowledge bases into datalog: both the rewritten shapes (in which negation is not needed) as the rules to complete the ABox can be considered datalog rules. Such rewritings have been studied in depth, see for instance [HMS07, BtCLW14, BO15, AOS20] and many more. The main related work in this direction is the work on query rewriting for Horn- $\mathcal{SHIQ}$ by Eiter et al. [EOŠ⁺12]. This work considers conjunctive queries, which means that all ways how these queries can be seen as tree-like structures need to be guessed to fit the universal model structure. The positive case of the problem studied in this work has a different challenge: the SHACL shapes are already tree-shaped, but can refer to each other, making the possible structures to be rewritten an infinite set, which inspired to consider the building blocks of the universal core model instead. The calculus to determine whether $\mathcal{T} \models C \sqsubseteq D$ in Horn- $\mathcal{SHIQ}$ presented in that work is of essential value to make our construction possible.

The work by Borgwardt and Forkel [BF19] considers conjunctive queries with guarded negation over $\mathcal{ELH}_\perp$ ontologies. Their rewriting approach considers filters to store the structure of rewritten negated parts of the query, to make sure they are satisfied by a node that behaves in the right way. We on the contrary use the normal form of SHACL to break up the constraints into small pieces that can be fully evaluated in the environments described by the types. In this way, we can address the stratified negation and recursion in the rewriting.

Towards Practical Validation

Towards a practical approachable algorithm, we take a closer look at the definitions and rewriting techniques presented in the previous chapter. In particular, we will restrict the ontology language to $\mathcal{ELHI}$ and $DL-Lite_{\mathcal{R}}$, and the SHACL fragment to be semi-positive, that is, no negation in front of shape names, and show in which way this simplifies some techniques: for instance, the need to store negative information in the quadruples mostly disappears, as we no longer have to do the complement reasoning to make the step to the next stratum.

As seen in Section 4.7, we find that SHACL validation in presence of ontologies, even for less expressive logics like $\mathcal{ELHI}$ or $DL-Lite_{\mathcal{R}}$, remains EXPTIME-complete in combined complexity. That is, it will not be possible to find algorithms that will not cause any exponential blow-up. However, the rewriting as presented before is best-case exponential, which means there is room for improvement in finding an algorithm that is only worst-case exponential. The main cause for this blow-up is that we consider all possible ways how a small part of the data looks like, resulting in having to take among others all satisfiable subsets of concepts.

That is, to go around this, for $\mathcal{ELHI}$, we consider a *data-aware* version of the techniques presented. This means we have to compromise its applicability of the rewritten shapes to datasets that conform to certain constraints. Before we get into $\mathcal{ELHI}$ in Section 5.2, we look at the definitions for $DL-Lite_{\mathcal{R}}$ in Section 5.1, which highlight the intuition that in $DL-Lite_{\mathcal{R}}$ only the set of incoming roles is relevant, and defines the whole shape of the tree below in the core universal model.

5.1 $DL-Lite_{\mathcal{R}}$

Here, we look into how to compute a finite representation of a core universal model for a given $DL-Lite_{\mathcal{R}}$ KB. In a nutshell, recall that the good successor configuration tells us for each configuration of a node and its immediate ‘neighbourhood’, the precise configuration of blank nodes that should be introduced as immediate successors. In the construction of the *austere*

universal model, note that the idea is that each chase step introduces as few fresh successors as possible, without merging any successors unless the TBox is forcing us to do so.

However, note that for $DL\text{-}Lite_{\mathcal{R}}$, in comparison to the definition for Horn- $\mathcal{ALCHIQ}$, we need to save much less information. Because axioms have such a limited reach in $DL\text{-}Lite_{\mathcal{R}}$, it turns out it is sufficient to only store one role name in the description of the different successors.

Recall that a *basic concept* is a concept name $A \in N_C$ or an expression of the form $\exists r.\top$ such that $r \in \overline{N}_R$.

Definition 5.1. For a set of roles R , let $cl_{\mathcal{T}}(R)$ be the set of all roles r' such that $\mathcal{T} \models r \sqsubseteq r'$ and $r \in R$. Given a $DL\text{-}Lite_{\mathcal{R}}$ TBox $\mathcal{T}$ and a set of basic concepts U , a *good successor configuration* $succ_{\mathcal{T}}(U)$ for U is a possibly empty set of roles such that:

- (R1) If $C \in U$, $\mathcal{T} \models C \sqsubseteq \exists r.\top$ and $\exists r.\top \notin U$, then there exists $r' \in succ_{\mathcal{T}}(U)$ such that $r \in cl_{\mathcal{T}}(\{r'\})$;
- (R2) If $r \in succ_{\mathcal{T}}(U)$, then there exists $C \in U$ such that $\mathcal{T} \models C \sqsubseteq \exists r.\top$;
- (R3) There does not exist $\{r, r'\} \subseteq succ_{\mathcal{T}}(U)$ such that $r \in cl_{\mathcal{T}}(\{r'\})$;
- (R4) If $r \in succ_{\mathcal{T}}(U)$, then $r \notin cl_{\mathcal{T}}(\{r \mid \exists r.\top \in U\})$.

This is illustrated in the following example.

Example 5.2. Consider the TBox $\mathcal{T}$ containing three axioms:

$$\begin{aligned} PetOwner &\sqsubseteq \exists hasPet.\top & hasWingedPet &\sqsubseteq hasPet \\ PetOwner &\sqsubseteq \exists hasWingedPet.\top \end{aligned}$$

The good successor configuration $succ_{\mathcal{T}}(\{PetOwner\})$ is the set $\{hasWingedPet\}$, which trivially satisfies (R1) to (R4). Indeed $succ_{\mathcal{T}}(\{PetOwner\})$ cannot contain $hasPet$ by (R3).

All other sets of basic concepts will have empty sets as successor configurations. In particular, the set $U = \{PetOwner, \exists hasWingedPet.\top\}$ cannot have any of the roles of the TBox as a successor role due to (R4), since $cl_{\mathcal{T}}(\{hasWingedPet\}) = \{hasWingedPet, hasPet\}$.

A direct consequence of Proposition 4.5 is the following.

Lemma 5.3. For each set of basic concepts U , there exists a unique good successor configuration $succ_{\mathcal{T}}(U)$ for U .

The good successor configuration prescribes how to satisfy the TBox axioms locally. For $DL\text{-}Lite_{\mathcal{R}}$, we use it in a similar way to build a universal model layer by layer that we call *austere*.

Definition 5.4. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $DL\text{-}Lite_{\mathcal{R}}$ KB, let $N_I(\mathcal{A})$ be the set of individuals occurring in $\mathcal{A}$ and let $N_{\mathcal{K}}$ be the set of finite words of the form $ar_1r_2 \dots r_n$, with $n \geq 1$, $a \in N_I(\mathcal{A})$ and $r_1, \dots, r_n$ are roles, such that the following hold:

1. Let $U_a = \{A \mid A(a) \in \mathcal{A}\} \cup \{\exists r. \top \mid r(a, a') \in \mathcal{A}\}$, then $r_1 \in \text{succ}_{\mathcal{T}}(U_a)$;
2. For every $1 \leq i < n$, $r_{i+1} \in \text{succ}_{\mathcal{T}}(U_{r_i})$, where $U_{r_i} = \{\exists r_i^-. \top\}$.

We denote by $\text{tail}(w)$ the last role appearing in a word $w \in N_{\mathcal{K}}$. The austere universal model $\text{can}(\mathcal{K})$ of $\mathcal{K}$ is the interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}} := N_I(\mathcal{A}) \cup N_{\mathcal{K}}$ such that for all $a \in N_I(\mathcal{A})$, concept names A and roles r , the following hold:

1. $a^{\mathcal{I}} := a$;
2. $A^{\mathcal{I}} := \{a \mid \mathcal{K} \models A(a)\} \cup \{w \in N_{\mathcal{K}} \mid \mathcal{T} \models \exists \text{tail}(w)^-. \top \sqsubseteq A\}$;
3. $r^{\mathcal{I}} := \{(a, a') \mid \mathcal{K} \models r(a, a')\} \cup \{(w_1, w_2) \mid w_2 \in N_{\mathcal{K}}, w_2 = w_1 r', r \in \text{cl}_{\mathcal{T}}(\{r'\})\}$.

Just like for Horn- $\mathcal{ALCHIQ}$, we find that for each consistent $DL\text{-}Lite_{\mathcal{R}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the austere universal model $\text{can}(\mathcal{K})$ exists and is unique. Moreover, all other results, like it being the unique core universal model, naturally also transfer to the here presented version for $DL\text{-}Lite_{\mathcal{R}}$.

5.2 Data-Aware Good Successor Configuration

In contrast to $DL\text{-}Lite_{\mathcal{R}}$, in $\mathcal{ELHI}$ the relevant context is not only the incoming roles r , but also the concepts satisfied at the already existing neighbours. For simplicity and as part of the data-aware approach, we focus on describing the good successors of the blank nodes introduced during the chase, which have a unique predecessor in the forest-like structure. For describing the successor configurations of $\mathcal{ELHI}$, we return to the usage of types: recall the 2-, and $1\frac{1}{2}$ -types as introduced in Subsection 4.1.1. Furthermore, we extend the *inverse* function inv to $1\frac{1}{2}$ -types by setting $\text{inv}(u) := (\pi_2(u), (\pi_1(u))^-)$. (Recall that π_1 and π_2 are the first and second projection of a tuple, respectively, as defined in the preliminaries.)

Defining the successors of all types that may occur in the chase for any possible ABox would necessarily lead to a strictly exponential behaviour. While exponentiality is in general unavoidable, as standard reasoning tasks are EXPTIME complete, we want to avoid it for as many instances as possible. Hence in the following we build the good successor configuration carefully, to avoid best-case exponential behaviour. We start from the input ABox $\mathcal{A}$, and extend it to satisfy all applicable axioms, but we restrict the satisfaction of existential axioms to only create blank nodes that are directly neighbouring the ABox. We call this structure $\mathcal{A}_1$. The freshly introduced blank nodes in $\mathcal{A}_1$ have exactly one predecessor and can thus be described by a type. We take exactly those types as initial *live types*, and compute the *pre-good* configuration of successors that each live type needs. In each step of the computation we add to the live types the new successor types. It might happen that we encounter a so-called *loop computation*: the created successors may propagate some concept B back to the current node. This information is saved in a relation L as the pair of the type and the implied concept, and the pre-good successor pairs are updated with this information. Unlike $DL\text{-}Lite_{\mathcal{R}}$, derived information in $\mathcal{ELHI}$ can propagate through the model, and in particular, affect the concepts satisfied by ABox individuals. Thus the

information from the loop computation L must be used to update also the extended ABox, from which we retrieve fresh, updated live types for the next round of building pre-good successors. This continues in rounds until the process terminates.

Finally, let $\frac{\mathcal{T}, N_I}{\rightarrow}$ be the restriction of $\frac{\mathcal{T}}{\rightarrow}$ to matches in N_I , defined precisely as follows. Recall the definition of matches and the function f_e as defined in Section 4.2.

Definition 5.5. For an $\mathcal{ELHI}$ TBox $\mathcal{T}$, a set $I \subseteq N_I \cup N_B$ and ABoxes $\mathcal{A}, \mathcal{A}'$, we let $\mathcal{A} \xrightarrow{\mathcal{T}, I} \mathcal{A}'$ if

$$\begin{aligned} \mathcal{A}' = \mathcal{A} \cup & \bigcup_{(e, C \sqsubseteq \exists r.B) \in m(\mathcal{T}, \mathcal{A}), e \in I} f_e(\exists r.B) \cup \bigcup_{(e, C \sqsubseteq \forall r.B) \in m(\mathcal{T}, \mathcal{A})} f_e(\forall r.B) \\ & \cup \bigcup_{(e, C \sqsubseteq B) \in m(\mathcal{T}, \mathcal{A})} f_e(B) \cup \bigcup_{((e, e'), r \sqsubseteq r') \in m(\mathcal{T}, \mathcal{A})} r'(e, e'), \end{aligned}$$

where $B \in N_C$, $\{r, r'\} \subseteq \overline{N}_R$ and C an $\mathcal{ELHI}$ concept.

With this in place, we can now define the good successor configuration. Because we want to do this in a data-aware way, the definition now takes the shape of a least fixed point computation.

Definition 5.6. Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$. Let $\mathcal{A}_1$ be the unique (up to isomorphism) set of atoms such that $\mathcal{A}(\frac{\mathcal{T}, N_I}{\rightarrow})^\omega \circ \xrightarrow{core} \mathcal{A}_1$. Let F_i be the following set of types, for each $i \geq 1$

$$\begin{aligned} F_i := \{t \in \mathcal{F} \mid & n \in N_B, c \in N_I, r \in \overline{N}_R, \pi_1(t) = \{A \in N_C \mid A(n) \in \mathcal{A}_i\}, \\ & \pi_2(t) = \{r \in \overline{N}_R \mid r(n, c) \in \mathcal{A}_i\}, \\ & \pi_3(t) = \{A \in N_C \mid A(c) \in \mathcal{A}_i\}\}. \end{aligned}$$

Construct the pre-good successor relation, $psucc_i \subseteq \mathcal{F} \times \mathcal{P}(\mathcal{S})$, initially set to $(t, \{\}) \in psucc_i$ for all $t \in F_i$, and the loop computation relation, $L \subseteq \mathcal{F} \times N_C$, initially empty, for each $i \geq 1$ by applying the following rules exhaustively. Here, we use the notation $\overline{A}, \overline{u}, \overline{u}'$ as a shorthand for $\{A_1, \dots, A_n\}, \{u_1, \dots, u_m\}$, resp. $\{u'_1, \dots, u'_{m'}\}$.

1. If $\{(t, A_1), \dots, (t, A_n)\} \subseteq L$ and $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B \in \mathcal{T}$, then update L by adding (t, B) ;
2. If $(t, \overline{u}) \in psucc_i$ and $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $\overline{A} \sqsubseteq \pi_2(u_j)$, but $B \not\sqsubseteq \pi_2(u_j)$, then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \{u_1, \dots, (\pi_1(u_j), \pi_2(u_j) \cup \{B\}), \dots, u_m\})$;
3. If $(t, \overline{u}) \in psucc_i$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $r \in \pi_1(u_j)$ and $A \in \pi_2(u_j)$, but $B \not\sqsubseteq \pi_1(t)$, then update L by adding (t, B) ;
4. If $(t, \overline{u}) \in psucc_i$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $r^- \in \pi_1(u_j)$ and $A \in \pi_1(t) \cup \{A' \mid (t, A') \in L\}$, but $B \not\sqsubseteq \pi_2(u_j)$, then update the entry $(t, \overline{u})$ in $psucc_i$ to $(t, \{u_1, \dots, (\pi_1(u_j), \pi_2(u_j) \cup \{B\}), \dots, u_m\})$;

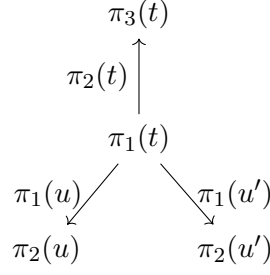


Figure 1: Structure of the good successor configuration

5. If $(t, \bar{u}) \in psucc_i$ and $r \sqsubseteq r' \in \mathcal{T}$ such that there exists $1 \leq j \leq m$, with $r \in \pi_1(u_j)$, but $r' \notin \pi_1(u_j)$, or $r^- \in \pi_1(u_j)$, but $r'^- \notin \pi_1(u_j)$ then update the entry $(t, \bar{u})$ in $psucc_i$ to $(t, \{u_1, \dots, (\pi_1(u_j) \cup \{r'\}, \pi_2(u_j)), \dots, u_m\})$;
6. If $\{(t, \bar{u}), (t', \bar{u}')\} \subseteq psucc_i$ such that there exists $1 \leq j \leq m$, with $\pi_1(t) = \pi_3(t')$, $\pi_1(u_j)^- = \pi_2(t')$ and $\pi_2(u_j) = \pi_1(t')$, then update $psucc_i$ to $(t, \{u_1, \dots, (\pi_1(u_j), \pi_2(u_j) \cup \{A' \mid (t', A') \in L\}), \dots, u_m\})$;
7. If $(t, \bar{u}) \in psucc_i$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$ such that $A \in \pi_1(t)$, and $r \notin \pi_2(t)$ or $A \notin \pi_3(t)$, and for all $1 \leq j \leq m$, $r \notin \pi_1(u_j)$ or $B \notin \pi_2(u_j)$, then update the entry $(t, \bar{u})$ in $psucc_i$ to $(t, \bar{u} \cup \{(\{r\}, \{B\})\})$;
8. If $(t, \bar{u}) \in psucc_i$ and there exists $1 \leq j, k \leq l$ such that $j \neq k$, $\pi_1(u_j) \subseteq \pi_1(u_k)$ and $\pi_2(u_j) \subseteq \pi_2(u_k)$, then update the entry $(t, \bar{u})$ in $psucc_i$ to $(t, \bar{u} \setminus \{u_j\})$;
9. If $(t, \bar{u}) \in psucc_i$, then for each $1 \leq j \leq l$, such that $((inv(u_j), \pi_1(t)), \bar{u}') \notin psucc_i$, for any $\bar{u}'$, add $((inv(u_j), \pi_1(t), \{\})$ to $psucc_i$.

Now, let $\mathcal{B}_i$ be the smallest set of atoms such that $\mathcal{A}_i \subseteq \mathcal{B}_i$ and, if $(t, B) \in L$ and there exists $e \in N_B$, $a \in N_I$ such that $\pi_1(t) \subseteq \{A \in N_C \mid A(e) \in \mathcal{B}_i\}$, $\pi_2(t) \subseteq \{r \in \overline{N}_R \mid r(e, a) \in \mathcal{B}_i\}$ and $\pi_3(t) \subseteq \{A \in N_C \mid A(a) \in \mathcal{B}_i\}$, then $B(e) \in \mathcal{B}_i$. Define $\mathcal{A}_{i+1}$ as the unique (up to isomorphism) set of atoms such that $\mathcal{B}_i(\xrightarrow{\mathcal{T}, N_I})^\omega \circ \xrightarrow{core} \mathcal{A}_{i+1}$.

Proposition 5.7. *There exists an $n \geq 1$ such that $\mathcal{A}_n = \mathcal{A}_{n+1}$. We use the notation $\mathcal{A}_{\mathcal{T}}^+$ for $\mathcal{A}_n$.*

First, note that by construction, $\mathcal{A}_i \subseteq \mathcal{A}_{i+1}$. Furthermore, the amount of blank nodes neighbouring a node in the data, such that they do not subsume each other, is restricted by the amount of existential axioms in $\mathcal{T}$.

We can now define the good successor configuration as the limit of $psucc_n$, and the set $\mathcal{F}_{\mathcal{T}, \mathcal{A}}$ of live types as the types occurring in it.

Definition 5.8. Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$. Let $n \geq 1$ be the smallest n such that $\mathcal{A}_n = \mathcal{A}_{n+1}$. Set $\text{succ}_{\mathcal{T}} = \text{psucc}_n$ and let $\mathcal{F}_{\mathcal{T}, \mathcal{A}} \subseteq \mathcal{F}$ be such that $t \in \mathcal{F}_{\mathcal{T}, \mathcal{A}}$ iff there exists $\bar{u}$ such that $(t, \bar{u}) \in \text{succ}_{\mathcal{T}}$.

In the good successor configuration, we only add successors that are directly required by the axioms and whose set of roles can be summarised by one role that implies all others. We denote by $\text{cl}_{\mathcal{T}}(r) \subseteq \bar{N}_R$ the smallest set such that $r \in \text{cl}_{\mathcal{T}}(r)$ and if $r' \in \text{cl}_{\mathcal{T}}(r)$ and $r' \sqsubseteq r'' \in \mathcal{T}$ or $r'^- \sqsubseteq r''^- \in \mathcal{T}$, then $r'' \in \text{cl}_{\mathcal{T}}(r)$.

Proposition 5.9. Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$. If $\text{succ}_{\mathcal{T}}(t, \bar{u})$, then for each $u \in \bar{u}$ there exists a $A \sqsubseteq \exists r.B \in \mathcal{T}$ such that $A \in \pi_1(t)$, $\text{cl}_{\mathcal{T}}(r) = \pi_1(u)$ and $B \in \pi_2(u)$.

When computing the good successor configuration, we are computing all loops, that is, all implications that may propagate back during the chase. In a nutshell, using the good successors is analogous to doing the chase with the closure of $\mathcal{T}$ under semantic consequences $\mathcal{T} \models C \sqsubseteq D$.

Proposition 5.10. Let $\mathcal{T}^* := \{C \sqsubseteq D \mid \mathcal{T} \models C \sqsubseteq D\}$, and let $\mathcal{B}$ be the unique (up to isomorphism) set of atoms such that $\mathcal{A}(\xrightarrow{\mathcal{T}^*, N_I})^{\omega} \circ \xrightarrow{\text{core}} \mathcal{B}$, then $\mathcal{A}_{\mathcal{T}}^+ \cong \mathcal{B}$.

The proof relies on the fact that all relevant loop computations are covered in the construction of the good successor configuration. As we consider live types, all possible configurations in the anonymous parts are considered and all inferred loops are saved in the loop computation list.

5.3 Core Universal Model for $\mathcal{ELHI}$

The good successor configuration allows us to easily build the desired core universal model.

We introduce some additional notation. Let $\mathcal{A}$ be a set of atoms. For $X \subseteq N_I \cup N_B$, we define the set $n_{\mathcal{A}}(X)$ of neighbours of X . We also define the type $\text{type}_{\mathcal{A}}(e)$ of a blank node $e \in N_B$ with $|n_{\mathcal{A}}(e)| = 1$, where $e' \in n_{\mathcal{A}}(e')$.

$$\begin{aligned} n_{\mathcal{A}}(X) &:= \bigcup_{e \in X} \{x \in N_I \cup N_B \mid r(e, x) \in \mathcal{A} \vee r(x, e) \in \mathcal{A}, r \in N_R\} \\ \text{type}_{\mathcal{A}}(e) &:= (\{A \in N_C \mid A(e) \in \mathcal{A}\}, \{r \in \bar{N}_R \mid r(e, e') \in \mathcal{A}\}, \{A \in N_C \mid A(e') \in \mathcal{A}\}) \end{aligned}$$

We can now build the model, starting from $\mathcal{A}_{\mathcal{T}}^+$ and adding successors according to $\text{succ}_{\mathcal{T}}$.

Definition 5.11. Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$, define the core universal model $\text{core}(\mathcal{T}, \mathcal{A})$ as the union of the sequence $X_1, X_2, \dots$, that is recursively defined in the following way

- $X_1 := \mathcal{A}_{\mathcal{T}}^+$;

$$\begin{aligned} [\neg a]_S^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus \{a^{\mathcal{I}}\} \\ [\neg A]_S^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \end{aligned}$$

Figure 5.1: Evaluation of semi-positive SHACL shape expressions.

- X_{n+1} is constructed from X_n by adding, for each $e \in N_B$ such that $|n_{X_n}(e)| = 1$ and $\text{succ}_{\mathcal{T}}(\text{type}_{X_n}(e), \bar{u})$ holds for some $\bar{u}$, the following atoms for each $u \in \bar{u}$, with a fresh $e' \in N_B$ for each $u \in \bar{u}$
 - $\{r(e, e'), r^-(e', e)\} \subseteq X_{n+1}$ for each $r \in \pi_1(u)$;
 - $A(e') \in X_{n+1}$ for each $A \in \pi_2(u)$.

Note that we can see each entry in the sequence $X_1, X_2, \dots$ as layers such that the freshly added blank nodes in X_i have distance i to the ABox.

It follows directly from the results presented in the previous chapter that $\text{can}(\mathcal{T}, \mathcal{A})$ is a model, a core and universal. As do all the results regarding coinciding with the results of the core and stable chase, when they terminate.

5.4 Semi-Positive SHACL Validation

We now return the focus to the problem of SHACL validation in the presence of $\mathcal{ELHI}$ ontologies. However, here, we will focus on *semi-positive* SHACL. This extends the rewriting for positive SHACL as presented in Section 4.4 by allowing light forms of negation, without needing to take complements as discussed in Section 4.5. That is, in *semi-positive* SHACL let shape expressions be given by

$$\varphi ::= a \mid \neg a \mid A \mid \neg A \mid \varphi \wedge \varphi \mid \exists E.\varphi \mid s,$$

for $a \in N_I$, $A \in N_C$, $s \in N_S$ and E a regular path expression. As $\neg a$ and $\neg A$ are not part of the syntax of SHACL as presented in Chapter 3, we extend the there presented semantics in Figure 5.1.

In a similar way as proven in Proposition 4.29, we find that each constraint set can be rewritten into a constraint set in the following normal form. Note here that $s \leftarrow s'$ is equivalent to $s \leftarrow s' \wedge s'$. That is, we say a semi-positive SHACL constraint is in normal form if all constraints are of one of the following forms

$$\begin{array}{lll} \text{(NC'1)} \ s \leftarrow a & \text{(NC'2)} \ s \leftarrow \neg a & \text{(NC'3)} \ s \leftarrow A \\ \text{(NC'4)} \ s \leftarrow \neg A & \text{(NC'5)} \ s \leftarrow s' \wedge s'' & \text{(NC'6)} \ s \leftarrow \exists r.s', \end{array}$$

where $a \in N_I$, $r \in \overline{N}_R$, $\{s, s', s''\} \subseteq N_S$ and $A \in N_C$. Note that $s \leftarrow \exists(\prod R).s'$ was mainly needed in the previous normal form to express Horn- $\mathcal{ALCHIQ}$ concepts in SHACL, and is not needed to capture $\mathcal{ELHI}$.

Let us revisit reducing SHACL validation in presence of ontologies to validation that no longer needs to use the TBox. Given an $\mathcal{ELHI}$ KB $(\mathcal{T}, \mathcal{A})$, or more specifically, the $\text{succ}_{\mathcal{T}}$ that represents a core universal model of $(\mathcal{T}, \mathcal{A})$, we transform a SHACL document $(\mathcal{C}, \mathcal{G})$ into an updated SHACL document $(\mathcal{C}_{\mathcal{T}, \mathcal{A}}, \mathcal{G})$ such that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}'$ validates $(\mathcal{C}_{\mathcal{T}, \mathcal{A}}, \mathcal{G})$, for some set of atoms $\mathcal{A}'$ that can be easily obtained from $\mathcal{A}$ and the construction of $\text{succ}_{\mathcal{T}}$. Unlike the similar rewriting for $DL\text{-}Lite_{\mathcal{R}}$, the variant we introduce here does not work for *every* set of atoms $\mathcal{A}$: we can only guarantee soundness and completeness if the set of atoms $\mathcal{A}$ is such that $\text{succ}_{\mathcal{T}}$ represents the core universal model of $(\mathcal{T}, \mathcal{A})$, meaning that all types occurring in the anonymous part of $\text{can}(\mathcal{T}, \mathcal{A})$ are live types. We compromise the data independence, but in this way, we avoid the best case exponentially of its predecessor [AOOŠ23], and the version presented in the previous chapter.

We use a fresh concept name $\hat{A} \in N_C$ to decorate the outer layer of $\mathcal{A}_{\mathcal{T}}^+$, and define $\mathcal{A}_{\mathcal{T}}^* := \mathcal{A}_{\mathcal{T}}^+ \cup \{\hat{A}(e) \mid e \in \Delta^{\mathcal{A}_{\mathcal{T}}^+} \cap N_B\}$. The elements in our rewriting are triples $(t, P, H) \subseteq \mathcal{F}_{\mathcal{T}, \mathcal{A}} \times \mathcal{P}(\{\exists r.s \mid r \in \overline{N}_R, s \in N_S\}) \times \mathcal{P}(N_S)$: t is a live type, P a set of basic shape assumptions (note we no longer need the negative assumptions captured in Q in the previous chapter, as shape names can no longer be referred to in a negative way) and H a set of shape names. The starting set I of our rewriting is such that $(t, P, H) \in I$ iff

- if $\exists r.s' \in P$, there exists $s \leftarrow \exists r.s' \in \mathcal{C}$ and $r \in \pi_2(t)$, and
- $s \in H$ iff there exists $\exists r.s' \in P$ such that $s \leftarrow \exists r.s' \in \mathcal{C}$.

Our rewriting algorithm adds shape names s to the heads H . Specifically, we update $(t, P, H) \in I$, to $(t, P, H \cup \{s\})$ when any of the following holds:

1. $s \leftarrow A \in \mathcal{C}$, $A \in \pi_1(t)$, $s \notin H$, or
2. $s \leftarrow \neg A \in \mathcal{C}$, $A \notin \pi_1(t)$, $s \notin H$, or
3. $s \leftarrow \neg a \in \mathcal{C}$, $s \notin H$, or
4. $s \leftarrow s_1 \wedge s_2 \in \mathcal{C}$, $\{s_1, s_2\} \subseteq H$, $s \notin H$, or
5. $s \leftarrow \exists r.s_1 \in \mathcal{C}$, and there exists $(t', P', H') \in I$ such that, for some $u \in \bar{u}$ with $\text{succ}_{\mathcal{T}}(t, \bar{u})$, we have $t' = (\text{inv}(u), \pi_1(t))$, $s_1 \in H'$, $r^- \in \pi_2(t)$ and $\{s \mid \exists r.s \in P'\} \subseteq H$.

Let $\text{sat}(I)$ be the result of exhaustively adding shape names as above. The triples in $\text{sat}(I)$ are then translated into the set of constraints $\mathcal{C}_{\mathcal{T}, \mathcal{A}}$ in the following way.

Let $\mathcal{C}_{\mathcal{T}, \mathcal{A}}$ be the smallest set of constraints such that $\mathcal{C} \subseteq \mathcal{C}_{\mathcal{T}, \mathcal{A}}$, and such that if $(t, P, H) \in \text{sat}(I)$, then for each $s \in H$ the following constraint is contained in $\mathcal{C}_{\mathcal{T}, \mathcal{A}}$

$$s \leftarrow \bigwedge_{A \in \pi_1(t)} A \wedge \bigwedge_{A \in N_C^{\mathcal{T}} \setminus \pi_1(t)} \neg A \wedge \bigwedge_{\exists r.s' \in P} \exists r.s' \wedge \hat{A},$$

where $r \in \pi_2(t)$ is such that for all $r' \in \pi_2(t)$ we have $r' \in cl_{\mathcal{T}}(r)$, and $N_C^{\mathcal{T}}$ only contains all concept names appearing in $\mathcal{T}$. Note that the existence of r is guaranteed by Proposition 5.9.

This gives us the desired rewriting. Its proof of correctness is analogous to the one presented in Chapter 4.

Theorem 5.12. *Given an $\mathcal{ELHI}$ knowledge base $(\mathcal{T}, \mathcal{A})$ and $(\mathcal{C}, \mathcal{G})$ a semi-positive SHACL SHACL document. Then it holds that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}_{\mathcal{T}}^*$ validates $(\mathcal{C}_{\mathcal{T}, \mathcal{A}}, \mathcal{G})$.*

Proof Sketch. First of all, note how the above rules relate to the rewriting rules of Definition 4.32: the starting set I takes care of rules 1 and 2; rules 1, 2 and 3 of this rewriting are captured by rule 3; and the last two rules of this rewriting, are the simplified versions of the last two rules of Definition 4.32. Note that the main idea is to evaluate the rewritten shapes on the nodes e such that $|e| = 2$ in $can_2(\mathcal{T}, \mathcal{A})$, where we annotated exactly those nodes e with the fresh concept $\hat{A}$. This also means we know that each of these nodes has exactly one neighbour, namely the nominal a such that $ar^- = e$. Moreover, this means we do not need to consider constraints of the form $s \leftarrow a$. Finally, the proofs of Propositions 4.35 and 4.37 can be mirrored, such that only the case $e \in N_{\mathcal{K}} \setminus N_I$ is relevant. For the induction steps, note that if $\{(t, P, H), (t, P', H')\} \subseteq I$, then it is also possible to find $(t, P \cup P', H'') \in I$ such that $H \cup H' \subseteq H''$.

Since we are taking the input $\mathcal{A}$ into account, we do not have to consider all possible types and $\mathcal{C}_{\mathcal{T}}$ is not best-case exponential. However, there are still ways to optimise the rewriting. The first and simplest optimisation is to restrict the form of the starting sets even more, that is, not all $s \in N_S$ will ever interact with each other. So, let $rel(s) \subseteq N_S$ be defined as the smallest set such that $s \in rel(s)$; if $s' \leftarrow s'' \wedge s''' \in \mathcal{C}$ and $\{s', s'', s'''\} \cap rel(s) \neq \{\}$, then $\{s', s'', s'''\} \in rel(s)$; and if $s' \leftarrow \exists r. s'' \in \mathcal{C}$ and $\{s', s''\} \cap rel(s) \neq \{\}$, then $\{s', s''\} \in rel(s)$. The restriction is then that (t, P, H) is only considered to be in the starting set in case there exists $s \in N_S$ such that $H \subseteq rel(s)$. This optimisation is mostly useful for sets of constraints that relatively use a small amount of role names. Another more involved optimisation is based on the close correspondence between the SHACL fragment we are considering and $\mathcal{ELHI}$ in normalised form. We can translate the SHACL constraints into $\mathcal{ELHI}$ axioms, introducing for each existing concept name a fresh concept name to denote the absence of this concept name in $succ_{\mathcal{T}}$. A similar algorithm to the one in Section 5.2 will then decorate the good successor configuration with all relevant shape names. We believe that the latter version may be better suited for implementation.

However, all these optimisations cannot avoid EXPTIME-complexity in the worst case. That is, from the results in the previous chapter we can easily derive the following.

Theorem 5.13. *In the presence of $\mathcal{ELHI}$ TBoxes, semi-positive SHACL validation is EXPTIME-complete in combined complexity and PTIME-complete in data complexity.*

5.5 Conclusion

In this chapter, we revisited a more efficient representation of the core universal model, for $DL\text{-}Lite_{\mathcal{R}}$ and $\mathcal{ELHI}$. For both this is possible in a data-independent way, however, for $\mathcal{ELHI}$,

we showed how considering a set of live types can decrease the amount of live types. That is, by considering the exact configurations that appear in the data, we have chosen to focus solely on computing the relevant parts, highly reducing the number of good successor configurations and TBox subsumptions that need to be computed. This makes this a first step towards an efficient implementation of the core chase.

We further focus on one of many application areas of this technique: we use the alternative representation of a core universal model to reduce SHACL validation combined with $\mathcal{ELHI}$ TBoxes to plain SHACL validation. Also here, we could take all types for a data-independent rewriting, as described in Chapter 4, but restricting it to the live types allows us to avoid an exponential computation in many cases, as not all combinations of types and assumptions need to be considered. This means the proposed algorithm also brings us closer to implementing the combination of SHACL with ontologies. We are confident that the proposed techniques can most likely be optimised further, but leave this for future work. To keep the presentation simple we focused on semi-positive SHACL in this chapter, but we note that the rewriting presented in the last section can be extended to SHACL with stratified negation in the same way as demonstrated in the previous chapter.

5.6 Related Work

In the previous chapter, we considered Horn- $\mathcal{ALCHIQ}$, the there presented related work remains relevant for this part. However, here, we looked at two simpler logics. For an overview of which DL underlies the OWL 3 profiles, we refer the reader to the work by Krotzsch [Krö12]. In particular, it is pointed out here that $DL-Lite_{\mathcal{R}}$ is the logic underlying OWL 2 QL, whereas $\mathcal{EL}^{++}$ has this same status for OWL 2 EL.

Recently, Oudshoorn et al. [OGA26] implemented a pure rewriting of SHACL and OWL into plain SHACL for $\mathcal{EL}^-$, a fragment of $\mathcal{EL}^{++}$ [BLB08] that restricts the usage of existential restrictions on the right-hand side of axioms.

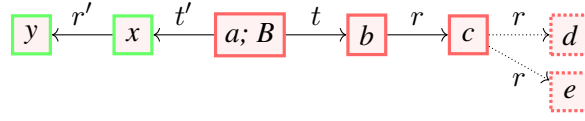
Another approach of SHACL validation in presence of ontologies is presented by Ke et al. [KZA24]: it carefully considers which reasoning results are relevant for validation. That is, they carefully consider which facts need to be materialised, and combine this with a light rewriting, for OWL LD [GHKP12]. Unlike the rewriting presented in the previous chapter, the rewriting by Ke et al. remains twofold for each data graph in the sense that both their system RE-SHACL and a standard SHACL validator need to be run. The strength of their approach is clearly the working implementation and evaluation, but for a different DL.

Counting Over Regular Expressions

In this chapter, we take an orthogonal approach towards combining SHACL and ontologies: instead of focussing on SHACL with negation, we now shift our intention towards counting over regular expressions. In the previous chapters, we noted that a *certain answer semantics*, providing an answer that holds in all models, does not provide meaningful answers if SHACL constraints contain negation. However, when counting the amount of nodes reachable, the certain answer semantics in the shape of guaranteeing that each model has a certain amount of witnesses, becomes meaningful again. That is, in this chapter, we let go of the semantics of validating SHACL over the universal core model, and focus on guarantees for all models, as is common in (positive) query answering settings.

That is, given a regular expression E and a name of an individual a in the data, can we determine the highest number n such that all models of the given knowledge base have at least n witnesses? Where witnesses are the nodes reachable by E from a . Determining which model of a given knowledge base provides the minimal number of reachable nodes turns out to be far from straightforward, although starting with a minimal universal model is helpful: as it is universal, there exists a homomorphism from this minimal model into a model witnessing the minimal number of nodes reachable, and as it is minimal, we avoid making unnecessary copies of reachable elements. To illustrate this, consider the following.

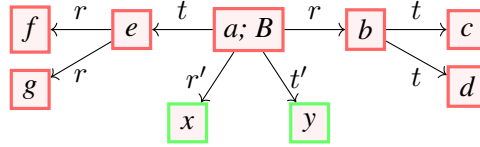
Example 6.1. Let $\mathcal{T} = \{B \sqsubseteq \exists t'. \top, \exists t'^-. \top \sqsubseteq \exists r'. \top\}$ and set $\mathcal{A} = \{B(a), t(a, b), r(b, c)\}$ and $\mathcal{A}' = \mathcal{A} \cup \{r(c, d), r(c, e)\}$. We want to determine the least number of nodes reachable from the node a by the regular expression $(tr)^*(t'r'r)^*(t'r')^*$ over all models of $(\mathcal{T}, \mathcal{A})$ and of $(\mathcal{T}, \mathcal{A}')$. Possible (and minimal) universal models can be constructed by adding atoms $t'(a, x)$ and $r'(x, y)$ to the data; here the addition of the dotted roles and nodes indicates the minimal universal model for $(\mathcal{T}, \mathcal{A}')$.



In both models, there are three reachable nodes: a , c and y . If we merge c and y (in the sense of creating a diamond), note that the result would still be a model, for both $\mathcal{A}$ and $\mathcal{A}'$. For $\mathcal{A}$, this would decrease the number of reachable nodes to 2: only a and c/y . However, if we perform the same merge in the minimal universal model of $(\mathcal{T}, \mathcal{A}')$, this increases the number of reachable nodes from a to four: a , c/y , d and e . That is, to decide whether this merge is reasonable, information in another part of the data graph is relevant, namely the number of nodes reachable by $r(t'r'r)^*$ from c . It turns out that two reachable nodes is the minimum for $(\mathcal{T}, \mathcal{A})$, and three for $(\mathcal{T}, \mathcal{A}')$. ♡

In the above example, the minimal universal model may or may not provide the final answer. In particular, the minimal universal model does not have the property of being the model with the smallest number of answers, for instance when it contains infinite repetitive structures. The complexity grows further as the possibilities for merging nodes increase, leaving little hope for a uniform approach.

Example 6.2. We want to determine the reachable nodes from a by $(r \cup r')^+ \cup (t \cup t')^+$. Let $\mathcal{T} = \{B \sqsubseteq \exists r'. \top, B \sqsubseteq \exists t'. \top\}$ and $\mathcal{A} = \{B(a), r(a, b), t(b, c), t(b, d), t(a, e), r(e, f), r(e, g)\}$. The minimal universal model can be constructed by adding the atoms $r'(a, x)$ and $t'(a, y)$.



In this model there are four reachable nodes: b , e , x and y . This can be reduced to three by merging x and y , after which merging x/y with either b or e brings the number back to four. If instead of merging x and y we decided to directly merge x with e and y with b we could have achieved only two reachable nodes; the best we can do for $(\mathcal{T}, \mathcal{A})$. ♡

This culminates in counting regular expressions over ontologies being rather hard; only very mild expressivity is required to lift the complexity of determining the certain answer semantics for counting queries to coNP-hardness in data complexity. As validation of plain, recursive, SHACL can be performed in polynomial time in data complexity in the well-founded semantics [OŠ24], and for stratified constraints in the stable model semantics [ACO⁺20], and moreover, it has been shown that for meaningful fragments of SHACL, determining validation in presence of ontologies remains decidable in polynomial time in data complexity (Theorem 4.56), this chapter explores which aspects of the counting queries problem in SHACL can be solved in polynomial time.

This chapter is organised as follows. The problem of counting the number of nodes reachable via a regular path expression is defined in Section 6.1. In Section 6.2 we provide a series of coNP-hardness results. This is followed by the identification of a tractable fragment in Section 6.3. The algorithm itself is presented in Section 6.4, and Section 6.5 extends our tools to show that we can reason about the number of reachable nodes without materialising any model.

6.1 Problem Statement

Motivated by SHACL cardinality constraints, we are interested in computing the range of possible nodes for a regular path expression and a starting node a . Following Kostylev and Reutter [KR15], we give the semantics of counting queries as a set of cardinalities.

Recall that for each language defined by a regular path expression, there exists a non-deterministic automaton (NFA) that recognises the exact same language, and the other way around. That is, we can use a regular expression E and an automaton $\mathcal{M}$ interchangeably in the following definition, as long as they recognise the exact same language.

Definition 6.3. *Let $\mathcal{K}$ be a KB. A counting query for $\mathcal{K}$ is a pair (a, E) , where $a \in N_I$ and E is a regular path expression or NFA. A number k belongs to the set of cardinality certain answers of (a, E) over $\mathcal{K}$, we write $k \in \text{certCard}(a, E, \mathcal{K})$, if there is a model $\mathcal{I}$ of $\mathcal{K}$ such that the set $\{e \mid (a, e) \in E^{\mathcal{I}}\}$ has exactly k elements.*

An observation, and note that this is about the size of the set of answers, is that for $DL\text{-}Lite_{\mathcal{R}}$ ontologies the cardinality of the set $\text{certCard}(a, E, \mathcal{K})$ is always 0, 1 or infinity [KR15]: if there is a model $\mathcal{I}$ that produces at least one pair (a, b) in $E^{\mathcal{I}}$ with $a \neq b$, we can always add fresh paths and elements to create a sequence $\mathcal{I}_1, \mathcal{I}_2, \dots$ of models in which there is one additional pair, or two additional pairs, and so on. For this reason, we only focus on understanding what is the *minimum* value in $\text{certCard}(a, E, \mathcal{K})$. Further, since the motivation for this framework is integrating schema constraint into ontological reasoning tasks for graph databases, in which graphs tend to be orders of magnitude larger than schemas and ontologies, we study the data complexity of the problem, in which we consider the TBox and the query to be fixed. More precisely, we study the following problem. We note that it does not matter how k is encoded.

MINCOUNT

- Given: TBox $\mathcal{T}$, counting query (a, E)
- Input: ABox $\mathcal{A}$, $k \in \mathbb{N}_{>0}$, in binary
- Decide: $k \leq \min\{\text{certCard}(a, E, (\mathcal{T}, \mathcal{A}))\}$

6.2 MINCOUNT is Hard

For the case of $DL\text{-}Lite_{\mathcal{R}}$ ontologies, we show that the coNP-hard bound stands whenever one allows inverses, either when the query uses inverse roles p^- for $p \in N_R$, or in ontologies, using

role inclusions of the form $p \sqsubseteq p'^{-}$ for $\{p, p'\} \subseteq N_R$. Moreover, we show that allowing slightly more expressive DL constructs also lead to coNP-hardness, even if queries do not use inverses. We start with the latter, as the proof allows us to introduce the reductions we use in the remainder of the section.

Proposition 6.4. *MINCOUNT is coNP-hard in data complexity for every DL containing axioms of the form $A \sqsubseteq \exists p.\top$, $\exists p.A \sqsubseteq A'$ and $A \sqsubseteq \neg A'$, where $\{A, A'\} \subseteq N_C$, $p \in N_R$ and regular path expressions that are words over N_R .*

Proof. We prove this via a reduction from non-3-colourability to MINCOUNT. Let the regular expression be

$$E = aux \cdot hasCol.$$

The set of axioms considered is the following:

$$\begin{aligned} \mathcal{T} = \{ & V \sqsubseteq \exists hasCol.\top, & \exists edge.Green \sqsubseteq AGreen, \\ & Red \sqsubseteq \neg ARed, & \exists edge.Blue \sqsubseteq ABlue \\ & Green \sqsubseteq \neg AGreen, & \exists hasCol.Red \sqsubseteq Red \\ & Blue \sqsubseteq \neg ABlue, & \exists hasCol.Green \sqsubseteq Green \\ & \exists edge.Red \sqsubseteq ARed, & \exists hasCol.Blue \sqsubseteq Blue \}. \end{aligned}$$

The main intuition is that nodes labelled with a colour are assigned that colour, whereas the anti-colour, the name of the colour preceded by an ‘A’, denotes the colours excluded for this node, as they are already assigned to a neighbour. Given an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let the ABox $\mathcal{A}$ consist of the atoms:

$$\begin{aligned} \mathcal{A} := \{ & V(e) \mid e \in \mathcal{V} \} \cup \{ Red(r), Green(g), Blue(b) \} \cup \{ edge(e, e') \mid \{e, e'\} \in \mathcal{E} \} \cup \\ & \{ aux(a, e) \mid e \in \mathcal{V} \cup \{s\} \} \cup \{ hasCol(s, r), hasCol(s, g), hasCol(s, b) \}. \end{aligned}$$

Now assume $3 \geq \min\{certCard(a, E, (\mathcal{T}, \mathcal{A}))\}$. First note that three nodes are reachable from a by E already in $\mathcal{A}$, namely r, g and b . Moreover, in a model, each node e of the graph needs to have an outgoing edge labelled with $hasCol$. As all such e are reachable by aux from a , the $hasCol$ edge needs to point to r, g or b to not increase the number of reachable nodes to 4 directly. Furthermore, note that two nodes connected by an $edge$ -edge cannot point to the same r, g or b node, as that would label them with both the colour and anti-colour at the same time, which is disallowed in a model. Thus, if there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ such that the amount of reachable nodes is 3, a 3-colouring for the original graph corresponds to the assignment of colours by $hasCol$ as in the model $\mathcal{I}$.

For the other direction assume there exists a 3-colouring. In this case, it is immediate there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$ such that 3 nodes are reachable from a by E : simply send the $hasCol$ edge of each node to the colour it is assigned in the 3-colouring. Thus, $\mathcal{G}$ is not 3-colourable iff $4 \leq \min\{certCard(a, E, (\mathcal{T}, \mathcal{A}))\}$, as required. $\clubsuit$

Next we continue with coNP-hardness for $DL\text{-}Lite_{\mathcal{R}}$ ontologies in which either the query or the ontology features inverse roles.

Theorem 6.5. *MINCOUNT is coNP-hard in data complexity even when restricted to the following cases, for $A \in N_C$, $\{p, p'\} \subseteq N_R$:*

- Any DL allowing concept inclusions of the form $A \sqsubseteq \exists p.\top$ and regular path expressions that are words over $\bar{N}_R$.
- Any DL allowing both concept inclusions $A \sqsubseteq \exists p.\top$ and role inclusions $p \sqsubseteq p'^{-}$, and regular path expressions that are words over N_R .
- Any DL allowing both concept inclusions $A \sqsubseteq \exists p^{-}.\top$ and role inclusions $p \sqsubseteq p'^{-}$, and regular path expressions that are words over N_R .

Since $p \sqsubseteq p'^{-}$ if and only if $p^{-} \sqsubseteq p'$, this result suggests that the only hope for an efficient algorithm is to prohibit any form of inverses both in queries and in role inclusions. Indeed, in the following sections we show that settings without inverses do lead to tractable algorithms. For now, we devote the remainder of this section to the proof of Theorem 6.5.

Just as with Proposition 6.4, all three cases in Theorem 6.5 are shown via reductions from the non-3-colourability problem. We recall the intuition behind the proof of the previous proposition and extend it a little in the following reduction. In the next proof, the main ideas are simply pushed further. Moreover, for succinctness of presentation, we will

Suppose we want to decide whether a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is not 3-colourable. We represent $\mathcal{G}$ as an ABox $\mathcal{A}$ with one concept name V such that $V^{\mathcal{A}} := \mathcal{V}$ and one role $edge$ with $edge^{\mathcal{A}} := \{(v, v') \mid \{v, v'\} \in \mathcal{E}\}$. Then, using the inclusion $V \sqsubseteq \exists hasCol.\top$, we associate one anonymous element per each node in the graph. This element is to represent the colour of the graph. As in the proof of Proposition 6.4, let $\Delta^{\mathcal{A}} = \mathcal{V} \cup \{a, s, r, g, b\}$, $aux^{\mathcal{A}} := \{(a, v) \mid v \in \mathcal{V}\} \cup \{(a, s)\}$ and $hasCol^{\mathcal{A}} := \{(s, r), (s, g), (s, b)\}$. Consider now the expression $E = aux \cdot hasCol$. Then, any interpretation $\mathcal{I}$ that minimises the number of nodes reachable from a via E will only have pairs $(a, r), (a, b), (a, g)$ in $E^{\mathcal{I}}$. Intuitively, this means that each of the anonymous nodes associated to nodes $v \in \mathcal{V}$ have been merged to some of r, b or g . This can be used to indicate a particular colour being guessed for each node. Of course, we need to check that such a guess is indeed a proper colouring. In the proof of Proposition 6.4 we could use the TBox to take care of this, but for the case of $DL\text{-}Lite_{\mathcal{R}}$ ontologies we need a new gadget. We add to $\mathcal{A}$ three new roles, $red, green, blue$, and a new element z . The interpretations in $\mathcal{A}$ are as follows: $red^{\mathcal{A}} := \{(a, r), (r, z)\}$, $green^{\mathcal{A}} := \{(a, g), (g, z)\}$, $blue^{\mathcal{A}} := \{(a, b), (b, z)\}$. Then, for expression $RedWord = red \cdot hasCol^{-} \cdot edge \cdot hasCol \cdot red$, and an interpretation $\mathcal{I}$ satisfying $E^{\mathcal{I}} = \{(a, r), (a, b), (a, g)\}$, we have that $(a, z) \in RedWord^{\mathcal{I}}$ if and only if the colouring represented in $\mathcal{I}$ assigns the colour red to two neighbouring nodes. It is then easy to see that the minimum number of different nodes reachable from a via the expression $E + RedWord + GreenWord + BlueWord$, with $GreenWord = green \cdot hasCol^{-} \cdot edge \cdot hasCol \cdot green$ and $BlueWord = blue \cdot hasCol^{-} \cdot edge \cdot hasCol \cdot blue$ in any interpretation of our KB is 4 iff $\mathcal{G}$ is not 3-colourable.

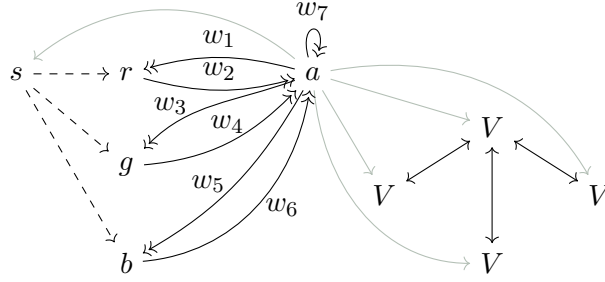


Figure 6.1: ABox construction for $\mathcal{G} = (\{1, 2, 3, 4\}, \{\{1, 2\}, \{1, 3\}, \{1, 4\}\})$. Black arrows represent the role *edge*, grey arrows *aux*, and dashed black arrows *hasCol*. Furthermore, $w_1 = \text{red}$, $w_2 = \text{red} \cdot \text{GreenWord} \cdot \text{BlueWord} \cdot \text{aux} \cdot \text{hasCol}$, $w_3 = \text{RedWord} \cdot \text{green}$, $w_4 = \text{green} \cdot \text{BlueWord} \cdot \text{aux} \cdot \text{hasCol}$, $w_5 = \text{RedWord} \cdot \text{GreenWord} \cdot \text{blue}$, $w_6 = \text{blue} \cdot \text{aux} \cdot \text{hasCol}$ and $w_7 = \text{RedWord} \cdot \text{GreenWord} \cdot \text{BlueWord}$.

The sketch above uses a regular expression involving the union of several words. In what follows, we show how to tighten it so that the path expression is just a single word.

Proposition 6.6. *MINCOUNT is coNP-hard in data complexity for every DL containing axioms of the form $A \sqsubseteq \exists r. \top$, where $A \in N_C$, $r \in N_R$ and regular expressions using only concatenation of roles and inverse roles.*

Proof. We prove this via a reduction of non-3-colourability to MINCOUNT. Let the regular expression be:

$$E = \text{RedWord} \cdot \text{GreenWord} \cdot \text{BlueWord} \cdot \text{aux} \cdot \text{hasCol},$$

where

$$\begin{aligned} \text{RedWord} &:= \text{red} \cdot \text{hasCol}^- \cdot \text{edge} \cdot \text{hasCol} \cdot \text{red} \\ \text{GreenWord} &:= \text{green} \cdot \text{hasCol}^- \cdot \text{edge} \cdot \text{hasCol} \cdot \text{green} \\ \text{BlueWord} &:= \text{blue} \cdot \text{hasCol}^- \cdot \text{edge} \cdot \text{hasCol} \cdot \text{blue}. \end{aligned}$$

Given an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, construct the ABox $\mathcal{A}$ in the following way: $\mathcal{A}$ contains the roles *edge*, *aux*, *hasCol*, *red*, *green* and *blue*, with the extensions as defined in Figure 6.2 and illustrated in Figure 6.1. Let $\mathcal{V} \subseteq N_I$, and every name not being v or v' a name for a specific node.

Furthermore, there is one concept name with a non-empty extension in $\mathcal{A}$: $V^{\mathcal{A}} := \mathcal{V}$.

Let the TBox $\mathcal{T}$ be defined as follows:

$$\mathcal{T} = \{V \sqsubseteq \exists \text{hasCol}. \top\}$$

Now, we find $\mathcal{G}$ is not 3-colourable iff $\min\{\text{certCard}(a, E, (\mathcal{T}, \mathcal{A}))\} \geq 4$. To see this, first note that $\min\{\text{certCard}(a, E, (\emptyset, \mathcal{A}))\} \geq 3$, as r , g and b are already reachable by E in $\mathcal{A}$.

$$\begin{aligned}
edge^A &:= \{(v, v') \mid \{v, v'\} \in \mathcal{E}\} \cup \{(rg_2, rg_3), \\
&\quad (rb_2, rb_3), (gr_2, gr_3), (gb_2, gb_3), (br_2, br_3), \\
&\quad (bg_2, bg_3), (er_2, er_3), (eg_2, eg_3), (eb_2, eb_3)\} \\
aux^A &:= \{(a, v) \mid v \in \mathcal{V}\} \cup \{(a, s), (ra, rh), (ga, gh), \\
&\quad (ba, bh)\} \\
hasCol^A &:= \{(s, r), (s, g), (s, b), (rh, a), (gh, a), (bh, a), \\
&\quad (rg_2, rg_1), (rg_3, rg_4), (rb_2, rb_1), (rb_3, rb_4), \\
&\quad (gr_2, gr_1), (gr_3, gr_4), (gb_2, gb_1), (gb_3, gb_4), \\
&\quad (br_2, br_1), (br_3, br_4), (bg_2, bg_1), (bg_3, bg_4), \\
&\quad (er_2, er_1), (er_3, er_4), (eg_2, eg_1), (eg_3, eg_4), \\
&\quad (eb_2, eb_1), (eb_3, eb_4)\} \\
red^A &:= \{(a, r), (r, rg_0), (a, gr_1), (gr_4, gr_5), (a, br_1), \\
&\quad (br_4, bg_0), (a, er_1), (er_4, eg_0)\} \\
green^A &:= \{(rg_0, rg_1), (rg_4, rb_0), (gr_5, g), (g, gb_0), \\
&\quad (bg_0, bg_1), (bg_4, bg_5), (eg_0, eg_1), (eg_4, eb_0)\}, \\
blue^A &:= \{(rb_0, rb_1), (rb_4, ra), (gb_0, gb_1), (gb_4, ga), \\
&\quad (bg_5, b), (b, ba), (eb_0, eb_1), (eb_4, a)\}
\end{aligned}$$

Figure 6.2: ABox construction: extension of roles.

Furthermore, note that every vertex $v \in V$ needs to have an outgoing *hasCol* edge to be a model, and that if this edge reaches a node different from r, g or b , at least four nodes are reachable by E from a . Finally, if for some model $\mathcal{I}$, we find $\{(v, x), (v', x)\} \subseteq hasCol$ and $(v, v') \in edge$, for $x \in \{r, g, b\}$ and $\{v, v'\} \subseteq V$, then a is reachable by E from a , ensuring at least four reachable nodes as well. ♣

Proposition 6.6 gives us the first part of Theorem 6.5. For the second part, all we need to do is to modify the TBox $\mathcal{T}$, adding an additional role inclusion: $hasCol \sqsubseteq hasColinv^-$, and modify the expressions *RedWord*, *GreenWord*, *BlueWord* used in the proof, as follows:

$$\begin{aligned}
RedWord &:= red \cdot hasColinv \cdot edge \cdot hasCol \cdot red \\
GreenWord &:= green \cdot hasColinv \cdot edge \cdot hasCol \cdot green \\
BlueWord &:= blue \cdot hasColinv \cdot edge \cdot hasCol \cdot blue.
\end{aligned}$$

For the third part, we again modify $\mathcal{T}$ so that the concept inclusion mentions *hasColinv*:

$$\mathcal{T} = \{V \sqsubseteq \exists hasColinv^-. \top, hasColinv \sqsubseteq hasCol^-\},$$

and use the modified expressions as explained for the second bullet point. This concludes the proof of Theorem 6.5.

Next, we find that in $DL\text{-}Lite_{\mathcal{R}}$ without inverse roles nor any role hierarchies in the ontology, we can still obtain coNP-hardness. It turns out that the problem is also hard in presence of role disjointness. That is, if there are no basic role hierarchies involved and at least one role disjointness axiom, we can provide a similar reduction.

Theorem 6.7. *MINCOUNT is coNP-hard in data complexity even when restricted to $DL\text{-}Lite_{\mathcal{R}}$ without any axioms of the form $p \sqsubseteq p'$, for $\{p, p'\} \subseteq \overline{N}_R$ and regular expressions over N_R .*

Proof. We, again, prove this via a reduction of non-3-colourability to MINCOUNT. Let the regular expression considered be

$$aux + aux_2 \cdot hasCol + aux_2 \cdot hasCol \cdot edge \cdot t.$$

In this case, we will find that a given input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is not 3-colourable iff $3n + 1 \leq \text{certCard}(a, E, (\mathcal{T}, \mathcal{A}))$, where $n = |\mathcal{V}|$. In this reduction, there are three colour nodes, rv, bv and gv for each vertex v , and all of these will be reachable by default (via aux), bringing the total number of reachable nodes to at least $3n$ in every model. The last part, $aux_2 \cdot hasCol \cdot edge \cdot t$, will ensure correctness of the 3-colouring in the sense that it will reach a node different from the $3n$ already reachable colour nodes iff two neighbouring nodes are assigned the same colour. Formally, let the TBox be given by

$$\begin{aligned} \mathcal{T} = \{ & V \sqsubseteq \exists hasCol. \top, & \exists hasCol^-. \top \sqsubseteq \exists t. \top, & \exists t^-. \top \sqsubseteq B, \\ & B \sqsubseteq \neg Colour, & u \sqsubseteq \neg hasCol \}. \end{aligned}$$

Furthermore, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let the ABox be given by

$$\begin{aligned} \mathcal{A} = \{ & V(v) \mid v \in \mathcal{V} \} \cup \{ Colour(cv) \mid c \in \{r, g, b\}, v \in \mathcal{V} \} \cup \\ & \{ edge(v, v'), edge(cv, cv') \mid c \in \{r, g, b\}, \{v, v'\} \in \mathcal{E} \} \cup \\ & \{ aux(a, cv) \mid c \in \{r, g, b\}, v \in \mathcal{V} \} \cup \{ aux_2(a, v) \mid v \in \mathcal{V} \} \\ & \{ u(v, cv') \mid c \in \{r, g, b\}, \{v, v'\} \subseteq \mathcal{V}, v \neq v' \}. \end{aligned}$$

Now every vertex needs to have a colour assigned, in the shape of an $hasCol$ outgoing edge. If this is not any of the $3n$ cv nodes, this directly increases the number of reachable nodes to at least $3n + 1$. For each v , the colour assigned has to be one of rv, bv or gv , as the role u is disjoint from $hasCol$. Thus, a 3-colouring is modelled in the sense that each vertex points to at least one of the three colour nodes of that specific vertex. To see that two neighbouring vertices cannot be assigned the same colour, we use the second usage of the $edge$ -edge between the same colour of neighbouring vertices. As only colours with incoming $hasCol$ -edges need to have an outgoing t -edge, this means that finding a reachable node via $aux_2 \cdot hasCol \cdot edge \cdot t$ corresponds to two neighbouring nodes being assigned the same colour. To ensure that the t -edges do not point to other colours, which would not increase the amount of reachable nodes, the concept B is used to make them disjoint from the set of colours. Thus, $\mathcal{G}$ is not 3-colourable iff $3n + 1 \leq \text{certCard}(a, E, (\mathcal{T}, \mathcal{A}))$. ♣

6.3 Fragment of DL-Lite_R

As we have seen in the previous section, if we aim for tractability of MINCOUNT we need to be very careful with mixing the direction of roles in ontologies or queries. Moreover, the same holds for role disjointness. Hence, for the remainder of this chapter we consider the following, strictly less expressive, fragment of DL-Lite_R.

Definition 6.8. A DL-Lite_R[→] TBox may contain concept inclusions of the form $C \sqsubseteq C'$ or $C \sqsubseteq \neg C'$, for C and C' basic concepts, and role inclusions of the form $p \sqsubseteq p'$, for $\{p, p'\} \subseteq N_R$.

Notions of a DL-Lite_R[→] knowledge base and model naturally extend from the DL-Lite_R setting. Note that the only difference between DL-Lite_R and DL-Lite_R[→] is that in DL-Lite_R[→] axioms of the form $p \sqsubseteq p'^{-}$ and $p \sqsubseteq \neg p'$ are not accepted. In this way, we avoid the occurrence of edges labelled with roles going both ways in universal models.

We also define the fragment of counting queries that do not use inverses. In the rest of this chapter, we will mainly use the automaton representation of the counting queries, $(a, \mathcal{M})$, as we will rely heavily on decorating structures with the states of the automaton. Recall that NFA's and regular expressions have the same expressivity, so this is a choice that solely simplifies presentation.

Definition 6.9. A counting query $(a, \mathcal{M})$ is one-way if the alphabet Σ of $\mathcal{M}$ is restricted to N_R .

We note there is one construct in DL-Lite_R[→] that still mentions inverse roles: the construct $C \sqsubseteq \exists p^{-} . \top$. The reason we can allow this construct is because it is not important for MINCOUNT when counting queries are one-way, as we can safely delete all axioms of this form without changing the minimum of $\text{certCard}(a, \mathcal{M}, \mathcal{K})$. Given a DL-Lite_R[→] TBox $\mathcal{T}$, let $\text{rem}(\mathcal{T})$ be the TBox $\mathcal{T}$ where all axioms of the form $C \sqsubseteq \exists p^{-} . \top$ are deleted. Moreover, recall the definition of the austere canonical model $\text{can}(\mathcal{T})$ for DL-Lite_R, and its naming convention for the blank nodes, as introduced in Definition 5.4.

Proposition 6.10. Given a DL-Lite_R[→] knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a one-way counting query $(a, \mathcal{M})$, then

$$\min\{\text{certCard}(a, \mathcal{M}, \mathcal{K})\} = \min\{\text{certCard}(a, \mathcal{M}, (\text{rem}(\mathcal{T}), \mathcal{A}))\}.$$

Proof. First, let $\mathcal{I}$ be a model of $(\mathcal{T}, \mathcal{A})$ such that $|\{(a, e) \in \mathcal{M}^{\mathcal{I}}\}| = \min\{\text{certCard}(a, \mathcal{M}, \mathcal{K})\}$. Clearly, as $\text{rem}(\mathcal{T}) \subseteq \mathcal{T}$, we find that $\mathcal{I}$ is a model of $(\text{rem}(\mathcal{T}), \mathcal{A})$ as well, thus it follows that $\min\{\text{certCard}(a, \mathcal{M}, \mathcal{K})\} \geq \min\{\text{certCard}(a, \mathcal{M}, (\text{rem}(\mathcal{T}), \mathcal{A}))\}$.

For the other direction, let $\mathcal{J}$ be a model of $(\text{rem}(\mathcal{T}), \mathcal{A})$ such that $|\{(a, e) \in \mathcal{M}^{\mathcal{J}}\}| = \min\{\text{certCard}(a, \mathcal{M}, (\text{rem}(\mathcal{T}), \mathcal{A}))\}$. Let $\mathcal{J}' := \text{can}(\mathcal{T}, \mathcal{J})$; clearly, $\mathcal{J}'$ is a model of $\mathcal{T}$. To see that $|\{(a, e) \in \mathcal{M}^{\mathcal{J}}\}| = |\{(a, e) \in \mathcal{M}^{\mathcal{J}'}\}|$, when building $\text{can}(\mathcal{T}, \mathcal{J})$ from $\mathcal{J}$, the only new blank nodes introduced are of the form $ep^{-}w$, for e a blank in $\mathcal{J}$, and $w \in (\overline{N}_R)^*$ the only facts connecting e and ep^{-} are of the form $p'(ep^{-}, e)$, for $p' \in N_R$, i.e., inverted roles. As $\mathcal{M}$ only uses labels in the alphabet N_R , this means that none of the nodes in the newly introduced trees can be reached by $\mathcal{M}$ from a . Thus, also $\min\{\text{certCard}(a, \mathcal{M}, \mathcal{K})\} \leq \min\{\text{certCard}(a, \mathcal{M}, (\text{rem}(\mathcal{T}), \mathcal{A}))\}$. ♣

In the remainder of this chapter we will only focus on one-way counting queries. Furthermore, without loss of generality, we will further assume every $DL\text{-}Lite_{\vec{\mathcal{R}}}$ TBox does not contain axioms of the form $C \sqsubseteq \exists p^-. \top$. For these ontologies we provide the following definition of an ontological type of a node, which is useful for the following sections.

Definition 6.11. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $DL\text{-}Lite_{\vec{\mathcal{R}}}$ KB, and $e \in N_B$ a blank node in $can(\mathcal{K})$. The ontological type of e is the set of all roles $r^- \in \overline{N}_R$ for which $(e', e) \in r^{can(\mathcal{K})}$, all concept names A such that $e \in A^{can(\mathcal{K})}$, and all roles $r \in N_R$ for which $(e, e') \in r^{can(\mathcal{K})}$. For two blank nodes e_1 and e_2 of $can(\mathcal{K})$, we write $e_1 \sim_m e_2$ if they share the same ontological-type.

Even though in $DL\text{-}Lite_{\vec{\mathcal{R}}}$ it suffices to store one role that captures all the above presented information implicitly, we choose to store all information explicitly in the ontological type. Naturally, these choices have the same expressive power, so it is merely a presentation choice: we will be considering models in which multiple nodes are merged into one such that a direct description of the exact in- and outcoming roles and concept names may be conceptually easier to grasp.

6.4 Polynomial-time Algorithm

In the rest of this chapter, without loss of generality, we assume that for all concept inclusions, $C \sqsubseteq D$ in any TBox appearing in this chapter, either C or D is a concept name, or formally: $\{C, D\} \cap N_C \neq \emptyset$.

Most tractable algorithms for query answering under knowledge bases rely on computing a model over which the query can be evaluated to obtain certain answers. Unfortunately, as seen in Examples 6.1 and 6.2, it seems implausible that this technique will suffice for MINCOUNT.

Thus we follow a different path: instead of a single model, we aim to compute a small set Γ of models, in which we are always guaranteed to find one that realises the overall minimum number of answers. We start by defining this condition, which we call counting-sufficient.

Definition 6.12. Let $\mathcal{K}$ be a KB, $(a, \mathcal{M})$ a counting query and Γ a set of models of $\mathcal{K}$. We say that Γ is counting-sufficient for $\mathcal{K}$ and $(a, \mathcal{M})$ if

$$\min\{certCard(a, \mathcal{M}, \mathcal{K})\} = \min_{\mathcal{I} \in \Gamma} |\{e \mid (a, e) \in \mathcal{M}^{\mathcal{I}}\}|.$$

Further, for an NFA $\mathcal{M}$ with states Q , we use $\mathcal{M}[q_1, q_2]$ ($q_1, q_2 \in Q$) to state the NFA in which we replace the initial state by q_1 and the set of final states by $\{q_2\}$.

In order to construct a counting-sufficient set for $DL\text{-}Lite_{\vec{\mathcal{R}}}$ knowledge bases and one-way counting queries $(a, \mathcal{M})$, we start with the universal core model (which coincides with the austere canonical model, as is discussed in Section 4.2) and then look for possible ways to merge blank nodes. However, this must be done with care, as there are too many possible choices for each blank node, and a possible infinite amount of blank nodes to start with, so considering all possible such interpretations would not be feasible.

As a first step, for each node, we collect all states of the automaton that can be reached in some run, which we call the reachable states.

Definition 6.13. For an interpretation $\mathcal{I}$, a node e and a counting query $(a, \mathcal{M})$, the set of reachable states on e in $\mathcal{I}$ with respect to $(a, \mathcal{M})$ is all states q of $\mathcal{M}$ such that $(a, e) \in \mathcal{M}[q_0, q]^{\mathcal{I}}$. We use $S_{\mathcal{I}}(e)$ to denote this set of states, or simply $S(e)$ when $\mathcal{I}$ is understood from context.

We are now ready to define the set of models we consider. Note that any type of universal model suffices, but the universal core model has the advantage it has the least amount of ontological types of all universal models.

In the context of the universal core model construction of $DL\text{-}Lite_{\mathcal{R}}$, we introduce the following notation. For each $e \in N_{\mathcal{K}}$, let e^p denote the (unique) parent of e , that is, there exists $r \in \overline{N}_R$ such that $e^p r = e$.

Definition 6.14. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a pair of a $DL\text{-}Lite_{\mathcal{R}}^{\rightarrow}$ TBox $\mathcal{T}$ and ABox $\mathcal{A}$, $can(\mathcal{K})$ the universal core model of $\mathcal{K}$, and $(a, \mathcal{M})$ a counting query. Define $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ as the set of models $\mathcal{G}$ for which there is a surjective homomorphism $h : \Delta^{can(\mathcal{K})} \rightarrow \Delta^{\mathcal{G}}$ such that:

- $\mathcal{G}$ is the homomorphic image of $can(\mathcal{K})$ under h , and
- For every pair of blank nodes e_1 and e_2 in $can(\mathcal{K})$ with $e_1 \sim_m e_2$ and parents e_1^p and e_2^p , respectively, if $h(e_1^p)$ and $h(e_2^p)$ have the same set of reachable states in $\mathcal{G}$ with respect to $(a, \mathcal{M})$, then $h(e_1) = h(e_2)$.

The main idea is that if the ontological types of two nodes are identical, and moreover their parents have the same set of reachable states, we can just as well replace them by one node, redirecting all incoming edges of the two nodes to this node. Note that this specifically works because we are working with one-way counting queries. As a next step, all possibilities of merging any of these nodes are considered. Both steps are captured in the above definition.

Hence, to construct $\Gamma_{\mathcal{K}, a, \mathcal{M}}$, we consider only those models $\mathcal{G}$ in which we map blank nodes $e_1 \sim_m e_2$ according to how we have mapped their parents e_1^p and e_2^p : if e_1^p and e_2^p have been mapped to two elements that are indistinguishable to $(a, \mathcal{M})$, then e_1 and e_2 have to be mapped to the same element. Next we prove $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ is counting-sufficient.

Proposition 6.15. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a pair of a $DL\text{-}Lite_{\mathcal{R}}^{\rightarrow}$ TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and $(a, \mathcal{M})$ a one-way counting query. The set $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ is counting-sufficient for $\mathcal{K}$ and $(a, \mathcal{M})$.

Proof. Assume $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ is not counting-sufficient. This means there is a model $\mathcal{J} \notin \Gamma_{\mathcal{K}, a, \mathcal{M}}$ such that the cardinality of $\{e \mid (a, e) \in \mathcal{M}^{\mathcal{J}}\}$ is strictly lower than that of $\{e \mid (a, e) \in \mathcal{M}^{\mathcal{G}}\}$ over any $\mathcal{G} \in \Gamma_{\mathcal{K}, a, \mathcal{M}}$. From $\mathcal{J}$ we construct a third model $\mathcal{G}^*$, which does belong to $\Gamma_{\mathcal{K}, a, \mathcal{M}}$, and show that $\{e \mid (a, e) \in \mathcal{M}^{\mathcal{G}^*}\} \leq \{e \mid (a, e) \in \mathcal{M}^{\mathcal{J}}\}$, which contradicts our assumption.

For readability, let us write $\mathcal{I} = can(\mathcal{K})$. Since $\mathcal{I}$ is universal, there is a homomorphism h from $\mathcal{I}$ to $\mathcal{J}$. We assume h is surjective and that $\mathcal{J}$ is the homomorphic image of $\mathcal{I}$ under h , as otherwise

one can drop from $\mathcal{J}$ all atoms with elements that are not in the image of h without increasing the number of answers to $(a, \mathcal{M})$. Since $\mathcal{J}$ is not in $\Gamma_{\mathcal{K}, a, \mathcal{M}}$, there must be a pair $e_1 \sim_m e_2$ of blank nodes in $\mathcal{I}$, with parents e_1^p and e_2^p such that $S_{\mathcal{J}}(h(e_1^p)) = S_{\mathcal{J}}(h(e_2^p))$, and for which $h(e_1) \neq h(e_2)$.

From $\mathcal{J}$ and h we show how to construct an interpretation $\mathcal{J}'$ together with a homomorphism h' that is a quotient of h (that is, h' is obtained from h by merging some target nodes) in which $h'(e_1) = h'(e_2)$, but where the number of answers to $(a, \mathcal{M})$ are not higher than in $\mathcal{J}$. The interpretation $\mathcal{G}^*$ is then constructed by repeatedly performing this operation until no two such blank nodes e_1 and e_2 can be found.

The operation we perform is called a *descendant swap*. Consider two blank nodes e_1 and e_2 that satisfy the conditions given above. and the homomorphism h from $\mathcal{I}$ to $\mathcal{J}$. A descendant swap of $h(e_1) \Rightarrow h(e_2)$ is an operation that produces a new mapping h' and a new model $\mathcal{J}'$, such that h' is a surjective homomorphism from $\mathcal{I}$ to $\mathcal{J}'$, $\mathcal{J}'$ is the homomorphic image of $\mathcal{I}$ under h' , and where $h'(e_1) = h'(e_2)$. The mapping h' is a copy of h , with some exceptions that are defined inductively. First, set $h'(e_1) = h(e_2)$. Then, consider all children e of e_1 and their image $h(e)$ in $\mathcal{J}$. Because $e_1 \sim_m e_2$, for each such child e there must be a node e' in $\mathcal{I}$ that is a child of e_2 , and that has the same type of e . Then, perform a descendant swap of $h(e) \Rightarrow h(e')$.

Finally, let $\mathcal{J}'$ be the homomorphic image of $\mathcal{I}$ under h' . Notice that $\Delta^{\mathcal{J}'} \subseteq \Delta^{\mathcal{J}}$.

Next we show that the answers for $(a, \mathcal{M})$ in $\mathcal{J}'$ is a (possibly equal) subset of those in $\mathcal{J}$. In order to do this, we claim that for any node c in $\mathcal{J}'$ the set $S_{\mathcal{J}'}(c)$ of reachable states on c in $\mathcal{J}'$ with respect to $(a, \mathcal{M})$ is a subset of the set $S_{\mathcal{J}}(c)$ of reachable states on c in $\mathcal{J}$.

Assume the contrary, so that there is an element e in $\mathcal{J}'$ such that $q \in S_{\mathcal{J}'}(e)$ but $q \notin S_{\mathcal{J}}(e)$. This means there is a sequence $r_1(a, e_1), r_2(e_1, e_2), \dots, r_n(e_{n-1}, e)$ of role facts in $\mathcal{J}'$ such that $q \in \delta(q_0, r_1 \cdots r_n)$ where q_0 and δ are the initial state and transition function of $\mathcal{M}$, respectively, and via a run with states $q_0, q_1, \dots, q_n$ with $q = q_n$. Let us further assume without loss of generality that e is the first element for which $S_{\mathcal{J}'}(e) \not\subseteq S_{\mathcal{J}}(e)$ among all e_i 's, i.e., $S_{\mathcal{J}'}(e_i) = S_{\mathcal{J}}(e_i)$ for $1 \leq i \leq n-1$ (otherwise we can just pick $e = e_i$ and use the shorter path). Then, role $r_n(e_{n-1}, e)$ is not in $\mathcal{J}$. But if this is the case, then this means $\mathcal{I}$ contains blank nodes $e_1 \sim_m e_2$ with parents e_1^p and e_2^p , respectively, such that $S_{\mathcal{J}}(h(e_1^p)) = S_{\mathcal{J}}(h(e_2^p))$, and the homomorphism h' was constructed subject to a descendant swap $h(e_1) \Rightarrow h(e_2)$; these are the only elements for which $\mathcal{J}'$ has role facts not in $\mathcal{J}$. Hence $h'(e_1) = h(e_2) = e$, and the parent e_1^p of e_1 satisfies $h'(e_1^p) = e_{n-1}$. But since $e_1 \sim_m e_2$, $\mathcal{J}$ must contain the role fact $r_n(h(e_2^p), h(e_2))$. Together with the fact that $S_{\mathcal{J}}(h(e_1^p)) = S_{\mathcal{J}}(h(e_2^p))$, this implies that q is indeed in $S_{\mathcal{J}}(e)$, because $q_{n-1} \in S_{\mathcal{J}'}(e_{n-1})$, $S_{\mathcal{J}'}(e_{n-1}) = S_{\mathcal{J}}(e_{n-1})$, and $q \in \delta(q_{n-1}, r_n)$. This is a contradiction, and proves the claim.

With this claim we immediately obtain that the number of answers for $(a, \mathcal{M})$ in $\mathcal{J}'$ is a (possibly equal) subset of those in $\mathcal{J}$, as a final state can be reached on an element in $\mathcal{J}'$ with respect to $(a, \mathcal{M})$ only if it can be reached in $\mathcal{J}$. ♣

Of course, we also need $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ to be small, of polynomial size with respect to $\mathcal{A}$, and we must be able to construct $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ efficiently. In order to do this, we prove that every model in $\Gamma_{\mathcal{K}, a, \mathcal{M}}$

can be obtained as the image of $\text{can}(\mathcal{K})$ under a homomorphism generated by a controlled process, which we then bound in terms of the ABox $\mathcal{A}$. More precisely, we first define a family of mappings σ that condition the way we can obtain models as the homomorphic image of $\text{can}(\mathcal{K})$. Then, we explain how to use these mappings to construct models of $\Gamma_{\mathcal{K},a,\mathcal{M}}$. We begin with some definitions.

For a given KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a counting query $(a, \mathcal{M})$, let PT be the set of all ontological types of blank nodes in $\text{can}(\mathcal{K})$. Define the set $T_{\mathcal{K},a,\mathcal{M}} = \{(s, t) \mid s \subseteq Q, t \in PT\}$ of pairs of sets of states and types, and the set $B_{\mathcal{K},a,\mathcal{M}} = \{e_\tau \mid \tau \subseteq T_{\mathcal{K},a,\mathcal{M}}\}$ of blank nodes.

A *generating mapping* for $\mathcal{K}$ and $(a, \mathcal{M})$ is a mapping $\sigma : T_{\mathcal{K},a,\mathcal{M}} \rightarrow \Delta^{\mathcal{A}} \cup B_{\mathcal{K},a,\mathcal{M}} \cup \{\perp\}$ that satisfies the following conditions.

- If $\sigma(s, t) = e_\tau$, for some $e_\tau \in B_{\mathcal{K},a,\mathcal{M}}$, then $(s, t) \in \tau$.
- If e_τ is in the image of some $\sigma(s, t)$, then for each $(s', t') \in \tau$, $\sigma(s', t') = e_\tau$.

Further, a generating mapping σ is *realising* if there is a model $\mathcal{G}$ of $\mathcal{K}$ that satisfies the following conditions. Recall here that $S_{\mathcal{G}}(e)$ denotes the set of states reachable from a by $\mathcal{M}$ in e over the interpretation $\mathcal{G}$.

- $\Delta^{\mathcal{G}} = \Delta^{\mathcal{A}} \cup \{e_\tau \mid e_\tau \text{ appears in the image of } \sigma\}$.
- $\mathcal{G}$ is the homomorphic image of $\text{can}(\mathcal{K})$ under a homomorphism h , that satisfies the following: for every blank e in $\text{can}(\mathcal{K})$, $h(e) = \sigma(S_{\mathcal{G}}(h(e^p)), t)$, where t is the type of e and e^p the parent of e .
- For every pair (s, t) such that $\sigma(s, t) = \perp$, there are no blank nodes e in $\text{can}(\mathcal{K})$ whose parent is e^p , whose type is t and where the set $S_{\mathcal{G}}(h(e^p))$ is s .

In this case, we say that $\mathcal{G}$ is a model *realised* by σ , with h its *realising homomorphism*. As generating mappings realise all models in $\Gamma_{\mathcal{K},a,\mathcal{M}}$ (up to isomorphism), they are particularly useful.

Lemma 6.16. *For any model $\mathcal{G} \in \Gamma_{\mathcal{K},a,\mathcal{M}}$ there is a generating mapping σ for $\mathcal{K}$ and $(a, \mathcal{M})$ such that $\mathcal{G}$ is isomorphic to a model realised by σ .*

Proof. Let $\mathcal{G} \in \Gamma_{\mathcal{K},a,\mathcal{M}}$, $\mathcal{I} = \text{can}(\mathcal{K})$ and $h : \mathcal{I} \rightarrow \mathcal{G}$ a surjective homomorphism such that $\mathcal{G}$ is the homomorphic image of $\mathcal{I}$ under h and such that for every pair of blank nodes e_1 and e_2 in $\mathcal{I}$ with $e_1 \sim_m e_2$ and parents e_1^p and e_2^p , respectively, if $h(e_1^p)$ and $h(e_2^p)$ have the same set of reachable states in $\mathcal{G}$ with respect to $(a, \mathcal{M})$, then $h(e_1) = h(e_2)$.

To define the mapping σ , we proceed in two steps, as follows. First, for each type t of a blank node present in $\mathcal{I}$, consider all blank nodes e of type t in $\mathcal{I}$. We can partition these blank nodes according to the set $S_{\mathcal{G}}(h(e^p))$ of states reachable from $(a, \mathcal{M})$ to the image of their parent e^p , each two such blank nodes with the same set of reachable states must be mapped to the same

element in $\mathcal{G}$. For each such set in this partition, if they are mapped to an element c in $\mathcal{G}$, we set $\sigma(S_{\mathcal{G}}(h(e^p)), t) = c$. For the remaining sets $s \subseteq Q$ for which there are no blank nodes e of type t with parent e^p such that $S_{\mathcal{G}}(h(e^p)) = s$, set $\sigma(s, t) = \perp$.

For the second step, for each blank node d in $\mathcal{G}$, let $(s_1, t_1), \dots, (s_n, t_n)$ be all pairs such that $\sigma(s_i, t_i) = d$. Then, define $\tau = \{(s_1, t_1), \dots, (s_n, t_n)\}$ and set $\sigma(s_i, t_i) = e_\tau$ for each $1 \leq i \leq n$.

The model $\mathcal{G}'$ realised by σ is isomorphic to $\mathcal{G}$: the isomorphism is the identity on constants and maps each blank node d in $\mathcal{G}$ to the corresponding blank node e_τ in $\mathcal{G}'$ according to the construction outlined above. This is well defined because $\mathcal{G}$ is the homomorphic image of $\mathcal{I}$ under h , so every blank node in $\mathcal{G}$ must be the image of a blank node in $\mathcal{I}$. $\clubsuit$

Hence, a bound on the number of models realised by generating mappings translates to a bound on the size of $\Gamma_{\mathcal{K}, a, \mathcal{M}}$. But note that the total number of possible generating mappings is bounded by $(|\Delta^{\mathcal{A}}| + |B_{\mathcal{K}, a, \mathcal{M}}| + 1)^{T_{\mathcal{K}, a, \mathcal{M}}}$, which is a polynomial since the size of $B_{\mathcal{K}, a, \mathcal{M}}$ and $T_{\mathcal{K}, a, \mathcal{M}}$ only depend on $\mathcal{T}$ and $\mathcal{M}$, which we consider to be fixed. Hence, the direct conclusion of the following lemma is that $\Gamma_{\mathcal{K}, a, \mathcal{M}}$ is of polynomial size.

Lemma 6.17. *Every generating mapping σ for $\mathcal{K}$ and $(a, \mathcal{M})$ realises at most one interpretation $\mathcal{G}$, up to isomorphism.*

Proof. Given a model $\mathcal{G}$ realising σ , we can associate a tuple (s, t) to any blank e in $\text{can}(\mathcal{K})$ with parent e^p , where t is the type of e and s corresponds to the set of states that can be reached on $h(e^p)$ in $\mathcal{G}$. In this case, we say that (s, t) is the tuple associated to e in $\mathcal{G}$.

Assume for the sake of a contradiction that the claim is false. Then there are two different interpretations $\mathcal{G}_1, \mathcal{G}_2$ realised for the same generating mapping σ , and let $h_{\mathcal{G}_1}$ and $h_{\mathcal{G}_2}$ be the corresponding realising homomorphisms for $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. Clearly, these two interpretations have the same domain. Further, for every tuple (s, t) in $T_{\mathcal{K}, a, \mathcal{M}}$, there is a blank in $\text{can}(\mathcal{K})$ associated to (s, t) in $\mathcal{G}_1$ if and only if there is a (possibly different) blank in $\text{can}(\mathcal{K})$ that is associated to (s, t) in $\mathcal{G}_2$. In other words, for every type t , there is a blank e_1 with type t , parent e_1^p and $S_{\mathcal{G}_1}(h(e_1^p)) = s$ if and only if there is a blank e_2 with type t , parent e_2^p and such that $S_{\mathcal{G}_2}(h(e_2^p)) = s$.

We will show that all blank nodes in $\text{can}(\mathcal{K})$ are associated to the same tuple (s, t) in both $\mathcal{G}_1$ and $\mathcal{G}_2$. Then by construction we verify that $h_{\mathcal{G}_1}(e) = h_{\mathcal{G}_2}(e)$ for all blank nodes e in $\text{can}(\mathcal{K})$, and since $\mathcal{G}_1$ and $\mathcal{G}_2$ are the homomorphic images of these homomorphisms we can immediately prove that $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic.

In order to show this we prove a somewhat stronger statement: for every element e in $\mathcal{G}_1$ and $\mathcal{G}_2$ (recall they have the same domain), we have that $S_{\mathcal{G}_1}(e) = S_{\mathcal{G}_2}(e)$.

For the sake of a contradiction, assume that there is an element d on which we can reach a state q in $\mathcal{G}_1$ but not in $\mathcal{G}_2$, i.e., for which $q \in S_{\mathcal{G}_1}(d), q \notin S_{\mathcal{G}_2}(d)$. As with the proof of Lemma 6.16, this is because of a path starting in a that uses a role fact that is in $\mathcal{G}_1$ but not on $\mathcal{G}_2$, and we can assume without loss of generality that this path ends in d exactly after visiting this role fact,

say $r(d', d)$, so that there is a state $q' \in S_{\mathcal{G}_1}$ with $q \in \delta(q', r)$; otherwise we can always pick an element for which the set of reachable states do not coincide but that is closer to a . Notice that the role fact cannot be of the form $r(d, d')$ because the query is one-way. Moreover, since there are no inverses in the axioms of $\mathcal{T}$, and $r(d', d)$ is not in $\mathcal{G}_2$, it further follows that d must be the image of a blank node e in $\text{can}(\mathcal{K})$, so that $h_{\mathcal{G}_1}(e) = d$, and if e^p is the parent of e , $h_{\mathcal{G}_1}(e^p) = d'$. Assume that e is associated with the tuple (s, t) in $\mathcal{G}_1$, that is, it has type t and $S_{\mathcal{G}_1}(h(e^p)) = s$, and therefore $\sigma(s, t) = d$. Because $\mathcal{G}_2$ is realising σ , there must exist an element e' (possibly distinct from e) associated to the tuple (s, t) in $\mathcal{G}_2$, so then $h_{\mathcal{G}_2}(e') = d$. But because e and e' are associated to the same pair (s, t) , the parent e'^p of e' satisfies $S_{\mathcal{G}_2}(h_{\mathcal{G}_2}(e'^p)) = s$, and since e and e' have the same type, the role fact $r(h_{\mathcal{G}_2}(e'^p), d)$ exists in $\mathcal{G}_2$. But then this shows that q does belong to $S_{\mathcal{G}_2}(d)$, through a path that leads to $h(e'^p)$ in state q' and then through $r(h_{\mathcal{G}_2}(e'^p), d)$. This is a contradiction.

We have shown that $S_{\mathcal{G}_1}(c) = S_{\mathcal{G}_2}(c)$ for every element c in $\mathcal{G}_1$ and $\mathcal{G}_2$. To show that all blank nodes in $\text{can}(\mathcal{K})$ are associated to the same tuple (s, t) in both $\mathcal{G}_1$ and $\mathcal{G}_2$, we perform an induction on the length of the name of the blank nodes, for this, recall the naming convention introduced in Definition 5.4. If a node is of the form ar , for $a \in \Delta^{\mathcal{A}}$, and of type t , then $S_{\mathcal{G}_1}(a) = S_{\mathcal{G}_2}(a)$ and hence ar is associated to (s, t) in both $\mathcal{G}_1$ and $\mathcal{G}_2$, which means $h_{\mathcal{G}_1}(ar) = h_{\mathcal{G}_2}(ar)$. And similarly for a node wr for $|w| > 1$, of type t , if we assume that $S_{\mathcal{G}_1}(w) = S_{\mathcal{G}_2}(w)$, then this also means wr is associated to the same tuple in both graphs. ♣

Finally, our last result shows that one can compute realising models efficiently in data complexity. This is not only important to show tractability of MINCOUNT, but will also play a role in the following section, where we show how to rewrite this algorithm.

Before we prove this, let $Bl(\sigma)$ contain all blanks occurring in the image of σ , or formally:

$$Bl(\sigma) := \{e_\tau \in B \mid e_\tau \text{ occurs in the image of } \sigma\}.$$

Lemma 6.18. *Given a generating mapping σ , there is a polynomial-time procedure (in data complexity) that outputs the realised model of σ if it exists, and otherwise indicates it does not.*

Proof. We first compute a set of atoms $\mathcal{G}$, then check whether its canonical interpretation is realising for σ . Next, we use a decoration that maintains which automaton states are reachable in each node, if we are to assume nodes from $\text{can}(\mathcal{K})$ are correctly mapped to $\mathcal{G}$ according to σ . To do this, let $\text{dec}(\sigma)$ be the smallest set of atoms such that $q_0(a) \in \text{dec}(\sigma)$, and moreover

- if $q' \in \delta(q, r)$, $r(e, e') \in \mathcal{A}$, $q(e) \in \text{dec}(\sigma)$, then $q'(e') \in \text{dec}(\sigma)$;
- for all $e \in Bl(\sigma)$, if $(s, t) \in \sigma^-(e)$, $q' \in \delta(q, r)$, $q \in s$, $r^- \in t$, then $q'(e) \in \text{dec}(\sigma)$.

Based on the above decoration, we define the following function $\rho_\sigma : Bl(\sigma) \cup \Delta^{\mathcal{A}} \rightarrow \mathcal{P}(Q)$, which maps nodes to sets of states of $\mathcal{M}$, as follows:

$$\rho_\sigma(e) := \{q \mid q(e) \in \text{dec}(\sigma)\}.$$

Now, we apply the following steps until no more atoms are added to $\mathcal{G}$. Here, let h be the (unique) realising homomorphism of σ .

- Start with $\mathcal{G} = \mathcal{A}$, $h(a) = a$ for each of these elements, and start with $\ell = 0$.
- For each blank node wr in $\text{can}(\mathcal{K})$ where $|w| = \ell$, let $s = \rho_\sigma(h(w))$ and t the type of wr . If $\sigma(s, t) = \perp$, exit reporting failure. Otherwise $h(wr) = \sigma(s, t)$. Add to $\mathcal{G}$ all atoms $A(h(wr))$ such that $wr \in A^{\text{can}(\mathcal{K})}$ and all atoms $r(h(w), h(wr))$ such that $(w, wr) \in r^{\text{can}(\mathcal{K})}$.
- After processing all such blank nodes, $\ell = \ell + 1$. If no more pairs were added to $\mathcal{G}$, return.

Next, check whether $S_{\mathcal{G}}(e) = \rho_\sigma(e)$ for each $e \in \Delta^{\mathcal{G}}$. If it is the case, then output $\mathcal{G}$. Otherwise σ is not realising. ♣

With this, we finally obtain our PTIME algorithm.

Theorem 6.19. *Problem MINCOUNT is in PTIME in data complexity when the knowledge base is in $\text{DL-Lite}_{\mathcal{R}}^{\rightarrow}$ and $(a, \mathcal{M})$ is a one-way counting query.*

Proof. We enumerate each generating mapping σ one by one. Note there are at most $(|\Delta^{\mathcal{A}}| + |B_{\mathcal{K}, a, \mathcal{M}}| + 1)^{T_{\mathcal{K}, a, \mathcal{M}}}$ of them. For each such mapping σ , we can discard or produce its realising model in polynomial time, according to Lemma 6.18, which is unique for σ (Lemma 6.17). We evaluate the counting query in this model, and maintain the minimum number of answers as we go over all possible models realising each different generating mapping σ . By Lemma 6.16, we can be sure we have generated models that are isomorphic to every model in $\Gamma_{\mathcal{K}, a, \mathcal{M}}$, which we know is counting-sufficient for one-way queries (Proposition 6.15). ♣

6.5 Decision Algorithm

Our PTIME algorithm involves analysing a counting-sufficient set of models, looking for the one that produces the least amount of answers. While tractable, this involves computing and materialising a number of graphs, which is something neither database systems nor logic solvers are well prepared to do. This, then, raises the question on whether there is any form of rewriting procedure that would allow computing the minimal number of answers simply by posing a query over the data.

Unfortunately, such a rewriting is not feasible in general: some queries may have a minimum number of answers that exceed the number of elements in the domain of the ABox. What we can do, however, is to solve the problem of deciding whether k is the smallest integer in the set of cardinality certain answers of a counting query. In other words, we cannot compute the minimum number of answers by a rewriting, but we can solve the problem of deciding if k is such a minimum.

To establish our rewriting we rely on the existence of a counting-sufficient set of models, and gear the rewriting so that it appropriately explores them—without any explicit materialisation. The result of our rewriting is a big disjunction of $\top$ and/or $\perp$, and can thus also be considered a decision algorithm, as the result no longer needs to be queried over a data graph.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $DL\text{-}Lite_{\mathcal{R}}^{\rightarrow}$ KB and $(a, \mathcal{M})$ a counting query. Our first step is to compute all sets of reachable states for the model being realised by a given generating mapping $\sigma : T_{\mathcal{K}, a, \mathcal{M}} \rightarrow \Delta^{\mathcal{A}} \cup B_{\mathcal{K}, a, \mathcal{M}} \cup \{\perp\}$ for $\mathcal{K}$ and $(a, \mathcal{M})$. To do this, for clarity, let us first define the domain of possible realising models as $\Delta_{\sigma} = \Delta^{\mathcal{A}} \cup B_{\mathcal{K}, a, \mathcal{M}}$. Furthermore, recall the definition of $dec(\sigma)$ as the smallest set of atoms such that $q_0(a) \in dec(\sigma)$, and moreover

- if $q' \in \delta(q, r)$, $r(e, e') \in \mathcal{A}$, $q(e) \in dec(\sigma)$, then $q'(e') \in dec(\sigma)$;
- for all $e \in Bl(\sigma)$, if $(s, t) \in \sigma^{-}(e)$, $q' \in \delta(q, r)$, $q \in s$, $r^{-} \in t$, then $q'(e) \in dec(\sigma)$.

Note that this decoration collects which automaton states are reachable in which nodes of the graph. Thus, when $q(e) \in dec(\sigma)$, for $q \in Q_F$, this means that e is reachable by E , the regular expression corresponding to $\mathcal{M}$, from a .

Based on the above decoration, we define the following function $\rho_{\sigma} : \Delta_{\sigma} \rightarrow \mathcal{P}(Q)$, which maps nodes to sets of states $\mathcal{M}$, as follows.

$$\rho_{\sigma}(e) := \{q \mid q(e) \in dec(\sigma)\}$$

Using this function, we can now filter out generating mappings that cannot produce a model. To this extent, given an element $e \in \Delta_{\sigma}$, let

$$\begin{aligned} LC(e) = \{A \in N_C \mid A(e) \in \mathcal{A}\} \cup \{ \exists r. \top \mid r \in \overline{N}_R, r(e, e') \in \mathcal{A} \} \cup \\ \bigcup_{(s, t) \in \sigma^{-}(e)} \left(\{A \in t\} \cup \{ \exists r. \top \mid r \in t \} \right). \end{aligned}$$

Intuitively, $LC(e)$ contains all basic concepts that would hold in e in the model realised by σ . Note that because we have collected the types of all nodes that are intuitively speaking merged in e , this simply corresponds to reading off the information from the there mapped types t , combined with the ABox atoms. Moreover, for $r \in N_R$, define the r -type as the ontological type given by $\{r_1^{-} \in \overline{N}_R \mid \mathcal{T} \models r \sqsubseteq r_1\} \cup \{A \in N_C \mid \mathcal{T} \models \exists r^{-}. \top \sqsubseteq A\} \cup \{r_1 \in \overline{N}_R \mid \mathcal{T} \models \exists r^{-}. \top \sqsubseteq \exists r_1. \top\}$. Intuitively, imagine we observe a node with an r -incoming role, then the ontological type of this node consists of the inverse of all roles directly implied by r , all concept names implied by $\exists r^{-}. \top$, and all roles introduced by an existential restriction. Since in $DL\text{-}Lite_{\mathcal{R}}^{\rightarrow}$ the ontological type of a blank node in $can(\mathcal{K})$ depends only on a single incoming role, the r -type coincides with the type of all blank nodes in $can(\mathcal{K})$ of the form wr .¹

Finally, let Π contain all generating mappings σ that verify all of the following conditions.

¹ See the construction of the austere canonical model for $DL\text{-}Lite_{\mathcal{R}}$ in Definition 5.4 for more details.

1. We do not merge incompatible types: for each $e \in \Delta_\sigma$ such that $\sigma^-(e) \neq \emptyset$, there are no $\{C, D\} \subseteq \text{LC}(e)$ such that $\mathcal{T} \models C \sqsubseteq \neg D$.
2. When a child is created, it is mapped according to σ : for each $e \in \Delta_\sigma$, $C \in \text{LC}(e)$ such that $\mathcal{T} \models C \sqsubseteq \exists r. \top$, let $s = \rho_\sigma(e)$ and t_r the r -type. Then, $\sigma(s, t_r) \neq \perp$. Moreover, $\{\delta(q', r) \mid q' \in s\} \subseteq \rho_\sigma(\sigma(s, t_r))$.

Lemma 6.20. *Every mapping σ in Π is realising, and every realising mapping σ is in Π .*

Proof. The fact that every realising mapping σ is in Π is obtained immediately by inspecting its corresponding realising model. Hence, all we need to show is that every mapping σ in Π is realising.

Fix $\sigma \in \Pi$. We construct an interpretation $\mathcal{G}_\sigma$ and then define a homomorphism $h : \text{can}(\mathcal{K}) \rightarrow \mathcal{G}_\sigma$ witnessing that σ is realising.

The interpretation $\mathcal{G}_\sigma$ is defined as follows:

- $\Delta^{\mathcal{G}_\sigma} = \Delta^{\mathcal{A}} \cup \text{Bl}(\sigma)$
- $A^{\mathcal{G}_\sigma} = \{e \in A \mid A(e) \in \mathcal{A}\} \cup \{e_\tau \in \text{Bl}(\sigma) \mid A \in \text{LC}(e_\tau)\}$.
- To define $r^{\mathcal{G}_\sigma}$, start with $r^{\mathcal{G}_\sigma} = r^{\mathcal{A}}$ for each $r \in N_R$. Then, apply the following rule until no more pairs are added: for every $e \in \Delta_\sigma^{\mathcal{G}}$, every $C \in \text{LC}(e)$ and every $r \in N_R$ such that $\mathcal{T} \models C \sqsubseteq \exists r. \top$, let $s = \rho_\sigma(e)$ and let t_r be the r -type. Because $\sigma \in \Pi$, we know that $\sigma(s, t_r) \neq \perp$. Then, add the pair $(e, \sigma(s, t_r))$ to $r^{\mathcal{G}_\sigma}$, and any $r_1^{\mathcal{G}_\sigma}$ such that $\mathcal{T} \models r \sqsubseteq r_1$.

By condition (1) of Π , the interpretation $\mathcal{G}_\sigma$ satisfies all negative concept of $\mathcal{T}$ (no element is labelled with $\mathcal{T}$ -incompatible basic concepts). Moreover, by construction of $r^{\mathcal{G}_\sigma}$, every existential requirement $\mathcal{T} \models C \sqsubseteq \exists r. \top$ with $C \in \text{LC}(e)$ is witnessed by the edge $(e, \sigma(\rho_\sigma(e), t_r)) \in r^{\mathcal{G}_\sigma}$. All together, this shows that $\mathcal{G}_\sigma \models \mathcal{K}$.

It remains to construct the realising homomorphism h , and show $\mathcal{G}_\sigma$ is indeed realised by h . We define $h : \Delta^{\text{can}(\mathcal{K})} \rightarrow \Delta^{\mathcal{G}_\sigma}$ as the identity over elements in $\mathcal{A}$. Moreover, for every blank node e of $\text{can}(\mathcal{K})$ with parent e^p and ontological type t , set $h(e) = \sigma(\rho_\sigma(h(e^p)), t)$.

We note that h is well-defined. Indeed, let e be a blank node of $\text{can}(\mathcal{K})$ with parent e^p and ontological type t . Such a blank node exists because an axiom $C \sqsubseteq \exists r. \top$ is triggered at e^p , and t is precisely the canonical r -successor type t_r . By the second condition of Π , letting $s := \rho_\sigma(h(e^p))$, we must have $\sigma(s, t_r) \neq \perp$.

By construction of $\mathcal{G}_\sigma$, it is easy to see that h is the homomorphic image of $\text{can}(\mathcal{K})$, and that h is realising. This finishes the proof. $\clubsuit$

Next, for every mapping in Π , we guess which nodes are in the answer of $(a, \mathcal{M})$. For this we define another family of functions Υ such that $v : \Delta_\sigma \rightarrow \{+, -\}$ for each $v \in \Upsilon$. If $|\{e \in \Delta_\sigma \mid v(e) = +\}| = i$, we say $v \in \Upsilon_i$. The main idea is that only nodes labelled with ‘+’

can be labelled with a final state. In this way, we check whether it is possible to assign final states to at most i nodes of the model realised.

$$\gamma(\sigma, v) := \begin{cases} \top & \text{for all } q \in Q_F, e \in \Delta_\sigma, \text{ if} \\ & q \in \rho_\sigma(e), \text{ then } v(e) = + \\ \perp & \text{otherwise.} \end{cases}$$

The only thing left to do is check whether this is possible for at least one of the mappings in Π .

Theorem 6.21. *Given a $DL\text{-}Lite_{\vec{\mathcal{R}}}$ knowledge base $(\mathcal{T}, \mathcal{A})$ and a one-way counting query $(a, \mathcal{M})$, we find that the minimum certain answer $\min\{certCard(a, \mathcal{M}, \mathcal{K})\} = k$ iff k is the largest natural number such that*

$$\bigvee_{\sigma \in \Pi, v \in \Upsilon_{k-1}} \gamma(\sigma, v)$$

evaluates to false.

Proof Sketch. Let Π contain all generating mappings as described. By Lemma 6.20, we find that all generating mappings σ that produce a model are exactly contained in Π (recall that each σ has at most one realising model (Lemma 6.17)). Thus, also the set of models generated by the mappings in Π must be counting-sufficient for one-way queries (Proposition 6.15). That is, there exists at least one model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$ generated by some σ such that the amount of nodes that are decorated with final states is optimal.

Secondly, note that $\min\{certCard(a, \mathcal{M}, \mathcal{K})\} = k$ iff k is the highest number such that it is not possible to find a $\sigma \in \Pi$ such that the amount of reachable states in $\mathcal{I}$ realised by σ is at most $k - 1$. As $\bigvee_{v \in \Upsilon_{k-1}} \gamma(\sigma, v)$ evaluating to false corresponds to not finding at most $k - 1$ states that may be decorated with a final state, it means that taking the disjunct over all $\sigma \in \Pi$ exactly checks the existence of such an $\mathcal{I}$.

6.6 Conclusion

We have investigated the problem of counting the number of answers to regular path queries under ontological constraints. Our results tighten the known complexity bounds for neighbouring settings and provide a more precise understanding of how counting interacts with expressive features of description logics. In particular, we show that inverse roles are a decisive source of hardness, leading to coNP-hardness bounds already in very basic fragments.

Beyond complexity-theoretic consequences, our results have implications for the design of graph schema languages such as SHACL. They indicate that inverse roles become particularly delicate when schema-level constraints are combined with ontological reasoning in the presence of counting features.

Regarding future work, it remains to delineate the exact frontier of tractability. While $\mathcal{EL}_{\perp}$ lies beyond the manageable boundary (cf. Proposition 6.4), it is open whether similar techniques

extend to $\mathcal{EL}$ or $\mathcal{ELH}$, or whether fundamentally new phenomena arise in those fragments. We also believe the coNP hardness bounds should persist even in the absence of inverse roles in the ontology, provided that queries are allowed to form trees. This is because they can simulate inverse-like interactions by enforcing that both endpoints of an edge satisfy the same property; some insight in this direction has already been shown by Bienvenu, Manière and Thomazo [BMT22].

6.7 Related Work

Apart from the works discussed in this thesis, different approaches exist towards reconciling SHACL and OWL [SKL19, KZA24]. Nevertheless, none of these works considers counting over regular expressions, one of the essential features of SHACL, in the presence of ontologies. The aim of this chapter is to address this gap in the literature.

Although conjunctive queries with some form of counting over knowledge bases have been studied before [CKNT08, KR15, CCLR20], even for unions of conjunctive queries [FLP21], and also in the more general setting of counting conjunctive queries (CCQs), in which there is complete freedom on the choice of which variables are counted [BMT20, BMT22], none of these works addresses counting the number of nodes reachable by a regular expression. In particular, we are not interested in the number of paths leading to a certain node, but only the size of the set of nodes reachable from an input node. That is, we address a specific form of counting rooted queries [NKK⁺19, BMT20, CCLR20], namely those with a line as underlying structure, which are at the same time a union of a possibly infinite set of CCQs with one counting variable.

To do this, we extend the framework of *cardinality certain answers* [KR15, BMT20], originally designed for conjunctive queries over ontologies, to counting the number of nodes reachable via a regular path expression (over ontologies).

In the terminology of graph query languages (see e.g. [PSH13]), the type of queries we consider are count-distinct queries: we are interested in the number of elements n that can be reached from a by means of $\mathcal{M}$, and not in the number of paths leading there.

Kostylev and Reutter [KR15] studied a version of MINCOUNT for conjunctive queries (CQs), showing the problem is already coNP-hard for very simple ontologies. These lower bounds were further refined by Bienvenu, Manière and Thomazo [BMT20] for what they call *rooted* CQs. However, these classes of queries can not be rewritten into regular path expressions, because they go beyond their expressive power. This opens up the possibility of polynomial algorithms for MINCOUNT.

Regarding the unfeasibility of a rewriting procedure that reduces the problem we study to a query over data, we remark that there may be other ways to relax the problem than discussed in this chapter and obtain a rewriting, for example, Calvanese et al. [CCLR20] use weights to multiply the counts of some elements in the data. It is open whether such a strategy can work in our case.

SHACL satisfiability via Description Logics

In this chapter, we consider satisfiability under the supported model semantics. We build on Description Logics (DLs), a well-known family of languages for Knowledge Representation and Reasoning that offers decades of research in the fine-grained study of logical fragments and the effect that the interaction between different shapes of subformulas has on the complexity of reasoning. The close relationship between DLs and SHACL is well-known, and in this chapter, we leverage it to paint a much finer boundary of SHACL fragments that have decidable satisfiability problems, both over unrestricted graphs and over graphs with a finite domain. We note that this connection to DLs works particularly well for the supported model semantics, as opposed to the well-founded or stable model semantics. For the latter one, an EXPTIME complexity result for satisfiability has been derived by comparing it to DL terminologies [DSŠ24], but for DL terminologies much less complexity results are known. For the well-founded semantics, instead, the connection to full hybrid μ -calculus will be insightful, which will be discussed in the next chapter. For these reasons, we first focus on the supported model semantics only in this chapter.

That is, we revisit the connection to DLs and explain how to translate complexity results in both ways. To emphasise this tight bond, we provide a DL inspired naming convention: we write $\mathcal{L}_S$ to denote the SHACL fragment similar to the DL $\mathcal{L}$. Moreover, we add some lack of finite model property results to the landscape: we show this for $\mathcal{ALC}_S$ plus counting over regular path expressions, which also provides an alternative undecidability proof; and, we show that adding either $\text{eq}(E, r)$ or $\text{disj}(E, r)$ to $\mathcal{ALC}_S$ also breaks the finite model property of $\mathcal{ALC}_S$.

7.1 SHACL Satisfiability

In this chapter we study the following reasoning problem, specifically for the supported model semantics.

(Finite) document satisfiability: Given a SHACL document $(\mathcal{C}, \mathcal{G})$, decide whether there exists a (finite) graph that satisfies $(\mathcal{C}, \mathcal{G})$ under $\mathbb{S}$.

We also study the following property, which guarantees that finite and general version coincide:

Finite model property: Is it the case that, under $\mathbb{S}$, every satisfiable SHACL document is finitely satisfiable?

Clearly, having the finite model property extends to less expressive fragments, whereas the opposite, not having the property, spreads to subsuming fragments. Similarly, for (finite) satisfiability, membership of a complexity class spreads to less expressive fragments, and hardness to the more expressive ones. In case a fragment has the finite model property, the membership results for general satisfiability extend to the finite setting.

The above presented problems are not the only ones one might consider: in [PKMN20], also another version of the document satisfiability problem is discussed: *constraint satisfiability*. This corresponds to the satisfiability problem when the constraint set only consists of one constraint, and only one target:

(Finite) constraint satisfiability Given a SHACL document $(s \leftarrow \varphi, s(c))$, decide whether there exists a (finite) graph that satisfies $(s \leftarrow \varphi, s(c))$ under $\mathbb{S}$.

As already noted in [PKMN20], the constraint version of the problem trivially reduces to the general version, which means upper bounds for complexity are preserved. We show here that for recursive SHACL also the other reduction holds. First, we note that for satisfiability purposes, we may restrict the form of the targets. Note that the following works construction works independent of the precise (recursive) SHACL fragment considered.

Lemma 7.1. *For each SHACL document $(\mathcal{C}, \mathcal{G})$ there exists a SHACL document $(\mathcal{C}', \mathcal{G}')$ such that $\mathcal{G}'$ only consists of shape atoms, such that there exists an $\mathcal{I}$ that validates $(\mathcal{C}, \mathcal{G})$ iff there exists a $\mathcal{J}$ that validates $(\mathcal{C}', \mathcal{G}')$ under the supported model semantics.*

Proof. Let $R \subseteq \overline{N}_R$ be the set of all roles appearing in $\mathcal{C}$. Let

$$\mathcal{C}' := \mathcal{C} \cup \left\{ \hat{s} \leftarrow \bigvee_{r \in R \cup R^-} \forall r. (\hat{s} \wedge \bigwedge_{(A,s) \in \mathcal{G}} (\neg A \vee s) \wedge \bigwedge_{(r,s') \in \mathcal{G}} (\neg \exists r. \top \vee s') \right\}$$

for a fresh shape name $\hat{s}$ and set

$$\mathcal{G}' := (\mathcal{G} \cup \{(a, \hat{s}) \mid (a, s) \in \mathcal{G} \text{ for some } s \in N_S\}) \setminus \{(\ell, s) \in \mathcal{G} \subseteq (N_C \cup \overline{N}_R) \times N_S\}.$$

The main idea is that $\hat{s}$ traverses through all nodes connected to any individual appearing in a target in $\mathcal{G}$, and enforces that all of these nodes that are labelled with A also satisfy s for each $(A, s) \in \mathcal{G}$, and that all nodes that have an outgoing r edge also satisfy s' for each $(r, s') \in \mathcal{G}$.

So, if we assume there exists an interpretation $\mathcal{I}$ that validates $(\mathcal{C}, \mathcal{G})$, then $\mathcal{I}$ also validates $(\mathcal{C}', \mathcal{G}')$, simply by the shape assignment that adds $\hat{s}(e)$ for all $e \in \Delta^{\mathcal{I}}$. For the other direction, assume there exists a $\mathcal{J}$ that validates $(\mathcal{C}', \mathcal{G}')$. Let $\mathcal{I}$ be the interpretation that is obtained from excluding all nodes not reachable by a path over $R \cup R^-$ from an individual appearing in $\mathcal{G}$. Then $\mathcal{I}$ is validating $(\mathcal{C}, \mathcal{G})$: simply remove all shape atoms of the form $\hat{s}(e)$ and all atoms $s(e)$ such that $e \notin \Delta^{\mathcal{I}}$. ♣

We can now use this result in the following reduction.

Proposition 7.2. *In recursive SHACL, the problems of deciding document satisfiability and constraint satisfiability are mutually reducible in polynomial time in $|\mathcal{C}| + |\mathcal{G}|$ under the supported model semantics.*

Proof. First of all, note that one reduction is trivial, and pointed out in [PKMN20]. Thus, this proof focusses on how to reduce document satisfiability to constraint satisfiability. We use ' $\varphi \leftrightarrow \psi$ ' as a shorthand for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, and ' $\varphi \rightarrow \psi$ ' for $\neg\varphi \vee \psi$.

Given a SHACL document $(\mathcal{C}, \mathcal{G})$, we assume that all targets in $\mathcal{G}$ are shape atoms (Lemma 7.1). We distinguish two cases: whether the SHACL fragment allows to use nominals in shape expressions, or not.

In case it does not contains nominals, satisfiability of $(\mathcal{C}, \mathcal{G})$ is equivalent to satisfiability of all $(\mathcal{C}, \mathcal{G}_c)$, where $\mathcal{G}_c := \{s(c) \in \mathcal{G}\}$, for all $c \in N_I$ such that c appears in $\mathcal{G}$. To see why this is sufficient, note we can assume that the interpretations $\mathcal{I}_c$ validating each of the $(\mathcal{C}, \mathcal{G}_c)$ have mutually disjoint sets of domain elements, as there are no nominals in the fragment. Thus $\mathcal{I}$, defined as $\Delta^{\mathcal{I}} := \bigcup_c \Delta^{\mathcal{I}_c}$, $A^{\mathcal{I}} := \bigcup_c A^{\mathcal{I}_c}$ for all $A \in N_C$ and $p^{\mathcal{I}} := \bigcup_c p^{\mathcal{I}_c}$ for all $p \in N_R$, validates $(\mathcal{C}, \mathcal{G})$. The other direction is immediate.

Secondly, we can replace multiple targets using the same c by a single target $s(c)$ for some fresh shape name s , simply by adding $s' \leftarrow \bigwedge_{s(c) \in \mathcal{G}} s$ to the set of constraints. That is, for every interpretation $\mathcal{I}$, we find $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G}_c)$ iff $\mathcal{I}$ validates $(\mathcal{C} \cup \{s' \leftarrow \bigwedge_{s(c) \in \mathcal{G}} s, \{s'(c)\}\})$.

The last step is to construct a single constraint that is implicitly firing the original constraints: satisfiability of $(\mathcal{C}, \{s(c)\})$ can be reduced to satisfiability of $(\{\hat{s} \leftarrow \hat{\varphi}\}, \{\hat{s}(c)\})$, for a fresh shape name $\hat{s}$, and $\hat{\varphi}$ defined in the following way:

$$\hat{\varphi} := s \wedge \bigvee_{r \in R \cup R^-} \forall r. (\hat{s} \wedge \bigwedge_{s' \leftarrow \varphi \in \mathcal{C}} (s' \leftrightarrow \varphi)),$$

where $R \subseteq \overline{N}_R$ contains all roles appearing in any constraint in $\mathcal{C}$. Note that this constraint first of all ensures that s needs to hold at c . Moreover, all nodes connected to c need to satisfy $\bigwedge_{s' \leftarrow \varphi \in \mathcal{C}} (s' \leftrightarrow \varphi)$. In particular $s \leftarrow \varphi \in \mathcal{C}$, for some φ . That is, c also needs to satisfy φ . For every shape name occurring in φ , again $\bigwedge_{s' \leftarrow \varphi \in \mathcal{C}} (s' \leftrightarrow \varphi)$ ensures that also these shape names satisfy the constraints collected in $\mathcal{C}$. Thus, for every interpretation $\mathcal{I}$, $\mathcal{I}$ validates $\mathcal{C}, \{s(c)\}$ iff $\mathcal{I}$ validates $(\{\hat{s} \leftarrow \hat{\varphi}\}, \{\hat{s}(c)\})$.

Combining the result of these three stages provides the required result: for each SHACL document $(\mathcal{C}, \mathcal{G})$, there exists a SHACL document $(s \leftarrow \varphi, s(c))$ such that $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{I}$ validates $(s \leftarrow \varphi, s(c))$ under the supported model semantics.

In case it does contain nominals, the first stage of the previous construction is no longer sound. Instead we can directly define a single constraint in the following way: satisfiability of $(\mathcal{C}, \mathcal{G})$, such that $\mathcal{G}$ only contains shape atoms may be reduced to satisfiability of $(\{\hat{s} \leftarrow \hat{\varphi}\}, \{\hat{s}(c)\})$ for $\hat{s}$ a fresh shape name and some $c \in N_I$, such that

$$\hat{\varphi} := \bigwedge_{s(c) \in \mathcal{G}} \exists \hat{p}. (c \wedge s \wedge \hat{s}) \wedge \bigvee_{r \in R \cup R^-} \forall r. (\hat{s} \bigwedge_{s \leftarrow \varphi \in \mathcal{C}} (s \leftrightarrow \varphi)),$$

where $R \subseteq \overline{N}_R$ contains all roles appearing in any constraint in $\mathcal{C}$ and $\hat{p} \notin R \cup R^-$.

In this reduction, the latter part remains the same to the previous reduction. That is, it safes all constraints inside this single constraint and accesses the right φ whenever s needs to be satisfied. The first half ensures that all individuals occurring in $\mathcal{G}$ are assigned their according shape names. Thus, we find that for each interpretation $\mathcal{I}$, we have that $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{I}'$ validates $(\{\hat{s} \leftarrow \hat{\varphi}\}, \{\hat{s}(c)\})$, where $\mathcal{I}'$ is given by $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{c\}$, $\hat{p}^{\mathcal{I}} = \{(c, a) \mid s(a) \in \mathcal{G}\}$. As $\hat{p} \notin R \cup R^-$, meaning that this addition of $\hat{p}$ -pairs is always possible, we indeed find the required result: for each SHACL document $(\mathcal{C}, \mathcal{G})$, there exists a SHACL document $(s \leftarrow \varphi, s(c))$ such that $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{I}'$ validates $(s \leftarrow \varphi, s(c))$ under the supported model semantics. ♣

Names for fragments of SHACL. Let $\mathcal{ALC}_S$ be the fragment of SHACL such that shape expressions φ are of the form:

$$\varphi ::= s \mid A \mid \top \mid \neg \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists r. \varphi \mid \forall r. \varphi,$$

for $r \in N_R$. Let $\exists r. \varphi$ be a shorthand for $\geq_1 r. \varphi$. Partly following the naming convention of Description Logics, we identify the SHACL fragments in the way presented in Table 7.1. We write $\mathcal{LX}_S$ to denote the SHACL fragment by extending $\mathcal{L}_S$ with the features described by some $X \subseteq \{\mathcal{O}, \mathcal{I}, \mathcal{F}, \mathcal{N}, \mathcal{Q}, \mathcal{E}, \mathcal{P}\}$. With the superscript $\mathcal{L}^e$, we denote that the feature $\text{eq}(r, r')$, for $\{r, r'\} \subseteq N_R$ is added to the fragment $\mathcal{L}$. Similarly, $\mathcal{L}^d$ corresponds to adding the feature $\text{disj}(r, r')$, also for $\{r, r'\} \subseteq N_R$. In case the fragment $\mathcal{L}$ contains the letter $\mathcal{I}$, $\{r, r'\} \subseteq \overline{N}_R$

Note that adding $\text{closed}(R)$ does not increase the expressivity of $\mathcal{ALC}_S$, when considering document satisfiability. Introducing $\geq_1 E. \varphi$ does increase expressivity of $\mathcal{ALC}_S$ in the supported model semantics, but not in, among others, the least-fixed point semantics (Proposition 4.29).

Lemma 7.3. *For each SHACL document $(\mathcal{C}, \mathcal{G})$ expressible in $\mathcal{ALC}_S$ extended with expressions of the form $\text{closed}(R)$, there exists a constraint set $\mathcal{C}'$ expressible in $\mathcal{ALC}_S$ such that $(\mathcal{C}, \mathcal{G})$ is (finitely) satisfiable iff $(\mathcal{C}', \mathcal{G})$ is (finitely) satisfiable.*

Proof. Since we are in the restricted context of document satisfiability, that is, roles not mentioned in the constraints are irrelevant, we may replace each occurrence of ‘ $\text{closed}(R)$ ’ by ‘ $\bigwedge_{r \in R^c} \neg \exists r. \top$ ’, where $R^c := \{r \in N_R \setminus R \mid r \text{ appears in } \mathcal{C}\}$, to construct $\mathcal{C}'$. ♣

Name	Syntax	Symbol
Nominals	c	$\mathcal{O}$
Inverses	r^-	$\mathcal{I}$
Functionality	$\leq_1 r. \top$	$\mathcal{F}$
Unqualified number restriction	$\geq_n r. \top$	$\mathcal{N}$
Qualified number restriction	$\geq_n r. \varphi$	$\mathcal{Q}$
Unqualified regular path counting	$\geq_n E. \top$	$\mathcal{E}$
Qualified regular path counting	$\geq_n E. \varphi$	$\mathcal{P}$

Table 7.1: Fragments of SHACL following the DL naming convention, extended with counting over regular paths.

7.2 SHACL to OWL and back again

Most of the results in this work are based on the tight connection between SHACL and DLs. In this section, we look at their connection and provide a translation that explains how to translate concepts expressible in $\mathcal{ALCOTQ}$ into SHACL constraints in the fragment $\mathcal{ALCOTQ}_S$, and vice versa. The purpose of this is twofold: first of all it makes all the complexity of satisfiability algorithms and complexity results applicable to the SHACL setting, and moreover it makes joint satisfiability of SHACL in presence of ontologies reducible to DL satisfiability too, as described in the last part of this section.

Translation. First of all, we note that for all logics between $\mathcal{ALC}$ and $\mathcal{ALCOTQ}$, given a TBox $\mathcal{T}$, it is possible to construct a TBox $\mathcal{T}'$ (in polynomial time in $|\mathcal{T}|$) only containing axioms of the form $A \equiv C$, for $A \in N_C$, such that for all concepts $B \in N_C$, we find that B is satisfiable w.r.t. $\mathcal{T}$ iff B' is satisfiable w.r.t. $\mathcal{T}'$, for some fresh $B' \in N_C$. That is, let $\mathcal{T}'$ be defined as follows

$$\mathcal{T}' := \{A_C \equiv \bigcap_{r \in R \cup R^-} \forall r. (A_C \sqcap C) \mid C \in X\} \cup \{B' \equiv B \sqcap \bigcap_{C \in X} A_C\}$$

where $R \subseteq \overline{N}_R$ contains all roles appearing in $\mathcal{T}$, and $X = \{\neg D \sqcup D' \mid D \sqsubseteq D' \in \mathcal{T}\}$.

To see why the equivalence holds, first note that subsumptions like $D \sqsubseteq D'$ are equivalent to $\top \equiv \neg D \sqcup D'$. By using the concepts A_C these equivalences are spread through the graph to all nodes that are reachable by any path over roles in $R \cup R^-$. That is, let $\mathcal{I}'$ be a model of $\mathcal{T}'$ such that some $e \in B^{\mathcal{I}'}$. Let $\mathcal{I}$ be given by restricting the domain of $\mathcal{I}'$ to all nodes reachable by any path over $R \cup R^-$ from e . As all these nodes must satisfy all A_C it follows that $\mathcal{I}$ is a model of $\mathcal{T}$ such that $e \in B^{\mathcal{I}}$. For the other direction, if $\mathcal{I}$ is any model of $\mathcal{T}$, note that for all nodes of $\mathcal{I}$, it holds that $\top \equiv \neg D \sqcup D'$, for all $D \sqsubseteq D' \in \mathcal{T}$, thus, it is immediate that every node can be labelled with all of the A_C .

That is, it will be sufficient to consider axioms of the form $A \equiv C$, for $A \in N_C \setminus \{\top\}$, such that for each $A \in N_C$, there is at most one concept C such that $A \equiv C \in \mathcal{T}$ (note this also follows from the above construction). Recall we set $s \leftarrow \varphi \in \mathcal{C}$ implies $(s)^{\mathcal{I}, S} = (\varphi)^{\mathcal{I}, S}$, which means that this set up aligns well with the supported model semantics of recursive SHACL. In what

$f(c) := c$	$g(c) := c$
$f(A) := s_A$	$g(A) := A, g(s) := A_s$
$f(\top) := \top$	$g(\top) := \top$
$f(\neg C) := \neg f(C)$	$g(\neg \varphi) := \neg g(\varphi)$
$f(C \sqcap C') := f(C) \wedge f(C')$	$g(\varphi \wedge \varphi') := g(\varphi) \sqcap g(\varphi')$
$f(C \sqcup C') := f(C) \vee f(C')$	$g(\varphi \vee \varphi') := g(\varphi) \sqcup g(\varphi')$
$f(\geq_n r.C) := \geq_n r.f(C)$	$g(\geq_n r.\varphi) := \geq_n r.g(\varphi)$
$f(\forall r.C) := \forall r.f(C)$	$g(\forall r.\varphi) := \forall r.g(\varphi)$

Figure 7.1: Translation functions mapping $\mathcal{ALCOIQ}$ concepts into $\mathcal{ALCOIQ}_S$ shape expressions, and vice versa.

follows, we will define two translations and show that the problems of satisfiability in $\mathcal{ALCOIQ}$ and in $\mathcal{ALCOIQ}_S$ are mutually reducible, which lays the basis for extracting satisfiability results from the rich DL literature in the rest of this chapter.

Let us define two translations: f , a function translating any $\mathcal{ALCOIQ}$ concept into a shape expression, and g , a function in the opposite direction, translating any shape expression expressible in $\mathcal{ALCOIQ}_S$ into an $\mathcal{ALCOIQ}$ concept. These functions are recursively defined in Table 7.1, where $s_A \in N_S$ is a fresh shape name introduced for every concept name $A \in N_C$, and A_s is a fresh concept name for each $s \in N_S$. Note that fragments are preserved: an $\mathcal{L}_S$ shape expression translates into an $\mathcal{L}$ concept, and vice versa.

Proposition 7.4. *Let $\mathcal{L}$ be any DL between $\mathcal{ALC}$ and $\mathcal{ALCOIQ}$, then the problems of deciding satisfiability in $\mathcal{L}$ and in $\mathcal{L}_S$ are mutually reducible in polynomial time in size of the input TBox resp. SHACL document.*

Proof. In the following proof, we will consider the reductions for (the SHACL fragment corresponding to) $\mathcal{ALCOIQ}$. For the other fragments $\mathcal{L}$, a simple induction suffices to see that $\mathcal{L}$ concepts are translated into $\mathcal{L}_S$ constraints, and vice versa. Below, we will discuss the different directions separately, and show how to reduce the problems.

For the first case, let $\mathcal{T}$ be an $\mathcal{ALCOIQ}$ TBox such that all axioms are of the form $A \equiv C$, and such there exists at most one axiom of this form for each $A \in N_C$ (see discussion before this proof). Let $\mathcal{G} = \{s_A(c) \mid c \in A^{\mathcal{I}}\}$ and

$$\mathcal{C} = \{s_A \leftarrow f(C) \wedge A \mid A \equiv C \in \mathcal{T}\} \cup \{s_A \leftarrow A \mid A \equiv C \notin \mathcal{T}\}.$$

The idea here is twofold: first, we want to decorate every node that satisfies A with the shape name s_A , and secondly, we want to encode the axioms of the form $A \equiv C$. Note we only want shape names to occur as heads of at most one constraint, leading to the above combination of ideas. Then, $\mathcal{I}$ is a model of $\mathcal{T}$ such that $A^{\mathcal{I}} \neq \emptyset$ iff $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ under the supported model semantics. For the direction from left to right, note that the shape assignment given by $\{s_A(c) \mid c \in A^{\mathcal{I}}\}$ is a supported assignment containing all targets, which can be proven by a straightforward induction that shows that for each $\mathcal{I}$, $c \in C^{\mathcal{I}}$ iff c validates $f(C)$. For the

other direction, the target set already enforces that $\{s_A(c) \mid c \in A^{\mathcal{I}}\}$ is contained in the shape assignment, thus the same induction argument can be used to conclude the required.

For the second case, let $(\mathcal{C}, \mathcal{G})$ be any $\mathcal{ALCOIQ}_S$ SHACL document such that $\mathcal{G}$ only contains shape atoms (Lemma 7.1). Let $\mathcal{I}$ be such that it validates $(\mathcal{C}, \mathcal{G})$ under the supported model semantics. Let S be the shape assignment of $\mathcal{C}$ over the interpretation $\mathcal{I}$ containing all targets in $\mathcal{G}$. We define $\mathcal{I}'$ as follows $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$, for all $A \in N_C \setminus \{A_s \mid s \in N_S\}$, $A^{\mathcal{I}} = A^{\mathcal{I}'}$ and for all $A \in \{A_s \mid s \in N_S\}$, $A_s^{\mathcal{I}'} = \{c \in N_I \mid s(c) \in S\}$. Furthermore, $p^{\mathcal{I}} = p^{\mathcal{I}'}$ for all $p \in N_R$. Here, the idea is to store the shape assignment S in $\mathcal{I}'$ by using the concept names of the form A_s . Then we find that $\mathcal{I}$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{I}'$ is a model of $\{A_s \equiv g(\varphi) \mid s \leftarrow \varphi \in \mathcal{C}\}$, such that if $s(c) \in \mathcal{G}$, then $c \in A_s^{\mathcal{I}'}$. With an induction on the construction of φ , we can show that $c \in g(\varphi)^{\mathcal{I}'}$ iff c validates φ , which corresponds to $s(c) \in S$, for $s \leftarrow \varphi \in \mathcal{C}$. Thus, we find that c validates φ iff $c \in g(\varphi)^{\mathcal{I}'}$ iff $s(c) \in S$ iff $c \in A_s^{\mathcal{I}'}$, where the latter follows by definition of $\mathcal{I}'$. This suffices to conclude this case and thus the both reductions. ♣

The main idea of this proof is that shape and concept names can be considered to be very similar, namely as unary labels for individuals. However, this is specific to the setting of determining (finite) satisfiability under the supported model semantics. To capture different semantics we need other approaches, as is discussed in the next chapter. Also reasoning problems like validation work differently: in that case there is a given interpretation, and the question is whether there exists a shape assignment, whereas in satisfiability, the question is whether there exists an interpretation over which there exists an assignment. The double ‘there exists’ makes this translation to DLs possible.

Joint Satisfiability of SHACL and OWL. In light of the first half of this thesis, we also consider the following reasoning problem, which is asking whether given SHACL documents are satisfiable in presence of ontologies. That is, we are looking for interpretations that are models of the given ontologies, and also validate the given SHACL documents.

(Finite) Joint Document Satisfiability: Given a SHACL document $(\mathcal{C}, \mathcal{G})$ and a set of axioms $\mathcal{T}$, decide whether there exists a (finite) interpretation $\mathcal{I}$ that satisfies $(\mathcal{C}, \mathcal{G})$ under $\mathbb{S}$ and is a model of $\mathcal{T}$.

Given the above presented translation, and equivalence of it follows that the complexity of deciding (finite) joint satisfiability of SHACL under the supported model semantics in presence of OWL corresponds to the complexity of deciding (finite) satisfiability in the least-expressive description logic capturing the expressivity of both the translated SHACL fragment, as the OWL fragment.

7.3 Inverses, Nominals and Counting

The following propositions are well-known results in the Description Logic community. These results extend to our setting, as was observed in the previous section.

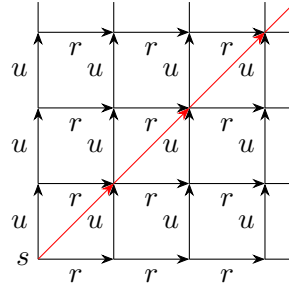


Figure 7.2: Infinite grid that, after adding s_f and s_g as label to every node, shows satisfiability of s . The red diagonal arrows denote the role d .

Proposition 7.5 (for instance [Rud11, LAHS05, BCM⁺03]). *$\mathcal{ALCIO}_S$ and $\mathcal{ALCOQ}_S$ have the finite model property, $\mathcal{ALCIF}_S$ does not, under the supported model semantics.*

Proposition 7.6 ([BCM⁺03, LST05], and their references). *Deciding (finite) satisfiability in $\mathcal{ALC}_S$, $\mathcal{ALCIO}_S$, $\mathcal{ALCOQ}_S$, and $\mathcal{ALCIIQ}_S$ under the supported model semantics is EXPTIME-complete.*

Proposition 7.7. *Deciding (finite) satisfiability in $\mathcal{ALCOIF}_S$ and $\mathcal{ALCOIQ}_S$ under the supported model semantics is NEXPTIME-complete.*

The lower bound for $\mathcal{ALCOIF}_S$ follows from constructing a torus of finite size [Tob00]; the upper bound from translating $\mathcal{ALCOIQ}_S$ into the two-variable fragment of first-order logic with counting quantifiers C^2 [Tob01], in which (finite) satisfiability is NEXPTIME-complete [Pra05].

Proposition 7.8. *$\mathcal{ALCE}_S$ and $\mathcal{ALCP}_S$ do not have the finite model property under the supported model semantics.*

Proof. Consider the following constraints, with the target $s(0, 0)$:

$$\begin{aligned} s &\leftarrow \forall u^+. =_1 r^+ d. \top \wedge \forall (r \cup u)^*. (s_f \wedge s_g) \\ s_f &\leftarrow =_1 u. \top \wedge =_1 r. \top \wedge =_1 (ru \cup ur). \top \\ s_g &\leftarrow \neg \geq_1 d. \top \vee \forall u^+. \neg \geq_1 d. \top \end{aligned}$$

Here, $=_1 \varphi$ is a shorthand for $\geq_1 \varphi \wedge \leq_1 \varphi$. Clearly, a way to satisfy the above constraints is in a simple grid on the natural numbers with a diagonal, where s true in $(0, 0)$ and s_f and s_g validated everywhere. Here the interpretation of d is $\{(i, i), (i + 1, i + 1) \mid i \in \mathbb{N}\}$, for u it is $\{((i, j), (i, j + 1)) \mid \{i, j\} \subseteq \mathbb{N}\}$, and for r the set $\{((i, j), (i + 1, j)) \mid \{i, j\} \subseteq \mathbb{N}\}$.

Assume for contradiction there exists a finite model. As s_f must hold in s and every individual reachable by u , there exists $a_0, \dots, a_i$ such that a_0 is reachable by u^* from $(0, 0)$ and $\{(a_0, a_1), \dots, (a_{i-1}, a_i), (a_i, a_0)\}$ is contained in the interpretation of u . Note that because of having to validate $=_1 r. \top \wedge =_1 (ru \cup ur). \top$ in every individual reachable by r or u , it can be concluded that the set of individuals $\{b_0, \dots, b_j\}$ reachable by r from any individual in $\{a_0, \dots, a_i\}$

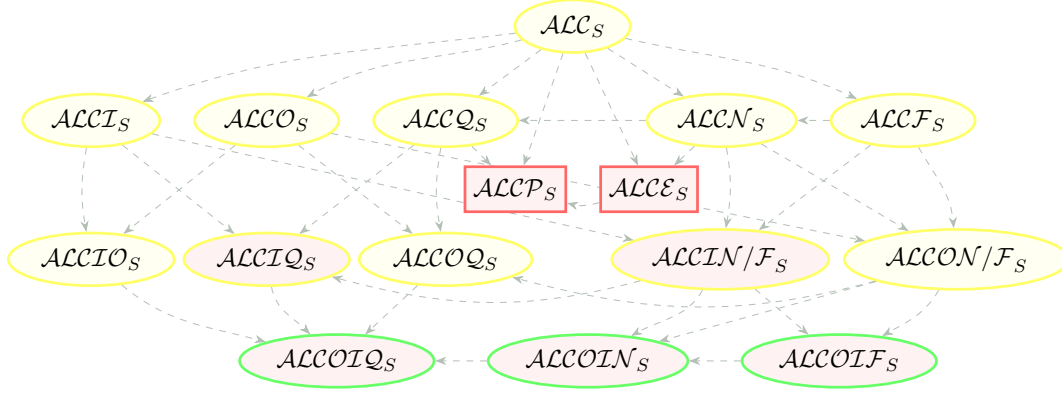


Figure 7.3: Decidability and complexity of SHACL fragments under the supported model semantics. Ellipse-shaped nodes denote (finite) satisfiability is decidable in EXPTIME (yellow border), or NEXPTIME (green border). Squared-shaped nodes indicate satisfiability is undecidable. A yellow filling indicates the presence of the finite model property, whereas a red filling stands for the lack of it. Arrows indicate subsumption of fragments.

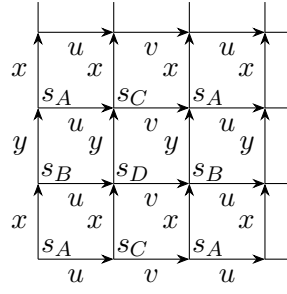
must also contain a loop in the interpretation of u . Clearly, this generalises to: every individual reachable by r^+ from any individual in $\{a_0, \dots, a_i\}$ has a u^+ -path leading to itself. As every individual appearing in a loop of u 's cannot have an outgoing d -edge, because of the constraint $s_g \leftarrow \neg \geq_1 d.\top \vee \forall u^+.\neg \geq_1 d.\top$, it follows that every individual reachable by r^+ from any individual in $\{a_0, \dots, a_i\}$ cannot have an outgoing d -edge. As all individuals in $\{a_0, \dots, a_i\}$ are reachable by u^+ from $(0, 0)$, we cannot validate the first conjunct of s in $(0, 0)$. This is the contradiction which concludes the proof. ♣

Note the above proof produces a grid, which means only a few more rules need to be introduced to reduce the undecidable domino problem [Ber66] to $\mathcal{ALCE}_S$. It is easy to check this is possible, making the satisfiability problem undecidable. This result is already known for different sublogics of $\mathcal{ALCE}_S$, which is discussed in the remainder of this section.

More Fine-Grained Analysis. In the following, we will restrict the expressivity of the regular expressions used in $\geq_n E.\top$ and $\geq_n E.\varphi$. That is, with $\mathcal{ALCN}(X)_S$ or $\mathcal{ALCQ}(X)_S$, for X any combination of the role constructs $*$, $\circ$ and $\cup$, we denote the SHACL fragment allowing regular expressions build from only the role constructs in X in number restrictions. That is, $\mathcal{ALCN}(*, \circ, \cup)_S = \mathcal{ALCE}_S$ and $\mathcal{ALCQ}(*, \circ, \cup)_S = \mathcal{ALCP}_S$. We note that the translation presented in Section 7.2 naturally extends to also capture $*$, $\circ$ and $\cup$ in the number restrictions. Again, we can rely on the vast DL literature: the derived complexity results are the following.

Proposition 7.9. *Satisfiability in $\mathcal{ALCN}(\circ)_S$ under the supported model semantics is EXPTIME-complete.*

This is a direct consequence of satisfiability in the description logic $\mathcal{ALCN}(\circ)$ being decidable in EXPTIME; decidability is shown in Theorem 5 of [BS96], and this algorithm is mentioned to


 Figure 7.4: Infinite grid that can be enforced in $\mathcal{ALCN}(*, \cup)_S$.

be in EXPTIME in the second to last paragraph of this paper. Slight extensions of this DL are already undecidable, as is discussed in Section 3.1 of [BS99].

Proposition 7.10. *Satisfiability in $\mathcal{ALCN}(*, \cup)_S$ under the supported model semantics is undecidable.*

Proof. In this proof, we first show how to enforce the existence of a grid, as is inspired by grid construction in the undecidability proof of unrestricted $\mathcal{SHN}$ in [HST00]. This suffices to proof undecidability, as it is possible to encode a domino tiling problem over this grid in the standard way, for instance like in [BS96].

That is, satisfiability of $s_A(c)$ corresponds to existence of a grid, where the constraints are defined as follows.

$$\begin{aligned} s_A &\leftarrow \neg s_B \wedge \neg s_C \wedge \neg s_D \wedge \exists u.s_B \wedge \exists x.s_C \wedge \leq_3 (u \cup x)^*.\top \\ s_B &\leftarrow \neg s_A \wedge \neg s_C \wedge \neg s_D \wedge \exists v.s_A \wedge \exists x.s_D \wedge \leq_3 (v \cup x)^*.\top \\ s_C &\leftarrow \neg s_A \wedge \neg s_B \wedge \neg s_D \wedge \exists u.s_D \wedge \exists y.s_A \wedge \leq_3 (u \cup y)^*.\top \\ s_D &\leftarrow \neg s_A \wedge \neg s_B \wedge \neg s_C \wedge \exists v.s_C \wedge \exists y.s_B \wedge \leq_3 (v \cup y)^*.\top \end{aligned}$$

The visualisation of the grid enforced by these constraints can be found in Figure 7.4. First, note that the shape names s_A , s_B , s_C and s_D are mutually excluding each other. Secondly, each of them needs exactly two neighbours via specific edges: u and v for the vertical ones, and x and y for the horizontal edges, such that they alternate as is depicted in the figure. This alternation means that any of the $(p \cup q)^*$ makes exactly the nodes (x_1, y) , (x, y_1) and (x_1, y_1) reachable from (x, y) . That is, if the node reachable by $p \circ q$ would not be merged with the node reachable by $q \circ p$, more than 3 nodes would be reachable. Because of the 4 different shapes names that are excluding each other, this is the only option. Thus, satisfiability in $\mathcal{ALCN}(*, \cup)_S$ under the supported model semantics can be reduced to existence of a tiling over an infinite grid, which concludes this proof. ♣

Deciding (finite) satisfiability in $\mathcal{ALCQ}(\cup)_S$ under the supported model semantics is EXPTIME-complete. This result is subsumed by Proposition 7.13 in the next section.

7.4 Equality and Disjointness

Recall we introduced the superscripts $\mathcal{L}^d$ and $\mathcal{L}^e$ to denote the addition of the features $\text{disj}(r, r')$ and $\text{eq}(r, r')$, respectively. Following the naming convention introduced in the previous section, for X any combination of the role constructs $*$, $\circ$ and $\cup$, let $\mathcal{L}(X)^d$, resp. $\mathcal{L}(X)^e$, be the SHACL fragment allowing regular expressions build from only the role constructs in X in the disjointness, resp. equality feature, and in number restrictions, in case $\mathcal{N}$ or $\mathcal{Q}$ is contained in $\mathcal{L}$. That is, recursive SHACL as introduced in the preliminaries, and for satisfiability purposes, corresponds to $\mathcal{ALCCOIQ}(*, \circ, \cup)_S^{d,e}$.

We start with a positive result: adding disjointness does not increase complexity, although the finite model property is easily lost.

Proposition 7.11. *Deciding satisfiability in $\mathcal{ALCI}(*, \circ, \cup)_S^d$ under the supported model semantics is EXPTIME-complete, and this fragment does not have the finite model property. In fact, $\mathcal{ALC}(*, \circ)_S^d$ already lacks this property.*

Proof. We note that this fragment can be expressed in PDL with intersection, for which satisfiability is EXPTIME-complete when the nesting depth of intersection can be bounded (Theorem 4.8 in [GLL09]). To see this, note that $\text{disj}(E, r)$ is equivalent to the expression $\forall(E \cap r). \perp$, which states that for all nodes reachable by both E as r , $\perp$ must hold, or in other words: there are no nodes reachable by both E and r . As the amount of nestings of ‘ $\cap$ ’ in this expression is bounded by 1, the tighter upper bound of EXPTIME can indeed be derived.

For the lack of finite model property, consider the following shapes graph $(\mathcal{C}, \mathcal{G})$:

$$\mathcal{C} = \{s \leftarrow \text{disj}(rr^+, r) \wedge \exists r.s\}$$

and set $\mathcal{G} = \{s(a)\}$. Clearly, the infinite chain of r ’s, in which every individual is labelled with an s is an infinite model. In fact, it must be possible to homomorphically map this chain into any interpretation that validates $(\mathcal{C}, \mathcal{G})$. As $\text{disj}(rr^+, r)$ has to be true in each individual on the chain, it suffices to check that each approach to loop this chain breaks the disjointness. ♣

Even though equality and disjointness might appear to be duals, this belief is quickly crashed: equality is much harder and easily leads to undecidability.

Proposition 7.12. *Deciding satisfiability in $\mathcal{ALC}(\circ)_S^e$ under the supported model semantics is undecidable and $\mathcal{ALC}(*, \circ)_S^e$ does not have the finite model property.*

Proof. The undecidability result directly follows from the undecidability results for Description Logics with role value maps [Sch89], as those directly allow to express role axioms of the form $r_0 \circ \dots \circ r_n = r$. Alternatively, an easy way to also see why the equality feature leads to undecidability is the following constraint set, which encodes a grid.

$$s \leftarrow \text{eq}(ur, d) \wedge \text{eq}(ru, d) \wedge \exists r.s \wedge \exists u.s \wedge \forall r.s \wedge \forall u.s$$

For the lack of finite model property, consider the following SHACL document $(\mathcal{C}, \mathcal{G})$:

$$\mathcal{C} = \{s \leftarrow \text{eq}(r^*, t) \wedge \neg \text{eq}(r^+, t) \wedge \exists r.s\},$$

and set $\mathcal{G} = \{s(a)\}$. Clearly, the infinite chain of r 's, with t the reflexive and transitive closure of r , in which every individual is labelled with an s is an infinite model. In fact, it must be possible to homomorphically map this chain into any interpretation that validates $(\mathcal{C}, \mathcal{G})$. As $\text{eq}(r^*, t) \wedge \neg \text{eq}(r^+, t)$ has to be true in each individual on the chain, it suffices to check that each approach to loop this chain breaks successful validation. ♣

It looks much better when solely allowing ‘ $\cup$ ’ in the equality and disjointness axioms: (finite) satisfiability in $\mathcal{ALC}(\cup)_S^{\text{d,e}}$ is EXPTIME-complete. In fact, this holds for much stronger fragments.

Proposition 7.13. *Deciding satisfiability in $\mathcal{ALCTQ}(\cup)_S^{\text{d,e}}$, and (finite) satisfiability in the fragments $\mathcal{ALCCOQ}(\cup)_S^{\text{d,e}}$ and $\mathcal{ALCTIO}(\cup)_S^{\text{d,e}}$ under the supported model semantics is EXPTIME-complete, and the latter two fragments have the finite model property.*

Proof. Note that for R a union of roles, $\text{eq}(R, r)$ may be reduced to $\forall((R \setminus r) \cup (r \setminus R)).\perp$, where $E \setminus E' := E^{\mathcal{I}} \setminus E'^{\mathcal{I}}$, and $\text{disj}(R, r)$ to $\forall(R \cap r).\perp$. Thus, in case only ‘ $\cup$ ’ is allowed, the equality and disjointness features can be reduced to simple roles expressible in the description logics $\mathcal{ZIQ}$, $\mathcal{ZOO}$, resp. $\mathcal{ZOI}$, which are the description logics $\mathcal{ALCTQ}$, $\mathcal{ALCCOQ}$, resp. $\mathcal{ALCTIO}$ extended with regular expressions over roles, Boolean role expressions and inclusion axioms, and concepts of the form $\exists S.\text{Self}$, where S is built by taking any combination of disjunctions, conjunctions and/or set difference over $\overline{N}_R$. For all these logics, satisfiability is known to be decidable in EXPTIME [CEO09]. Furthermore, $\mathcal{ZOO}$, and $\mathcal{ZOI}$ have the finite model property [BK22]. ♣

We note that the results described in this chapter do not provide a complete picture of all known decidability results in the DL setting.

7.5 Conclusion

We looked at the tight connection between Description Logics and SHACL. In this way, we derived many new complexity results for deciding (finite) satisfiability in SHACL. Specifically, for the general satisfiability problem the picture looks quite complete: as far as we know, only some small fragments remain unclear, like $\mathcal{ALC}(*, \cup)_S^{\text{e}}$, or $\mathcal{ALC}(*)_S^{\text{d,e}}$. However, when looking at finite satisfiability, the status is quite the opposite: a lot of work remains to be done. Specifically in the setting of SHACL, one of the standard tools for managing concrete data sets, the latter case is of uttermost importance.

Another direction for future work is to look at different semantics: in this chapter, we considered (finite) satisfiability under the supported model semantics. However, there are more possibilities to consider: for instance the stable-model, or well-founded semantics. The latter one will be discussed in the next chapter.

7.6 Related Work

Several insights into the working of SHACL have been achieved by examining its tight connection to Description Logics (DLs) [LSR⁺20, BJVdB22b, Ort23, DSŠ24]. Specific to this chapter is extracting the results for satisfiability under the supported model semantics. Of these papers, only Leinberger et al. provide a concrete translation which they use to derive decidability results for containment of single constraints. However, as is pointed out by Bogaerts et al. [BJVdB22b], there are some issues with their translation. Bogaerts et al. [BJVdB22b] and Ortiz [Ort23] focussed on the similarities between SHACL and DLs, but did not provide any concrete translations. Finally, Di Stefano and Šimkus mainly wrote about description logic terminologies, and only described what their results mean for satisfiability of SHACL under the stable model semantics.

It was already known that satisfiability testing for full expressive SHACL is undecidable under the supported model semantics [PKM22]. Pareti et al. [PKM22] also provide some positive results for satisfiability testing of recursive SHACL constraints under the supported model semantics via a translation into decidable fragments of first-order logic. This picture has been extended extensively in this chapter, and none of the results follows from theirs. A tool for testing SHACL satisfiability based on this translation into a decidable fragment of first-order logic is presented by Pareti [Par24].

Static Analysis: Well-Founded Validation

In the previous chapter, we discussed deciding satisfiability under the supported model semantics. However, there are many more interesting static analysis problems, for example *implication* (*a.k.a. containment*), and in this chapter we continue the study of these *static analysis services*.

The task that was the topic of the previous chapter, satisfiability, asks the question to decide whether a given SHACL document is consistent, i.e., whether it validates in at least one data graph. This can be used to test whether different SHACL documents are compatible: it rules out combinations of contradicting constraints.

Providing procedures for deciding satisfiability is also dependent on the semantics considered: deciding whether there exists a validating stable model shape assignment is not the same as deciding the existence of any validating shape assignment. Note however that procedures deciding satisfiability under the supported model semantics provide a bound on the same problem for different semantics: if there is not any shape assignment possible validating the targets, then also not a minimised one as required for the stable model or well-founded semantics.

In this chapter, we will also consider implication. This task asks to decide whether all graphs that validate one document must necessarily also validate another document. This can be helpful when simplifying sets of constraints: it makes it possible to compare which graphs would be validated. Also note that checking implications in both directions allows to decide whether two SHACL documents are equivalent. In general, these services are a stepping stone towards automated transformations and optimisation of SHACL documents.

Naturally, the choice of semantics also matters for deciding implication. In fact, we will first present some undecidability results for deciding implication in SHACL under the stable- and supported model semantics. Afterwards, we move our focus to the well-founded semantics: we uncover its connection to the full hybrid μ -calculus, and derive various complexity results based on this insight.

8.1 Fragment and Normal Form

In this chapter, we will consider shape expressions φ that are build using the following grammar:

$$\varphi ::= s \mid c \mid A \mid \neg s \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall r. \varphi \mid \exists r. \varphi,$$

where $s \in N_S$, $c \in N_I$, $A \in N_C$, and $r \in \overline{N}_R$. Recall the SHACL naming convention introduced in Section 7.1. That is, we will refer to this fragment as $\mathcal{ALC}\mathcal{IO}_S$. If in the above fragment, r is restricted to N_R , that is, we can only access roles in one direction, we are in the fragment $\mathcal{ALCO}_S$. If, on the other hand, the fragment does not use any $c \in N_I$, we are in $\mathcal{ALCI}_S$. If both restrictions apply, this is referred to as $\mathcal{ALC}_S$.

Normal Form. To simplify presentation, we will mostly focus on SHACL constraints in which there is no nesting of constructors. Specifically, we focus on constraints that have one of the following forms:

$$\begin{array}{llll} s \leftarrow A & s \leftarrow a & s \leftarrow \neg s' & s \leftarrow \exists r. s' \\ s \leftarrow \forall r. s' & s \leftarrow s' \wedge s'' & s \leftarrow s' \vee s'' & \end{array}$$

where $A \in N_C$, $a \in N_I$, $r \in \overline{N}_R$, and $\{s, s', s''\} \subseteq N_S$. For all semantics we consider, transforming an arbitrary SHACL document into one where all constraints are in normal form while preserving validation is straightforward. Intuitively, this is because whenever a complex subformula ψ occurs in a shape constraint $s \leftarrow \varphi$, we can simply replace ψ in φ by a fresh shape name s' and add $s' \leftarrow \psi$ to the constraint collection.

8.2 Static Analysis of SHACL Specifications

In this chapter we extend the type of reasoning problems studied compared to Chapter 7. More precisely, we study the following decision problems for the different semantics $\mathbb{S}$, where $\mathbb{S}$ ranges over the least-fixed point, supported model, stable model and well-founded semantics. Note that the first two coincide exactly with the problems studied in the previous chapter.

(Finite) document satisfiability: Given a SHACL document $(\mathcal{C}, \mathcal{G})$, decide whether there exists a (finite) graph that satisfies $(\mathcal{C}, \mathcal{G})$ under $\mathbb{S}$.

Finite model property: Is it the case that, under $\mathbb{S}$, every satisfiable SHACL document is finitely satisfiable?

(Finite) document implication: Given two SHACL documents $(\mathcal{C}_1, \mathcal{G}_1)$ and $(\mathcal{C}_2, \mathcal{G}_2)$, decide whether every (finite) graph that satisfies $(\mathcal{C}_1, \mathcal{G}_1)$ under $\mathbb{S}$ also satisfies $(\mathcal{C}_2, \mathcal{G}_2)$.

We also consider the special cases where we target for validation of one specific shape name.

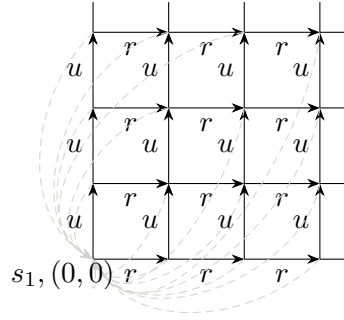


Figure 8.1: Infinite grid that, after adding s as label to every node, shows undecidability of shape implication w.r.t. constraints under the supported model semantics. The grey dashed arrows indicate the extension of the role l .

(Finite) shape satisfiability w.r.t. constraints: Given a set of constraints $\mathcal{C}$ and a shape name s , decide whether there exists a (finite) graph that satisfies $(\mathcal{C}, s)$ under $\mathbb{S}$.

(Finite) shape implication w.r.t. constraints: Given shape constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ and shape names s_1 and s_2 , decide whether every node that satisfies $(\mathcal{C}_1, s_1)$ under $\mathbb{S}$ in a graph $\mathcal{G}$ also satisfies $(\mathcal{C}_2, s_2)$.

First, we recall that considering SHACL in normal form does not affect the above reasoning tasks, as this is a direct consequence of Proposition 4.29.

As we saw in the previous chapter, document satisfiability under the supported model semantics, or equivalently, shape satisfiability w.r.t. constraints, is comparable to satisfiability in the corresponding DL fragment. However, where in DLs, containment and satisfiability are mutually reducible, this is here not the case, as the following theorem illustrates.

The main intuition here is that implication requires another layer of quantification; whereas satisfiability problems only search for existence of a model and a shape assignment, for implication, the question is whether there exists a graph that has a validating shape assignment for the first SHACL document, however, for all shape assignments of the shape names of the second document, we find non-validation.

Theorem 8.1. *Shape implication w.r.t. constraints for (stratified) $\mathcal{ALC}\mathcal{IO}_S$ is undecidable, under the supported model semantics.*

Proof. We prove this by reducing the tiling problem to our setting, a well-known undecidable problem [Ber66]. A tiling problem (D, U, R) asks whether it is possible to decorate a grid with a tiling, such that these tiles are mutually excluding each other and have specific requirements for which domino tiles they can observe to the right and above. Formally, the tiling problem asks whether it is possible to provide a function $t : \mathbb{N} \times \mathbb{N} \rightarrow D$, for $D \subseteq N_S$ such that if $t(x, y) = d$, $t(x, y + 1) = d_u$ and $t(x + 1, y) = d_r$, then $d_u \in U(d)$ and $d_r \in R(d)$.

First, we assume (without loss of generality) that $\mathbb{N} \times \mathbb{N} \subseteq N_I$. Now, let

$$\mathcal{C} = \{s_1 \leftarrow (0, 0) \wedge s, s \leftarrow \exists l. (0, 0) \wedge \exists u. s \wedge \exists r. s \wedge \bigvee_{d \in D} d \wedge \neg s_\perp\} \cup \mathcal{C}_{tiles} \cup \mathcal{C}_{error},$$

where $\mathcal{C}_{tiles}$ contains the constraints ensuring the correct horizontal and vertical relations between the tiles: $d \leftarrow \exists u. \bigvee_{d' \in U(d)} d' \wedge \exists r. \bigvee_{d'' \in R(d)} d''$, for each $d \in D$. Moreover, the error shape $s_\perp$ is assigned in case two or more tiles are assigned to the same constant: $\mathcal{C}_{error} = \{s_\perp \leftarrow \bigvee_{\{d, d'\} \in D, d \neq d'} d \wedge d'\}$. Moreover, we let

$$\mathcal{C}' = \{s_a \leftarrow s_a, s_b \leftarrow s_b, s' \leftarrow (0, 0) \wedge \exists l^-. (\exists r. \exists u. s_a \wedge \exists u. \exists r. s_b) \wedge \forall l^-. (s_a \vee s_b \wedge \neg(s_a \wedge s_b))\}.$$

Claim 8.2. *There exists an interpretation $\mathcal{I}$ such that $\mathcal{I}$ validates $(\mathcal{C}, s_1)$ and not $(\mathcal{C}', s')$ iff there exists a tiling (D, U, R) .*

($\Leftarrow$). Let $\mathcal{I}$ be given by $\Delta^{\mathcal{I}} = \mathbb{N} \times \mathbb{N}$, $r^{\mathcal{I}} = \{((x, y), (x + 1, y)) \mid \{x, y\} \subseteq \mathbb{N}\}$, $u^{\mathcal{I}} = \{((x, y), (x, y + 1)) \mid \{x, y\} \subseteq \mathbb{N}\}$ and $l^{\mathcal{I}} = \{((x, y), (0, 0)) \mid \{x, y\} \subseteq \mathbb{N}\}$. The shape assignment $S = \{s_1((0, 0))\} \cup \{s((x, y)) \mid \{x, y\} \subseteq \mathbb{N}\} \cup \{d((x, y)) \mid t((x, y)) = d\}$ shows that under the supported model semantics we find that $\mathcal{I}$ validates $(\mathcal{C}, s_1)$. Note that because $\mathcal{I}$ is a grid, it is not possible to assign s' to $(0, 0)$: s_a and s_b are disjoint and for every (x, y) the paths ru and ur lead to the same node: $(x + 1, y + 1)$. Thus, $\mathcal{I}$ does not validate $(\mathcal{C}', s')$.

($\Rightarrow$). Let $\mathcal{I}$ be such that $\mathcal{I}$ validates $(\mathcal{C}, s_1)$ and not $(\mathcal{C}', s')$. Without loss of generality, we assume that all nodes are reachable by a path over u and r from $(0, 0)$. Below, we will derive that $\mathcal{I}$ must have a grid structure, that is: for each node, the set of nodes reachable by ru coincides with the set of nodes reachable by ur , and is not empty.

First, note that to satisfy $(\mathcal{C}, s_1)$ it must be possible to map an infinite binary tree (with u and r edges) into $\mathcal{I}$, with every node being connected via l to the root $(0, 0)$. As $\mathcal{I}$ does not validate $\mathcal{I}'$, this means it is not possible to find a shape assignment of s_a, s_b and s' such that $s'(c)$ is contained for at least one $c \in \Delta^{\mathcal{I}}$. As we know $(0, 0) \in \Delta^{\mathcal{I}}$, this means that $\mathcal{I}$ does not validate $\mathcal{I}'$ iff $(0, 0)$ does not validate s' .

So, let us take a closer look at the constraint. It is a conjunction of 3 parts: (i) $(0, 0)$, (ii) $\exists l^-. (\exists r. \exists u. s_a \wedge \exists u. \exists r. s_b)$ and (iii) $\forall l^-. (s_a \vee s_b \wedge \neg(s_a \wedge s_b))$. Clearly, part (i) cannot be the part that is causing non-validation, and the same holds for part (iii): this is expressing that for all nodes in the graph (as they are all reachable by l^-), either s_a or s_b holds, but not both. Note that $s_a \leftarrow s_a$ and $s_b \leftarrow s_b$ can be assigned to any node on any graph under the supported model semantics.

Thus, it must be part (ii) that is causing the non-validation. We claim that part (ii) can only be not validated in case for all nodes we find that the set of nodes reachable by ur coincides with ru and is not empty. Indeed: if there exists a node such that this is not the case, it is possible to find both an $\exists u. \exists r. s_a$ and an $\exists r. \exists u. s_b$ successor, meaning that part (ii) is validated.

So, we can assume that it is possible to map a grid structure to $\mathcal{I}$ such that every node is assigned a tile $d \in D$ ($\mathcal{I}$ validates $(\mathcal{C}, s_1)$). Simply translating the tiles on this grid to a function t suffices to conclude there exists a tiling for (D, U, R) . ♣

Note that it also follows from this proof that there exists a finite $\mathcal{I}$ such that $\mathcal{I}$ validates $(\mathcal{C}, s_1)$ and not $(\mathcal{C}', s')$ iff it is possible to find a recurring tiling (D, U, R) . That is, the problem remains undecidable in the finite setting.

The problem remains undecidable even if we require the set of constraints to be the same in both documents: in the proof, we can replace both $\mathcal{C}$ and $\mathcal{C}'$ by their union. Not surprisingly, the undecidability applies also to the stable semantics, but only in the non-stratified setting.

Theorem 8.3. *Shape implication w.r.t. constraints for $\mathcal{ALC}\mathcal{IO}_S$ is undecidable, under the stable model semantics.*

Proof Sketch. To see this, simply update $\mathcal{C}'$ to $\mathcal{C}' = \{s_a \leftarrow \neg s_b, s_b \leftarrow \neg s_a, s' \leftarrow \exists l^-. (\exists r \exists u. s_a \wedge \exists u. \exists r. s_b)\}$, compared to the previous proof. Here, we use the non-stratifiedness of $\mathcal{C}'$ to enforce the choice of an s_a or s_b label. The rest of the proof works in the same way. ♣

We shift our attention to SHACL containment under the well-founded semantics in the rest of this chapter, and find we can recover decidability of containment. We note it will sometimes be convenient to treat satisfiability as a special case of containment.

Proposition 8.4. *A document $(\mathcal{C}, \mathcal{G})$ in $\mathcal{ALC}\mathcal{IO}_S$ is (finitely) satisfiable under $\mathbb{S}$ iff $(\mathcal{C}, \mathcal{G})$ does not (finitely) imply $(\emptyset, \emptyset)$ under $\mathbb{S}$.*

8.3 Full Hybrid μ -calculus

We first provide some preliminary definitions for the full hybrid μ -calculus,¹ as we will translate SHACL documents into μ -calculus formulas in the rest of this chapter. Let N_V be a set of variables, disjoint from N_I, N_C and N_R . Then, a full hybrid μ -calculus formula is given by

$$\Phi ::= A \mid \neg A \mid a \mid \neg a \mid X \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid [r]\Phi \mid \langle r \rangle \mid \mu X. \Phi \mid \nu X. \Phi,$$

for $A \in N_C, a \in N_I, X \in N_V, r \in \overline{N}_R$. Let $\top := A \vee \neg A$ for a $A \in N_C$, and use ' $\perp$ ' as a shorthand for $\neg \top$.

The set of *subformulas* is defined recursively by setting $sub(C) := \{C\}$ and $sub(\neg C) := \{\neg C\}$ for each $C \in N_I \cup N_C \cup N_V$, $sub(\Phi * \Phi') := \{\Phi * \Phi'\} \cup sub(\Phi) \cup sub(\Phi')$, for each $*$ $\in \{\wedge, \vee\}$, $sub(@\Phi) := \{@\Phi\} \cup sub(\Phi)$ for each $@ \in \{\langle r \rangle, [r] \mid r \in \overline{N}_R\}$, and $sub(\sigma X. \Phi) := \{\sigma X. \Phi\} \cup sub(\Phi)$, for $\sigma \in \{\mu, \nu\}$. We say Ψ appears or is contained in Φ is $\Psi \in sub(\Phi)$.

A variable $X \in N_V$ is a *free variable* if X appears outside of the scope of a fixed point operator σX , for $\sigma \in \{\mu, \nu\}$. A formula is called *closed* if it contains no free variables.

Given an data graph $\mathcal{I}$, the semantics $\|\Phi\|_{\mathcal{I}}^{\mathcal{G}} \subseteq \Delta^{\mathcal{I}}$ of a closed formula Φ is defined as in Figure 8.2. This is exactly the standard Kripke semantics for modal logics, with the addition of fixed point variables needed to compute the fixed points. Here, $V : N_V \rightarrow \mathcal{P}(\Delta^{\mathcal{G}})$ is the function that keeps

¹ μ -calculus with inverse programs and nominals; we follow the naming convention of [BLMV08].

$$\begin{array}{ll}
 \|\Phi \wedge \psi\|_V^{\mathcal{I}} := \|\Phi\|_V^{\mathcal{I}} \cap \|\psi\|_V^{\mathcal{I}} & \|A\|_V^{\mathcal{I}} := A^{\mathcal{I}} \\
 \|\Phi \vee \psi\|_V^{\mathcal{I}} := \|\Phi\|_V^{\mathcal{I}} \cup \|\psi\|_V^{\mathcal{I}} & \|\neg A\|_V^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
 \|[r]\Phi\|_V^{\mathcal{I}} := \{e \in \Delta^{\mathcal{I}} \mid \text{if } (e, e') \in r^{\mathcal{I}}, \text{ then } e' \in \|\Phi\|_V^{\mathcal{I}}\} & \|a\|_V^{\mathcal{I}} := a^{\mathcal{I}} \\
 \|\langle r \rangle \Phi\|_V^{\mathcal{I}} := \{e \in \Delta^{\mathcal{I}} \mid \text{exists } e' \text{ s.t. } (e, e') \in r^{\mathcal{I}}, e' \in \|\Phi\|_V^{\mathcal{I}}\} & \|\neg a\|_V^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus a^{\mathcal{I}} \\
 \|\mu X. \Phi\|_V^{\mathcal{I}} := \bigcap \{U \subseteq \Delta^{\mathcal{I}} : \|\Phi\|_{V[X \rightarrow U]} \subseteq U\} & \|X\|_V^{\mathcal{I}} := V(X) \\
 \|\nu X. \Phi\|_V^{\mathcal{I}} := \bigcup \{U \subseteq \Delta^{\mathcal{I}} : U \subseteq \|\Phi\|_{V[X \rightarrow U]}\} &
 \end{array}$$

 Figure 8.2: Semantics of μ -calculus formulas.

$$\begin{array}{ll}
 \text{cln}(\Phi \wedge \Phi') := \text{cln}(\Phi) \wedge \text{cln}(\Phi') & \\
 \text{cln}(\Phi \vee \Phi') := \text{cln}(\Phi) \vee \text{cln}(\Phi') & \\
 \text{cln}(\langle r \rangle \Phi) := \langle r \rangle \text{cln}(\Phi) & \text{cln}(A) := A \\
 \text{cln}([r]\Phi) := [r]\text{cln}(\Phi) & \text{cln}(a) := a \\
 \text{cln}(\neg \Phi) := \neg \text{cln}(\Phi) & \text{cln}(X) := X \\
 \text{cln}(\sigma X. \Phi) := \begin{cases} \text{cln}(\Phi) & \text{if } X \in \text{sub}(\Phi) \\ \sigma X. \text{cln}(\Phi) & \text{if } X \notin \text{sub}(\Phi) \end{cases} &
 \end{array}$$

Figure 8.3: Cleaning function.

track of those variables, and maps each variable X to a set of nodes. During evaluation, we denote updates of the function V as follows: $V[X \rightarrow U](X) := U$, and $V[X \rightarrow U](X) := V(Y)$ for $X \neq Y$.

For a closed formula Φ , we may use $\mathcal{I}, c \models \Phi$ as a shorthand for $c \in \|\Phi\|_V^{\mathcal{I}}$, for $V(X) = \{\}$ for all $X \in N_V$. Note that because each variable X always occurs in the scope of a fixed point operator σX in a closed formula, it is unnecessary to provide a concrete valuation function V .

Although unusual, we allow fixed point operators $\sigma X. \Psi$ such that X is not contained in Ψ . We call a formula *clean* if there do not appear such fixed point operators in the formula. Note that if X is not contained in Ψ , removing or adding σX does not influence the semantics of the formula. That is, every formula can be *cleaned*, as defined recursively by the function cln , given in Figure 8.3, with the main advantage of improving the readability of the formula. Although we clean the formulas in the translation presented in the next section, we do make use of functions that are not clean when proving the correctness of this translation.

Proposition 8.5. *For each fully hybrid μ -calculus formula, and each data graph $\mathcal{I}$ and node $c \in N_I$, we find*

$$\mathcal{I}, c \models \Phi \quad \text{iff} \quad \mathcal{I}, c \models \text{cln}(\Phi).$$

A well-known classical result is that fixed points can be calculated by iteratively updating the valuation for X with the result of the previous approximation (see, e.g. Arnold and Niwinski [AN01]). That is, the (non-transfinite version of the) approximations of the least and greatest fixed point are given by

$$\begin{aligned} \|\mu^0 X.\Phi\|_V^{\mathcal{I}} &:= \emptyset, \\ \|\mu^{\alpha+1} X.\Phi\|_V^{\mathcal{I}} &:= \|\Phi\|_V^{\mathcal{I}[X \mapsto \|\mu^\alpha X.\Phi\|_V^{\mathcal{I}}]}, \\ \|\mu^\omega X.\Phi\|_V^{\mathcal{I}} &:= \bigcup_{\alpha < \omega} \|\mu^\alpha X.\Phi\|_V^{\mathcal{I}}, \end{aligned}$$

and

$$\begin{aligned} \|\nu^0 X.\Phi\|_V^{\mathcal{I}} &:= \Delta^{\mathcal{G}}, \\ \|\nu^{\alpha+1} X.\Phi\|_V^{\mathcal{I}} &:= \|\Phi\|_V^{\mathcal{I}[X \mapsto \|\nu^\alpha X.\Phi\|_V^{\mathcal{I}}]}, \\ \|\nu^\omega X.\Phi\|_V^{\mathcal{I}} &:= \bigcap_{\alpha < \omega} \|\nu^\alpha X.\Phi\|_V^{\mathcal{I}}. \end{aligned}$$

These approximations lead to the following infinite increasing resp. decreasing chains

$$\begin{aligned} \|\mu^0 X.\Phi\|_V^{\mathcal{I}} &\subseteq \|\mu^1 X.\Phi\|_V^{\mathcal{I}} \subseteq \dots \subseteq \|\mu^\alpha X.\Phi\|_V^{\mathcal{I}} \subseteq \dots \\ \|\nu^0 X.\Phi\|_V^{\mathcal{I}} &\supseteq \|\nu^1 X.\Phi\|_V^{\mathcal{I}} \supseteq \dots \supseteq \|\nu^\alpha X.\Phi\|_V^{\mathcal{I}} \supseteq \dots \end{aligned}$$

Since we only consider graphs with countably many nodes, we can assume $|\Delta^{\mathcal{I}}| \leq \omega$. Thus, we can conclude the following.

Proposition 8.6. *For all full hybrid μ -calculus formulas $\sigma X.\Phi$, and interpretation $\mathcal{I}$ such that $\Delta^{\mathcal{I}} \leq \omega$, we find*

$$\|\sigma^\omega X.\Phi\|_V^{\mathcal{I}} = \|\sigma X.\Phi\|_V^{\mathcal{I}}.$$

8.4 Translating SHACL into μ -calculus

In this section we continue with the actual translation of a SHACL document under the well-founded semantics into full hybrid μ -calculus. More precisely, the goal of this section is to provide a translation that takes as input a SHACL specification $\mathcal{C}$ and target shape name s and produces a μ calculus formula $\Phi_{\mathcal{C},s}$ such that, for every graph $\mathcal{I}$, the nodes that validate s with respect to $\mathcal{C}$ are exactly the nodes that satisfy $\Phi_{\mathcal{C},s}$.

The translation is described in Figure 8.4 and uses two mutually recursive functions $tr_{S,\mathcal{C}}^+$ and $tr_{S,\mathcal{C}}^-$. The desired $\Phi_{\mathcal{C},s}$ will be the (cleaned version of) $tr_{\emptyset,\mathcal{C}}^+(s)$. Recall that the well-founded model is defined as the least fixed point of the operator $W_{\mathcal{A},\mathcal{C}}$ in (3.1); it is thus not surprising that $tr_{\emptyset,\mathcal{C}}^+(s)$ takes the form of a least fixed point $\mu X \varphi$. The positive part $tr_{S,\mathcal{C}}^+$ will mimic the derivation of the positive atoms that are introduced by $T_{\mathcal{A},\mathcal{C}}(S)$ during the well-founded model computation, while the $tr_{S,\mathcal{C}}^-$ part will be the counterpart of the negative atoms

$$\begin{array}{ll}
 tr_{S,C}^+(s \wedge s') := tr_{S,C}^+(s) \wedge tr_{S,C}^+(s') & tr_{S,C}^-(s \wedge s') := tr_{S,C}^-(s) \vee tr_{S,C}^-(s') \\
 tr_{S,C}^+(s \vee s') := tr_{S,C}^+(s) \vee tr_{S,C}^+(s') & tr_{S,C}^-(s \vee s') := tr_{S,C}^-(s) \wedge tr_{S,C}^-(s') \\
 tr_{S,C}^+(\exists r.s) := \langle r \rangle tr_{S,C}^+(s) & tr_{S,C}^-(\exists r.s) := [r] tr_{S,C}^-(s) \\
 tr_{S,C}^+(\forall r.s) := [r] tr_{S,C}^+(s) & tr_{S,C}^-(\forall r.s) := \langle r \rangle tr_{S,C}^-(s) \\
 tr_{S,C}^+(A) := A & tr_{S,C}^-(A) := \neg A \\
 tr_{S,C}^+(a) := a & tr_{S,C}^-(a) := \neg a \\
 tr_{S,C}^+(\neg s) := tr_{S,C}^-(s) & tr_{S,C}^-(\neg s) := tr_{pos(S),C}^+(s) \\
 tr_{S,C}^+(s) := \begin{cases} X_s & \text{if } s \in S \\ \mu X_s. tr_{S \cup \{s\},C}^+(C(s)) & \text{if } s \notin S \end{cases} & tr_{S,C}^-(s) := \begin{cases} X_{\bar{s}} & \text{if } \bar{s} \in S \\ \nu X_{\bar{s}}. tr_{S \cup \{\bar{s}\},C}^-(C(s)) & \text{if } \bar{s} \notin S \end{cases}
 \end{array}$$

Figure 8.4: Translation functions, with emphasis on the negated translation of a negation, $tr_{S,C}^+(\neg s)$: the well-founded semantics does not treat positive and negative information equally, which is captured by only keeping the positive shape names in the set S : $pos(S) := S \setminus \{\bar{s} \in S\}$.

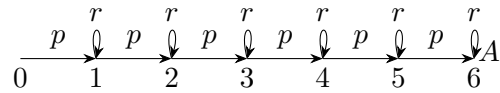
added by $\neg.U_{A,C}(S)$. It is this second part that is more interesting. Recall that $U_{A,C}(S)$ is the *greatest* unfounded set, that is, the largest set of atoms that do not break a certain condition ($c \notin [\mathcal{C}(s)]_{S \cup \neg.U}^A$ for all $s(c) \in U$). This seems to naturally correspond to a greatest fixed point, but we need a bit more than just a ν : note that when deciding whether $b \in [\bar{s}]_{S \cup \neg.U}^G$, we do not just check whether $\neg s(b) \notin \neg.U$, but also whether $\neg s(b) \notin S$, that is, the extension of the set is affected not only by the set U that is being computed as a greatest fixed point, but also by the set S that are computed by the least fixed point corresponding to $W_{A,C}(S)$. Therefore, our translation produces a formula with limited alternation: it takes the form $\mu X.\varphi(X)$, where $\varphi(X)$ may contain subformulas of the form $\nu Y.\psi(X, Y)$. There are, however, no fixed points nested within greatest fixed points: observe that the subscript S is changed to $pos(S)$ when we move to the negative part $tr_{S,C}^-$ of the translation.

In the following example, we illustrate that this limited alternation is needed, as the alternation-free μ -calculus is not expressive enough to capture the well-founded semantics.

Example 8.7. Recall the constraints $\mathcal{C} = \{s \leftarrow A \vee \exists p.\neg t, t \leftarrow \neg s \vee \exists r.t\}$ discussed in Example 3.7. Then

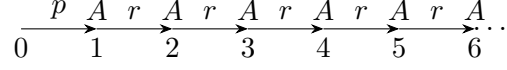
$$tr_{\emptyset,C}^+(s) = \mu X_s.(A \vee \langle p \rangle (\nu X_{\bar{t}}.(X_s \wedge [r]X_{\bar{t}}))).$$

Note that if we evaluate $tr_{\emptyset,C}^+(s)$ over the following structure $\mathcal{I}_1$



we find $\|tr_{\emptyset,C}^+(s)\|_{\mathcal{I}_1}^Z = \{0, \dots, 6\}$.

To see this is a formula expressing a property not expressible in alternation free μ -calculus, note that we also find $0 \in ||tr_{\emptyset, \mathcal{C}}^+(s)||_{\mathcal{I}_2}^{\mathcal{I}_2}$ for the following (infinite) structure $\mathcal{I}_2$.



When considering the alternating tree automaton (see Section 8.5) corresponding to this formula, note that we need both the parities 0 and 1 to accommodate that the state corresponding to μX_s is observed infinitely deep in the accepted run tree, however, at most finitely many times on the same branch. Alternating tree automata that use both the parities 0 and 1 fall into the Σ_2 fragment of the hierarchy for alternating tree automata as described in [Alb01]. Here, it is also shown that the μ -calculus fragment that exactly captures this fragment is strictly more expressive than the alternating free μ -calculus. $\heartsuit$

Note that the constraint definitions that are not reachable from a target are irrelevant to its validation, and hence to test s we can just use $\mathcal{C}_s$, defined in the following way.

Definition 8.8. Given a constraint set $\mathcal{C}$, let $cl_{\mathcal{C}}(s)$, the closure of a shape name s w.r.t. $\mathcal{C}$, be defined as the smallest set of shape names $s' \in N_S$ such that $s \in cl_{\mathcal{C}}(s)$, and furthermore, if $s' \in cl_{\mathcal{C}}(s)$ and $s' \leftarrow \varphi \in \mathcal{C}$ such that $s'' \in sub(\varphi)$, then $s'' \in cl_{\mathcal{C}}(s)$. Let the restriction of a constraint set $\mathcal{C}$ to a shape name s be the following constraint set:

$$\mathcal{C}_s := \{s' \leftarrow \varphi \in \mathcal{C} \mid s' \in cl_{\mathcal{C}}(s)\}.$$

We call these constraints reachable from s .

Recall that also $s' \in sub(\neg s')$. Thus, the following must hold.

Lemma 8.9. For each constraint set $\mathcal{C}$ and interpretation $\mathcal{I}$, we have $\mathcal{I}$ validates $(\mathcal{C}, s(a))$ iff $\mathcal{I}$ validates $(\mathcal{C}_s, s(a))$.

Without loss of generality, the proofs below assume that all constraints in $\mathcal{C}$ are reachable from s . We first establish an assisting lemma on the structure of the constructed modal μ -calculus formulas.

Lemma 8.10. Given $\nu X.\varphi \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$ such that $X \in sub(\varphi)$, then there is no $\mu X.\psi \in sub(\nu X.\varphi)$ such that $X \in sub(\psi)$.

Proof Sketch. Assuming the contrary quickly leads to the conclusion that in that case X would not have been introduced; only the positive shape names remain in the rule $tr_{\bar{S}, \mathcal{C}}^-(\neg s) := tr_{pos(S), \mathcal{C}}^+(s)$. $\clubsuit$

Given an interpretation $\mathcal{I}$, let $atoms_{\mathcal{I}}$ be a (partial) function that maps (approximations of) modal mu-calculus formulas to sets of atoms, defined in the following way

$$\begin{aligned} atoms_{\mathcal{I}}(\mu^i X_s.\varphi) &:= \{s(a) \mid a \in \|\mu^i X_s.\varphi\|_{\mathcal{V}}^{\mathcal{I}}\} \\ &\cup \{s'(a) \mid \mu X_{s'}.\psi \in sub(\varphi), a \in \|\mu X_{s'}.\psi\|_{\mathcal{V}}^{\mathcal{I}}\} \\ &\cup \{\neg s'(a) \mid \nu X_{\bar{s}}.\psi \in sub(\varphi), a \in \|\nu X_{\bar{s}}.\psi\|_{\mathcal{V}}^{\mathcal{I}}\}. \end{aligned}$$

We use the notation S_j as a shorthand for $W_{\mathcal{I},\mathcal{C}}^{\uparrow j}(\emptyset)$, whenever $\mathcal{I}$ and $\mathcal{C}$ are understood from the context.

In what follows, we first prove two propositions that will form the building blocks for the final proof of correctness of the presented translation. The main idea is that if we can prove that for each step in the least fixed point computations on both sides, the result will be included at some point on the other side too, this suffices; both are least fixed point computations, meaning that at some point a fixed point is reached. This is a way to handle the fact that the cycles on both sides in which new information is added have their own speed and do not per se align. Here on the one side there is the direct consequence operator defining the well-founded semantics, and on the other side the approximation functions for the modal μ -calculus. In the following we carefully consider new facts derived by approximating the least fixed points one step further, and explore how the same derivation is handled on the other side.

Proposition 8.11. *For each constraint set $\mathcal{C}$ in $\mathcal{ALCIO}_S$, each shape name s , each interpretation $\mathcal{I}$ and each natural number i , if $t'(b) \in W_{\mathcal{I},\mathcal{C}_s}^{\uparrow i}(\emptyset)$, for $t' = s'$, or $t' = \neg s'$, then there exists a j such that $t'(b) \in atoms_{\mathcal{I}}(\mu^j X_s.\varphi)$, for $\mu X_s.\varphi = tr_{\emptyset,\mathcal{C}}^+(s)$.*

Proof. Note that because we consider $\mathcal{C}_s$, we know there exists a subformula of the form $\{\mu X_{s''}.\psi, \nu X_{\bar{s}''}.\psi\} \cap sub(tr_{\emptyset,\mathcal{C}}^+(s)) \neq \emptyset$ for each $s'' \leftarrow \varphi \in \mathcal{C}_s$. Moreover, as the function $tr_{\emptyset,\mathcal{C}}^+$ is alternating between the positive (only introducing subformulas of the form $\mu X.\psi$) and negative version, $tr_{\mathcal{S},\mathcal{C}}^-$ (only introducing $\nu X.\psi$ -subformulas) exactly when some $s \leftarrow \neg s'$ shows up, we must also conclude that if $s''(b) \in S_i$ for some i , there must be $\mu X_{s''}.\psi \in sub(tr_{\emptyset,\mathcal{C}}^+(s))$, and similarly with $\nu X_{\bar{s}''}.$ for $\neg s''(b) \in S_i$. We prove this proposition by induction on i . For the induction basis, note $W_{\mathcal{I},\mathcal{C}_s}^{\uparrow 0}(\emptyset) = S_0 = \emptyset$. For the induction step, assume there exists some $t(b) \in S_i \setminus S_{i-1}$. We distinguish two cases: (i) $t = s'$ and (ii) $t = \neg s'$.

(i) Let $s' \leftarrow \varphi \in \mathcal{C}$ be the unique constraint with head s' . We distinguish the following cases.

- $\varphi = A$. Note that $s'(b) \in S_i$ implies that $b \in A^{\mathcal{I}}$. Furthermore, we find there exists some $\mu X_{s'}.\psi \in sub(tr_{\emptyset,\mathcal{C}}^+(s))$. Note that $\psi = tr_{S \cup s',\mathcal{C}_s}^+(A) = A$, independent of what is contained in S . As $\|\mu X_{s'}.\psi\|_{\mathcal{V}}^{\mathcal{I}} = A^{\mathcal{I}}$, we find $s'(b) \in atoms_{\mathcal{I}}(\mu^j X_s.\varphi)$ for each $j \geq 1$.
- $\varphi = \neg s''$. Note that if $s'(b) \in W_{\mathcal{I},\mathcal{C}_s}^{\uparrow i}(\emptyset)$, then $\neg s''(b) \in W_{\mathcal{I},\mathcal{C}_s}^{\uparrow i-1}(\emptyset)$, thus, by using the induction hypothesis, we find that there exists a j such that $\neg s''(b) \in$

$atoms_{\mathcal{I}}(\mu^j X_s.\varphi)$. Following the definition of $atoms_{\mathcal{I}}$, this means there must be a $\nu X_{s''}.\psi \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$ such that $b \in \|\nu X_{s''}.\psi\|_{V_j}^{\mathcal{I}}$, where V_j is such that $V_j(X_s) = \|\mu^{j-1} X_s.\varphi\|_{V_j}^{\mathcal{I}}$. Considering the definition of $tr_{\emptyset, \mathcal{C}}^+(s)$, we distinguish two options: (a) either we also find the subformula $\mu X_{s'}.\nu X_{s''}.\psi \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$, or (b), only the subformula $\mu X_{s'}.\psi \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$. As (b) is in contradiction with Lemma 8.10, we only consider case (a). If $X_{s'} \in sub(\psi)$, this means there is some subformula $\mu X_{s'}.\chi(X_{s'}) \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$ for which we can use a similar proof strategy using the least-fixed point approximation. If $X_{s'} \notin sub(\psi)$, this means that $\|\mu X_{s'}.\nu X_{s''}.\psi\|_{V_j}^{\mathcal{I}} = \|\nu X_{s''}.\psi\|_{V_j}^{\mathcal{I}}$, and thus $s'(b) \in atoms_{\mathcal{I}}(\mu^j X_s.\varphi)$.

- Other cases are treated similarly.

- (ii) Let U be the greatest unfounded set w.r.t. S_{i-1} , $\mathcal{I}$ and $\mathcal{C}_s$, we distinguish four cases: (a) $s'(b) \in U$ such that $s' \leftarrow A \in \mathcal{C}$ is the unique constraint with head s' ; (b) $s'(b) \in U$ such that $s' \leftarrow \neg s'' \in \mathcal{C}$ is the unique constraint with head s' ; (c) $s'(b) \in U$ such that $s' \leftarrow \exists r.s'' \in \mathcal{C}$ is the unique constraint with head s' and $\{e \in \Delta^{\mathcal{I}} \mid (b, e) \in r^{\mathcal{I}}\} = \emptyset$; and (d) all $s'(b) \in U$ that do not fit in any of the previous categories. Note that cases (a) and (c) can be treated similarly to the cases $\varphi = A$, resp. $\varphi = \forall r.s''$ in part (i). Thus, we will only address cases (b) and (d) here.

- (b) $\varphi = \neg s''$. Note that $[\varphi]_{S_{i-1} \cup \neg.U}^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid s''(a) \notin S_{i-1} \cup \neg.U\}$. Thus, $b \notin [\varphi]_{S_{i-1} \cup \neg.U}^{\mathcal{I}}$ implies that $s''(b) \in S_{i-1}$. Now, we may use the induction hypothesis to conclude there exists some j such that $s''(b) \in atoms_{\mathcal{I}}(tr_{\emptyset, \mathcal{C}}^+(s))$. The rest of the argument can be treated similar to the cases in (i).

- (d) Take some $s'(b) \in U$ and note that there must exist some $\nu X_{s'}.\psi \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$. Now let $L = \{a \in \Delta^{\mathcal{I}} \mid s'(a) \in U\}$, we aim to show that $L \subseteq \|\nu X_{s'}.\psi\|_{V[X_{s'} \mapsto L]}^{\mathcal{I}}$. We distinguish two cases: (α) either $X_{s'} \in sub(\psi)$, or (β) not. In the latter case, we may distinguish two settings again, either there exists some $\nu X_{s''}.\psi' \in sub(tr_{\emptyset, \mathcal{C}}^+(s))$ such that $\nu X_{s'}.\psi \in sub(\psi')$ and $X_{s''} \in sub(\psi)$, that is, $\nu X_{s'}.\psi$ appears within some “loop” of negated atoms, or not. The case of $\nu X_{s'}.\psi$ in the “loop” will follow from the way case (α) is considered, whereas the outside-of-the-loop-case will either fall directly in cases (a)-(c), or can be brought back to cases of the form (a)-(c) with techniques similar to the ones in (i).

That is, we solely focus on case (α) here; if $X_{s'} \in sub(\psi)$, there exist one or more chains of axioms $C_i = \{s_{i_1} \leftarrow \varphi_{i_1}, \dots, s_{i_n} \leftarrow \varphi_{i_n}\}$ such that $s_{i_1} = s'$, each φ_{i_j} is of one of the forms $s'' \wedge s'''$, $s'' \vee s'''$, $\exists r.s''$ or $\forall r.s''$, for each $1 \leq j \leq n-1$ we have $s_{i_{j+1}} \in sub(\varphi_{i_j})$, and lastly, we find $s_{i_1} \in sub(\varphi_{i_n})$. To avoid redundancy, we also assume $s' \neq s_{i_j}$ for each $1 < j \leq n$. Note that by combining the insights of Lemma 8.10 with the definition of the translation function, we find that for each $1 < j \leq n$, there exists some $\nu X_{s_{i_j}}.\psi_{i_j} \in sub(\psi)$ such that $\nu X_{s_{i_j}}.\psi_{i_j} \in sub(\psi_{i_{j-1}})$.

Now let U' be the greatest subset of U such that if $s(a) \in U'$, then there exists some chain of axioms C_i such that $s = s_{i_j}$ for some $1 \leq j \leq n$, and moreover, for all $1 \leq j \leq n$, if $s_{i_j}(a) \in U'$ and $s_{i_j} \leftarrow \varphi_{i_j} \in \mathcal{C}$, then

- * if $\varphi_{i_j} \in \{s_{i_{j+1}} \wedge s', s' \wedge s_{i_{j+1}}, s_{i_{j+1}} \vee s', s' \vee s_{i_{j+1}}\}$, then $s_{i_{j+1}}(a) \in U'$; and
- * if $\varphi_{i_j} \in \{\exists r.s_{i_{j+1}}, \forall r.s_{i_{j+1}}\}$, then there exists some $b \in \{b \in \Delta^{\mathcal{I}} \mid (a, b) \in r^{\mathcal{I}}\}$ such that $s_{i_{j+1}}(b) \in U'$,

where $s_{i_{n+1}}$ is used as a substitute name for s_{i_1} .

For each chain of axioms C_i , let $L_{i_j} = \{a \in \Delta^{\mathcal{I}} \mid s_{i_j}(a) \in U'\}$. As $s_{i_1} = s'$ for each chain of axioms C_i , we find $L_{i_1} = L_{k_1}$ for each pair of axiom chains C_i and C_k . We want to show that

$$L_{i_j} \subseteq \|\nu X_{s_{i_j}^-}.\psi_{i_j}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}} \quad (8.1)$$

for each constraint $s_{i_j} \leftarrow \varphi_{i_j}$ appearing in one of the axiom chains C_i , assuming that for each $s_{i_j} \leftarrow \varphi$ with $\varphi \in \{s_{i_{j+1}} \vee s', s' \vee s_{i_{j+1}}\}$, $\{a \in \Delta^{\mathcal{I}} \mid s_{i_j}(a) \in U'\} \subseteq \|\nu X_{s_{i_j}^-}.\psi'\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$, where $V(X_{s_{i_1}^-}) = L_{k_1}$ in case $s' = s_{k_l}$, that is, s' appears in some other chain, or where V is in the state it is in after approximating the previous layer in case s' does not appear in any other chain. To see (8.1) holds, note it is immediate that $L_{i_1} \subseteq \|\nu X_{s_{i_1}^-}.\psi_{i_1}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$. Now assume $L_{i_j} \subseteq \|\nu X_{s_{i_j}^-}.\psi_{i_j}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$, and let $s_{i_{j-1}}(b) \in L_{i_{j-1}}$. We distinguish four cases: conjunction, disjunction, and existential and universal quantification. To illustrate the main idea, we treat the conjunction case.

- * Let $s_{i_{j-1}} \leftarrow s_{i_j} \wedge s' \in C_i$ (or similarly $s_{i_{j-1}} \leftarrow s' \wedge s_{i_j} \in C_i$). In this case $\psi_{i_{j-1}} = \nu X_{s_{i_j}^-}.\psi_{i_j} \vee \chi$, for some χ representing the ν formula for s' not included in the discussed loop, and thus $\|\nu X_{s_{i_j}^-}.\psi_{i_j}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}} \cup \|\chi\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}} = \|\psi_{i_{j-1}}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$. As $X_{s_{i_{j-1}}^-} \notin \text{sub}(\psi_{i_{j-1}})$, we find $\|\psi_{i_{j-1}}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}} = \|\nu X_{s_{i_{j-1}}^-}.\psi_{i_{j-1}}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$. Moreover, we find for each $a \in L_{i_{j-1}}$, we have $a \in L_{i_j}$, or $a \in \|\chi\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$; suppose $a \in L_{i_{j-1}}$, then $a \notin [s_{i_j} \wedge s']_{S \cup \neg, U'}^{\mathcal{G}}$, which means $a \notin [s_{i_j}]_{S \cup \neg, U'}^{\mathcal{G}}$ or $[s_{i_j} \wedge s']_{S \cup \neg, U'}^{\mathcal{G}}$. In the first case, note that $[s_{i_j}]_{S \cup \neg, U'}^{\mathcal{G}} = \{c \in \Delta^{\mathcal{G}} \mid \neg s_{i_j} \notin S \cup \neg, U'\}$, thus $s_{i_j}(a) \notin U'$, which means that $a \in L_{i_j}$. In the second case, $a \notin [s']_{S \cup \neg, U'}^{\mathcal{G}}$, there are again two options: either $s' = s_{k_l}$, in which case a similar argument for the loop k suffices to show $a \in L_{k_l} \subseteq \|\chi\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$; or s' does not appear in any loop. In particular, this means there are no atoms of the form $s'(b) \in U'$, by definition of U' . If we now look at $a \notin [s']_{S \cup \neg, U'}^{\mathcal{G}}$, this implies $s'(a) \in U'$, or $\neg s'(a) \in S$. Thus, $\neg s'(a) \in S$. So, we are back in a non-loop case and can use the reasoning of before to conclude: $a \in \|\chi\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$. Combining all subsumptions and equalities leads to the conclusion that $L_{i_{j-1}} \subseteq \|\nu X_{s_{i_{j-1}}^-}.\psi_{i_{j-1}}\|_{V[X_{s_{i_1}^-} \mapsto L_{i_1}]}^{\mathcal{I}}$, as required.

♣

In the following proposition, the main change is the inversion of direction: we also need to show that when the approximation function of the μ -calculus formula adds new nodes to its extension, this is reflected in the least fixed operator defining the well founded semantics.

Recall that the set of atoms of a μ -calculus formula not just outputs the atoms of the form $s(b)$, where s corresponds to the outermost fixed point variable X_s ; to have a smoother induction proof, we also extract the shape atoms for the other shape names. In particular, those of the recursive cycle that makes s refer to itself.

Proposition 8.12. *For each constraint set $\mathcal{C}$ in $\mathcal{ALC}\mathcal{IO}_S$, each interpretation $\mathcal{I}$ and each natural number i , such that $\mu X_s.\varphi = \text{tr}_{\emptyset, \mathcal{C}}^+(s)$ if $s'(b) \in \text{atoms}_{\mathcal{I}}(\mu^i X_s.\varphi)$, then there exists j such that $s'(b) \in W_{\mathcal{I}, \mathcal{C}}^{\uparrow j}(\emptyset)$.*

Proof. As we are considering a least fixed point, it suffices to prove the following claim.

Claim 8.13. *Given some $\sigma X_t.\varphi$ and a valuation function V , and assume that for each variable $X_{t'} \in \text{sub}(\varphi)$ that is free in $\sigma X_t.\varphi$, we have, if $b \in V(X_{t'})$, then there exists a j such that $b \in [t']_{S_j}^{\mathcal{I}}$. In that case, there exists a j' such that $\|\sigma X_t.\varphi\|_V^{\mathcal{I}} \subseteq [t]_{S_{j'}}^{\mathcal{I}}$.*

We prove the above claim by induction on the construction of φ . We distinguish three cases: (i) $\sigma X.\varphi$ such that $X \notin \text{sub}(\varphi)$, and in case $X \in \text{sub}(\varphi)$: (ii) $\sigma = \mu$ and (iii) $\sigma = \nu$.

- (i) - $\varphi = A$. Clearly $\|\sigma X_t.A\|_V^{\mathcal{I}} = \|A\|_V^{\mathcal{I}} = A^{\mathcal{I}} = [A]_{S_j}^{\mathcal{I}}$, as $\sigma X_t.A$ can only be constructed in case $t \leftarrow A \in \mathcal{C}$, we find that if $b \in \|\sigma X_t.A\|_V^{\mathcal{I}}$, then $b \in [t]_{S_{j+1}}^{\mathcal{I}}$.
- $\sigma X_t.\varphi = \mu X_t.\nu X_{\bar{s}}.\psi$. As $X_t \notin \text{sub}(\psi)$, we find $\|\mu X_t.\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}} = \|\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}}$, then by IH, there exists some j such that $\|\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}} \subseteq [\neg s]_{S_j}^{\mathcal{I}}$. As $\mu X_t.\nu X_{\bar{s}}.\psi$ will only be constructed given some $t \leftarrow \neg s \in \mathcal{C}$, we find that if $b \in \|\mu X_t.\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}}$, then $b \in [t]_{S_{j+1}}^{\mathcal{I}}$.
- Other cases are treated similarly.
- (ii) - $\mu X_t.\varphi(X_t) = \mu X_t.\sigma X_s.\psi(X_t)$. In this case, we find $\|\mu X_t.\sigma X_s.\psi(X_t)\|_V^{\mathcal{I}}$ can be determined by using the approximation function $\|\mu^0 X_t.\sigma X_s.\psi(X_t)\|_V^{\mathcal{I}} = \emptyset$ and $\|\mu^{i+1} X_t.\sigma X_s.\psi(X_t)\|_V^{\mathcal{I}} = \|\sigma X_s.\psi(X_t)\|_{V[X_t \mapsto \|\mu^i X_t.\sigma X_s.\psi(X_t)\|_V^{\mathcal{I}}]}^{\mathcal{I}}$. Note that for the i -th approximated case, we can derive that there exists some j such that $\|\mu^i X_t.\sigma X_s.\psi(X_t)\|_V^{\mathcal{I}} \subseteq [t]_{S_j}^{\mathcal{I}}$ by a repetitive usage of the induction hypothesis. As we are considering a least fixed point, this eventually suffices.
- Other cases are treated similarly.
- (iii) $\nu X_t.\varphi(X_t)$. Following Lemma 8.10, all $\sigma X_{t'}.\psi \in \text{sub}(\varphi)$ such that $X_t \in \text{sub}(\psi)$ are such that $\sigma = \nu$. We collect all relevant shape names in the following set: let $W = \{s \in N_S \mid \nu X_{\bar{s}}.\psi \in \text{sub}(\varphi), X_t \in \text{sub}(\psi)\}$. Note that for all $\nu X_{\bar{s}}.\psi \in \text{sub}(\varphi)$ such that $X_t \notin \text{sub}(\psi)$ the induction hypothesis may be applied to derive that there exists some j such that $\|\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}} \subseteq [\neg s]_{S_j}^{\mathcal{I}}$. Let j' be the maximum of all these j . Now let $U = \{s(a) \mid a \in \|\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}}, \nu X_{\bar{s}}.\psi \in \text{sub}(\nu X_t.\varphi), s \in W\}$.

In what follows, we will show U is an unfounded set w.r.t. $S_{j'}, \mathcal{I}$ and $\mathcal{C}$. To this end, we have to show that for all $s(a) \in U$ we find $a \notin [\mathcal{C}(s)]_{S_{j'} \cup \neg U}^{\mathcal{I}}$. Note that if $s \leftarrow \varphi' \in \mathcal{C}$

such that $s \in W$, then φ' is of the form $\exists r.s', \forall r.s', s \leftarrow s' \wedge s''$ or $s \leftarrow s' \vee s''$. Thus, we may consider $\mathcal{C}(s) \in \{\exists r.s', \forall r.s', s' \wedge s'', s' \vee s''\}$:

- $\mathcal{C}(s) = \forall r.s'$. In this case, the subformula looks like $\nu X_{\bar{s}}.\langle r \rangle \nu X_{\bar{s}'}.\psi'$, or $\nu X_{\bar{s}}.\langle r \rangle X_{s'}$. Note that in the latter case, $s' = t$. As we assumed $a \in \|\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}}$, we find there must be a $b \in \Delta^{\mathcal{I}}$ such that $(a, b) \in r^{\mathcal{I}}$ and $b \in \|\nu X_{\bar{s}'}.\psi'\|_V^{\mathcal{I}}$, both if $\psi'' = \psi'$ and if $\psi'' = \varphi$. Thus, $s'(b) \in U$, which means $s'(b) \notin \neg.U$ and moreover, we find $s'(b) \notin S_{j'}$ (I.H. of the previous proposition), thus $b \notin [s']_{S_{j'} \cup \neg.U}^{\mathcal{I}}$. As $[\forall r.s']_{S_{j'}}^{\mathcal{I}} = \{a' \in \Delta^{\mathcal{I}} \mid \forall a' \in \Delta^{\mathcal{I}} : (a, a') \in r^{\mathcal{I}} \text{ implies } a' \in [s']_{S_{j'} \cup \neg.U}^{\mathcal{I}}\}$, we must conclude that, because of the witness b , $a \notin [\mathcal{C}(s)]_{S_{j'} \cup \neg.U}^{\mathcal{I}}$.
- $\mathcal{C}(s) = s' \vee s''$. In this case, the subformula looks like $\nu X_{\bar{s}}.\psi' \wedge \psi''$, for ψ' and ψ'' either a variable Y or a subformula of the form $\nu Y.\chi$. There are two possible options: either $\{s', s''\} \subseteq W$, in which case a similar argument as before suffices. However, if, without loss of generality, $s' \notin X$, the subformula looks like $\nu X_{\bar{s}}.(\nu X_{\bar{s}'}.\psi') \wedge \psi''$ we are back in the case of $\nu X_{\bar{s}'}.\psi \in \text{sub}(\varphi)$ such that $X_t \notin \text{sub}(\psi)$, and thus $\|\nu X_{\bar{s}'}.\psi\|_V^{\mathcal{I}} \subseteq [\neg s']_{S_{j'}}^{\mathcal{I}}$, as concluded before. As we assumed $a \in \|\nu X_{\bar{s}}.\psi\|_V^{\mathcal{I}}$, this means that also $a \in \|\nu X_{\bar{s}'}.\psi\|_V^{\mathcal{I}}$, and thus $a \in [\neg s']_{S_{j'}}^{\mathcal{I}}$. This corresponds to $\neg s'(a) \in S_{j'}$, so clearly also $a \notin [s']_{S_{j'} \cup \neg.U}^{\mathcal{I}}$. To argue why $a \notin [s'']_{S_{j'} \cup \neg.U}^{\mathcal{I}}$, a similar argument as above applies, as $s'' \in W$. Thus, we find $a \notin [\mathcal{C}(s)]_{S_{j'} \cup \neg.U}^{\mathcal{I}}$.
- Other cases are treated similarly.

♣

We can now prove the correctness of our translation by combining these two building blocks in the following way. Here, we use the approximation semantics of the μ -calculus, and the iterative computation of the well-founded semantics via the immediate consequence operator. In a nutshell, we showed that for every b , and i :

1. if $s(b)$ is in the i -th iteration of $W_{\mathcal{I}, \mathcal{C}_s}(\emptyset)$, then there exists a j such that b is in the extension of the j -th approximation of $\mu X_s.\varphi = \text{tr}_{\emptyset, \mathcal{C}}^+(s)$ (Proposition 8.11), and conversely
2. if b is in the extension of the j -th approximation of $\mu X_s.\varphi = \text{tr}_{\emptyset, \mathcal{C}}^+(s)$, then for some j we have that $s(b)$ is in the i -th iteration of $W_{\mathcal{I}, \mathcal{C}_s}(\emptyset)$ (Proposition 8.12).

As both are fixed points, this suffices, as is shown in the following proof.

Theorem 8.14. *For each shapes pair $(\mathcal{C}, s)$ in $\mathcal{ALCIO}_S$, each interpretation $\mathcal{I}$, and each node c of $\mathcal{I}$, we have*

$$\mathcal{I} \text{ validates } (\mathcal{C}, s(c)) \text{ under the well-founded semantics} \quad \text{iff} \quad \mathcal{I}, c \models \text{tr}_{\emptyset, \mathcal{C}}^+(s).$$

Proof. First, note that by Proposition 8.6 and the definition of the approximation functions, we find that

$$\|tr_{\emptyset, \mathcal{C}}^+(s)\|_V^{\mathcal{I}} = \|\mu X.\varphi\|_V^{\mathcal{I}} = \|\mu^\omega X.\varphi\|_V^{\mathcal{I}} = \bigcup_{\alpha < \omega} \|\mu^\alpha X.\varphi\|_V^{\mathcal{I}}.$$

Similarly, as the well-founded semantics is also calculated via a least fixed point, we find that the well-founded assignment consists of the atoms in

$$W_{\mathcal{I}, \mathcal{C}}^{\uparrow \omega}(\emptyset) = \bigcup_{\alpha < \omega} W_{\mathcal{I}, \mathcal{C}}^{\uparrow \alpha}(\emptyset).$$

By Propositions 8.11 and 8.12 we know that for each $\alpha < \omega$, there exists an $\alpha' < \omega$ and $\alpha'' < \omega$ such that $atoms_{\mathcal{I}}(\mu^\alpha X.\varphi) \subseteq W_{\mathcal{I}, \mathcal{C}}^{\uparrow \alpha'}(\emptyset)$ and $W_{\mathcal{I}, \mathcal{C}}^{\uparrow \alpha}(\emptyset) \subseteq atoms_{\mathcal{I}}(\mu^{\alpha''} X.\varphi)$. This suffices to conclude that $c \in \|tr_{\emptyset, \mathcal{C}}^+(s)\|_V^{\mathcal{I}}$ iff $s(c) \in W_{\mathcal{I}, \mathcal{C}}^{\uparrow \omega}(\emptyset)$. $\clubsuit$

This theorem gives us the desired reduction from shape satisfiability and implication to satisfiability in the full hybrid μ -calculus. Specifically, it now directly follows that the following problems can be expressed in full hybrid μ -calculus in the following way.

Corollary 8.15. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be sets of shape constraints $\mathcal{ALCIO}_S$ and s_1 and s_2 shape names. Then*

- *s is (finitely) satisfiable w.r.t. $\mathcal{C}_1$ under the well-founded semantics iff $tr_{\emptyset, \mathcal{C}_1}^+(s)$ is (finitely) satisfiable, and*
- *$(\mathcal{C}_1, s_1)$ (finitely) implies $(\mathcal{C}_2, s_2)$ under the well-founded semantics iff $tr_{\emptyset, \mathcal{C}_1}^+(s) \wedge \neg tr_{\emptyset, \mathcal{C}_2}^+(s)$ is not (finitely) satisfiable.*

Implication and Satisfiability of Documents. For SHACL document satisfiability and implication the reduction is not as straightforward as in the corollary above, as some work needs to be done to encode the different targets. However, with a greatest fixed point that traverses the whole structure, this can also be achieved. We use a fresh role p to be able to jump to certain parts of the structure. As we reduce to satisfiability of a formula, the fresh p 's will be created where necessary, and will not play a role in any of the constraints.

In what follows, we show how to reduce the (finite) document implication problem. Moreover, recall that in Proposition 8.4 we described how (finite) document satisfiability can be reduced to this case.

Now, for any document $(\mathcal{C}, \mathcal{G})$, let $\Theta_{\mathcal{C}, \mathcal{G}}$ be the formula defined as a conjunction of the following formulas.

$$\begin{array}{ll} \neg a \vee tr_{\emptyset, \mathcal{C}}^+(s) & \text{for all } (a, s) \in \mathcal{G} \\ \neg A \vee tr_{\emptyset, \mathcal{C}}^+(s) & \text{for all } (A, s) \in \mathcal{G} \\ \neg \langle r \rangle \top \vee tr_{\emptyset, \mathcal{C}}^+(s) & \text{for all } (r, s) \in \mathcal{G} \end{array}$$

When evaluated on a graph node, $\Theta_{\mathcal{C},\mathcal{G}}$ tells us whether the node satisfies all required target shapes. To propagate $\Theta_{\mathcal{C},\mathcal{G}}$ to all (relevant) nodes of the graph, we use a greatest fixed point formula; for a document $(\mathcal{C}, \mathcal{G})$, and a role name p , let

$$\Lambda_{\mathcal{C},\mathcal{G},p} = \nu X. (\Theta_{\mathcal{C},\mathcal{G}} \wedge \bigwedge_{r \in R \cup \{p\}} [r^-].X \wedge \bigwedge_{r \in R \cup \{p\}} [r].X),$$

where R is the set of role names that appear in $(\mathcal{C}, \mathcal{G})$.

Note that the formula $\Lambda_{\mathcal{C},\mathcal{G},p}$ is only reaching connected components. So, to ensure we reach every relevant node, it suffices to make sure we check $\Lambda_{\mathcal{C},\mathcal{G},p}$ in all nodes mentioned in $\mathcal{C}$ or $\mathcal{G}$. All nodes not connected to those nodes are irrelevant and can be discarded.

Theorem 8.16. *For each pair of SHACL documents $(\mathcal{C}, \mathcal{G})$ and $(\mathcal{C}', \mathcal{G}')$ in $\mathcal{ALC}\mathcal{IO}_S$, we find $(\mathcal{C}, \mathcal{G})$ (finitely) implies $(\mathcal{C}', \mathcal{G}')$ iff*

$$\bigwedge_{a \in I} \langle p \rangle (a \wedge \Lambda_{\mathcal{C},\mathcal{G},p}) \wedge \langle p \rangle \neg \Lambda_{\mathcal{C}',\mathcal{G}',p}$$

is not (finitely) satisfiable, where I is the set of individuals that appear in $(\mathcal{C}, \mathcal{G})$ or $(\mathcal{C}', \mathcal{G}')$, and p a fresh role.

Proof Sketch. First, note that $\Theta_{\mathcal{C},\mathcal{G}}$ holds in a node a iff for all $(\ell, s) \in \mathcal{G}$ such that $\ell = a$, or a is contained in the interpretation of $\ell = A$ or $\exists r. \top$ for $\ell = r$, then also $s(a)$ is contained in the well-founded assignment.

Secondly, notice that $\Lambda_{\mathcal{C},\mathcal{G},p}$ simply ensures that every node reachable by $R \cup R^- \cup \{p\}$ satisfies $\Theta_{\mathcal{C},\mathcal{G}}$. Thus, for each $\mathcal{I}$ if there exists a node a in each connected component that satisfies $\Theta_{\mathcal{C},\mathcal{G}}$, there exists a well-founded assignment of $\mathcal{C}$ on $\mathcal{I}$ validating all targets $\mathcal{G}$.

Lastly, for every $\mathcal{I}$ and SHACL document $(\mathcal{C}, \mathcal{G})$ we can enforce the existence of a fresh node a' and a fresh role p such that $\mathcal{I}$ is extended with the atoms $p(a', a)$ for each $a \in N_I$ occurring in $\mathcal{C}$ or $\mathcal{G}$ without changing the well-founded assignment of $\mathcal{C}$ on $\mathcal{I}$. Thus, by connecting all a occurring in both $(\mathcal{C}, \mathcal{G})$ and $(\mathcal{C}', \mathcal{G}')$, we find (finite) unsatisfiability of $\bigwedge_{a \in I} \langle p \rangle (a \wedge \Lambda_{\mathcal{C},\mathcal{G},p}) \wedge \langle p \rangle \neg \Lambda_{\mathcal{C}',\mathcal{G}',p}$ corresponds exactly to $(\mathcal{C}, \mathcal{G})$ (finitely) implies $(\mathcal{C}', \mathcal{G}')$.

This makes document implication reducible to a single full hybrid μ -calculus formula.

8.5 Complexity Results

Since the full hybrid μ -calculus is decidable, we obtain from Corollary 8.15 and Theorem 8.16 that the static analysis problems of interest are also decidable. This is particularly exciting in light of Theorem 8.1. However, we also want to obtain tight complexity results. The complexity of the μ -calculus and its variants is well understood, but unfortunately, our translation may incur in an exponential blow-up due to the possible repetition of the subformula associated with each shape name. Fortunately, this blow up affects only the representation of the translation as a single formula and not the complexity of the problem.

In this section, we adapt the automata-based technique for the fully hybrid μ -calculus by Sattler and Vardi [SV01] to decide satisfiability of the formulas that result from our translation in time that is single exponential in the size of the original input, and not of the translated formulas. This is followed by an extensive discussion of the complexity of the static analysis problems for the well-founded semantics.

Two-Way Alternating Parity Tree Automata. We recall 2ATA, which can move up and down possibly infinite trees with transitions that may move to combinations of states and directions. For $k \geq 1$, let $(\{1, \dots, k\}^*, \ell)$ be a k -ary Σ -labelled tree if ℓ maps each node $x \in \{1, \dots, k\}^*$ to its label $\ell(x) \in \Sigma$. For $1 \leq i \leq k$, we let $x \cdot i$ be the i th child of x . With $x \cdot 0$ we mean x itself, and $x \cdot -1$ indicates the parent of x ; the node y such that $x = y \cdot i$ for some $1 \leq i \leq k$. Let $\mathcal{B}(I)$ be the set of positive boolean formulas over I , built inductively by applying $\wedge$ and $\vee$ starting from $\perp$, $\top$ and the elements of I . For a set $J \subseteq I$ and a formula $\varphi \in \mathcal{B}(I)$, we say that J satisfies φ iff assigning $\top$ to the elements in J and $\perp$ to those in $I \setminus J$, makes φ true. For a positive integer k , let $[k] = \{-1, 0, 1, \dots, k\}$. A two-way alternating parity tree automaton (2ATA) running over infinite tree encodings with branching degree k , is a tuple $\mathbf{A} = \langle \Sigma, Q, \delta, q_0, p \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \rightarrow \mathcal{B}([k] \times Q)$ is the transition function, $q_0 \in Q$ is the initial state and $p : Q \rightarrow \mathbb{N}$ is the priority function.

A run of a 2ATA $\mathbf{A}$ over a k -ary tree $(\{1, \dots, k\}^*, \ell)$, is a $(\{1, \dots, k\}^* \times Q)$ -labelled tree $\mathcal{T}_r = (T_r, \ell_r)$ with $V_r \subseteq \{1, \dots, k\}^*$, satisfying:

1. $\epsilon \in V_r$ and $\ell_r(\epsilon) = (\epsilon, q_0)$.
2. Let $y \in V_r$ with $\ell_r(y) = (x, q)$ and $\delta(q, \ell(x)) = \varphi$. Then there is a (possibly empty) set $S = \{(c_1, q_1), \dots, (c_n, q_n)\} \subseteq [k] \times Q$ such that:
 - S satisfies φ , and
 - for all $1 \leq i \leq k$, we have that $y \cdot i \in V_r$ and $\ell_r(y \cdot i) = (x \cdot c_i, q_i)$.

Given an infinite path $p \subseteq V_r$, let $\text{inf}(p) \subseteq Q$ be the set of states that appear infinitely often in p . A run $\mathcal{T}_r$ of a 2ATA $\mathbf{A} = \langle \Sigma, Q, \delta, q_0, p \rangle$ is *accepting* if for every infinite path $p \subseteq V_r$, the state with the highest priority that occurs in $\text{inf}(p)$ is even.

Following Sattler and Vardi [SV01], we define *guesses*, which fix the entire configuration of the individuals mentioned in the constraint set. We can enumerate all such configurations and build an automaton for each of them.

Definition 8.17. A guess $\text{guess} = (G, f, C)$ for a constraint set $\mathcal{C}$ containing the nominals $a_1, \dots, a_l$ consists of a guess list $G = (\gamma_1, \dots, \gamma_l)$, a set of connections $C \subseteq N_I \times N_R(\mathcal{C}) \times N_I$ and a guess mapping $f : \{1, \dots, l\} \rightarrow \{1, \dots, l\}$, such that for each $1 \leq i, j \leq l$, $\emptyset \subsetneq \gamma_i \subseteq \text{sub}(\mathcal{C}) \cup \{\neg s \mid s \in \text{sub}(\mathcal{C})\}$ or $\gamma_i = \perp$, $a_i \in \gamma_{f(i)}$, $a_i \notin \gamma_j$, for all $j \neq f(i)$, if $N_I \cap \gamma_i = \emptyset$, then $\gamma_i = \perp$, and $(a_i, r, a_j) \in C$ implies $(a_j, r^-, a_i) \in C$.

Proposition 8.18. For each shapes pair $(\mathcal{C}, s)$ in $\mathcal{ALCCIO}_S$ and each guess $\mathbf{G}$ for $\mathcal{C}$, we can define a 2ATA $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ such that:

1. If $tr_{\emptyset, \mathcal{C}}^+(s)$ is satisfiable, then for some guess $\mathbf{G}$, the language recognised by $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ is non-empty.
2. If for some guess $\mathbf{G}$ the language of $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ is not empty, then $tr_{\emptyset, \mathcal{C}}^+(s)$ is satisfiable.
3. The number of states in $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ is linear in $|sub(\mathcal{C})|$.

The construction is a relatively straightforward adaptation of the translation in Sattler and Vardi [SV01]. The core of the construction uses two states $tr^+(\varphi)$ and $tr^-(\varphi)$ for each φ in $sub(\mathcal{C})$, corresponding to the two functions of the translation. Note that we can ignore the set S in this setting, as we no longer try to build a linear formula capturing a form of circularity. Moreover, the parity condition setting $p(tr^+(\varphi)) = 1$ for each φ in $sub(\mathcal{C})$, and for all other states $p(q) = 0$, reflects exactly the limited alternation of our translation.

The main idea is that the constructed automaton accepts a relaxation of a well-founded adorned pre-model (which has the shape of a forest) of which the corresponding model validates the given SHACL document. For the precise construction of the model out of pre-models of this type, we refer the reader to Sattler and Vardi [SV01].

Proof. We use $a_1, \dots, a_l$ to denote the nominals occurring in $\mathcal{C}$. Furthermore, we assume all input trees are k -ary full trees; all non-leaf nodes have k children, where k is the maximum of the amount of $\langle r \rangle \psi \in sub(\mathcal{C})$, and l plus one.

Let $\mathbf{A}(\mathcal{C}, s, \mathbf{G}) := \langle \Sigma, Q, \delta, \tilde{q}, p \rangle$, be defined as follows. Note that the automaton construction is very closely related to the automaton constructed by Sattler and Vardi [SV01]. Now, let

$$\Sigma = \{\perp, root\} \cup \{\sigma \mid \sigma \subseteq N_C(\mathcal{C}) \cup N_I(\mathcal{C}) \cup \overline{N}_R(\mathcal{C}) \cup \{\overset{r}{\rightarrow} a_i \mid r \in \overline{N}_R(\mathcal{C}), 1 \leq i \leq l\}\},$$

where $N_C(\mathcal{C})$, $N_I(\mathcal{C})$ and $\overline{N}_R(\mathcal{C})$ are the restrictions of the alphabets N_C , N_I and $\overline{N}_R$ to the symbols used in $\mathcal{C}$. The intuition of these symbols is the same as in the mirrored construction by Sattler and Vardi: nodes labelled with ‘ $\perp$ ’ or ‘ $root$ ’ are nodes not representing parts of the model, where the latter is reserved for the root node. This is the node that has all nominals as one of its children in the input tree, so that the information contained in the guess $\mathbf{G}$ can be effectively transmitted to each of the nominals. The σ -label of the other nodes describes exactly which nominal it is (if applicable), which concept names are true and which incoming roles this node has from its unique parent in the input tree.

Next, let the set of states and the priority function be given by

$$\begin{aligned} Q &= \{\perp, \tilde{q}, q_0, q'_0, q_1, \dots, q_l, q, q', q''\} \cup \{tr^+(\varphi), tr^-(\varphi) \mid \varphi \in sub(\mathcal{C})\} \\ &\quad \cup \{\neg s \mid s \in sub(\mathcal{C})\} \cup \{r, \neg r \mid r \in \overline{N}_R(\mathcal{C})\} \\ p(q) &= \begin{cases} 1 & \text{if } q = tr^+(\varphi), \varphi \in sub(\mathcal{C}) \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Note it is immediate that the amount of states is linear in $|sub(\mathcal{C})|$.

Finally, we define the transition function in the following way. The first step (performed by the transition for the starting state $\tilde{q}$) is to split the automaton into two no longer interacting automata, one containing the states $\{q_0, \dots, q_l, q, q''\}$. The idea of this automaton is to guarantee the input tree has a proper shape: the state (q_0) guarantees the root state is only found at the root node, and assigns the q_i states to its i th child, which on their turn check whether the input conforms to the guess. The state q checks whether no nominals show up: we assume all nominals are a source of a tree in the tree relaxation pre-model, and not showing up further down. Finally, the state q'' is there to ensure that the roots of the trees, the nominals, do not have any incoming edges. That is, for each $\sigma \in \Sigma$, let

$$\begin{aligned}\delta(\tilde{q}, \sigma) &= (0, q_0) \wedge (0, q'_0) \\ \delta(q_0, \sigma) &= \begin{cases} \bigwedge_{i=1}^l (i, q_i) \wedge \bigwedge_{i=l+1}^k (i, q) \wedge \bigwedge_{i=1}^k (i, q'') & \text{if } \text{root} = \sigma \\ \perp & \text{otherwise} \end{cases} \\ \delta(q'', \sigma) &= \begin{cases} \top & \text{if } r \notin \sigma, r^- \notin \sigma, \text{ for each } r \in \overline{N}_R(\mathcal{C}) \\ \perp & \text{otherwise.} \end{cases}\end{aligned}$$

In the following, let $1 \leq i \leq l$, $\hat{q} \in Q \setminus \{q_0, q_1, \dots, q_l, q'', q\}$, $r \in \overline{N}_R(\mathcal{C})$, $p \in N_I \cup N_C$, $\hat{\sigma} \in \Sigma$ and $\sigma \in \Sigma \setminus \{\perp\}$.

$$\begin{aligned}\delta(q_i, \hat{\sigma}) &= \begin{cases} \bigwedge_{i=1}^k (i, q) & \text{if } \gamma_i \cap (N_I \cup N_C) = \hat{\sigma} \cap (N_I \cup N_C), \text{root} \neq \hat{\sigma}, \\ & \text{and, for each } a \in N_I \cap \hat{\sigma} \text{ and } (a, r, a') \in C, \xrightarrow{r} a' \in \hat{\sigma} \\ \perp & \text{otherwise} \end{cases} \\ \delta(q, \hat{\sigma}) &= \begin{cases} \bigwedge_{i=1}^k (i, q) & \text{if } \hat{\sigma} \cap N_I = \emptyset \text{ and } \text{root} \neq \hat{\sigma} \\ \perp & \text{otherwise.} \end{cases}\end{aligned}$$

The purpose of the other states is to ensure that the input tree is also the relaxation of a well-founded adorned pre-model. This mostly corresponds to checking whether certain shape expressions hold in certain nodes in the graph. Therefore, most of the states here have the shape of a shape expression/its translation into full hybrid μ -calculus.

Before we get there, we first address the states in which we observe ' $\perp$ ', either as a label of a node in the input tree, or as state. This should be aligned: either both are labelled ' $\perp$ ' or none.

$$\begin{aligned}\delta(\hat{q}, \perp) &= \begin{cases} \top & \text{if } \hat{q} = \perp \\ \perp & \text{otherwise} \end{cases} \\ \delta(\perp, \hat{\sigma}) &= \begin{cases} \top & \text{if } \hat{\sigma} = \perp \\ \perp & \text{otherwise} \end{cases}\end{aligned}$$

Next, we look at the r -states: these are there to ensure that the considered node has indeed an

incoming r -edge, which corresponds to r being contained in σ .

$$\begin{aligned}\delta(r, \sigma) &= \begin{cases} \top & \text{if } r \in \sigma \\ \perp & \text{otherwise} \end{cases} \\ \delta(\neg r, \sigma) &= \begin{cases} \top & \text{if } r \notin \sigma \text{ and } \sigma \neq \text{root} \\ \perp & \text{otherwise} \end{cases}\end{aligned}$$

To ensure the pre-model property, we find the state q'_0 providing the children of the root node (and thus the roots of the forest) a set of subformulas according to the guess $\mathbf{G}$, captured by $\Gamma(i)$. Moreover, as we consider satisfiability of s independent of targets, it suffices to find a node that can be decorated with s . Without loss of generality, this can be any of the nominals, or a blank node which is a root in the pre-model forest.

$$\begin{aligned}\delta(q'_0, \sigma) &= \bigwedge_{i=1}^l \Gamma(i) \wedge \bigvee_{i=1}^k (i, tr^+(s)) \wedge \bigwedge_{i=1}^k ((1, q') \vee (i, \perp)) \\ \Gamma(i) &= \begin{cases} (i, \perp) & \text{if } \gamma_i = \perp \\ \bigwedge_{\varphi \in \gamma_i} (i, tr^+(\varphi)) & \text{if } \gamma_i \subseteq \text{sub}(\mathcal{C}) \cup \{\neg s \mid s \in \text{sub}(\mathcal{C})\} \end{cases}\end{aligned}$$

Furthermore, we find the state q' traversing all the nodes not labelled with ' $\perp$ ', ensuring the proper handling of the combination of modalities and the implicit roles coded in the form $\xrightarrow{r} a_i$ in the description σ of nodes.

$$\begin{aligned}\delta(q', \sigma) &= \bigwedge_{\xrightarrow{r^-} a_i \in \sigma, \forall r. s \in \gamma_{f(i)}} (0, tr^+(s)) \wedge \\ &\quad \bigwedge_{\xrightarrow{r^-} a_i \in \sigma, \{\exists r. s, \neg s'\} \subseteq \gamma_{f(i)}, s' \leftarrow \exists r. s \in \mathcal{C}} (0, tr^-(s)) \wedge \bigwedge_{i=1}^k ((i, q') \vee (i, \perp))\end{aligned}$$

This brings us to the final, and most straightforward cases. Note that here, we no longer need to first introduce a fixed point operator, followed by a later use of a fixed point variable to 'return' to an earlier part of the formula. Instead, we may just replace ' s ' by the constraint defining s . This

unravelling does not work in the translation as we aim to get a formula of finite length.

$$\begin{aligned}
\delta(tr^+(p), \sigma) &= \begin{cases} \top & \text{if } p \in \sigma \\ \perp & \text{otherwise} \end{cases} \\
\delta(tr^+(s \wedge s'), \sigma) &= (0, tr^+(s)) \wedge (0, tr^+(s')) \\
\delta(tr^+(s \vee s'), \sigma) &= (0, tr^+(s)) \vee (0, tr^+(s')) \\
\delta(tr^+(\neg s), \sigma) &= (0, tr^-(s)) \\
\delta(tr^+(s), \sigma) &= (0, tr^+(\mathcal{C}(s))) \\
\delta(tr^+(\exists r.s), \sigma) &= \begin{cases} \top & \text{if } \xrightarrow{r} a_i \in \sigma, s \in \gamma_{f(i)} \\ \bigvee_{j=1}^k ((j, tr^+(s)) \wedge (j, r)) & \text{otherwise} \end{cases} \\
\delta(tr^+(\forall r.s), \sigma) &= \begin{cases} \perp & \text{if } \xrightarrow{r} a_i \in \sigma, s \notin \gamma_{f(i)} \\ ((-1, tr^+(s)) \vee (0, \neg r^-)) \\ \wedge \bigwedge_{j=1}^k ((j, tr^+(s)) \vee (j, \neg r) \vee (j, \perp)) & \text{otherwise} \end{cases} \\
\delta(tr^-(p), \sigma) &= \begin{cases} \top & \text{if } p \notin \sigma \text{ and } \sigma \neq \text{root} \\ \perp & \text{otherwise} \end{cases} \\
\delta(tr^-(s \wedge s'), \sigma) &= (0, tr^-(s)) \vee (0, tr^-(s')) \\
\delta(tr^-(s \vee s'), \sigma) &= (0, tr^-(s)) \wedge (0, tr^-(s')) \\
\delta(tr^-(\neg s), \sigma) &= (0, tr^+(s)) \\
\delta(tr^-(s), \sigma) &= (0, tr^-(\mathcal{C}(s))) \\
\delta(tr^-(\exists r.s), \sigma) &= \begin{cases} \perp & \text{if } \xrightarrow{r} a_i \in \sigma, \neg s \notin \gamma_{f(i)} \\ ((-1, tr^-(s)) \vee (0, \neg r^-)) \wedge \\ \bigwedge_{j=1}^k ((j, tr^-(s)) \vee (j, \neg r) \vee (j, \perp)) & \text{otherwise} \end{cases} \\
\delta(tr^-(\forall r.s), \sigma) &= \begin{cases} \top & \text{if } \xrightarrow{r} a_i \in \sigma, \neg s \in \gamma_{f(i)} \\ \bigvee_{j=1}^k ((j, tr^-(s)) \wedge (j, r)) & \text{otherwise.} \end{cases}
\end{aligned}$$

♣

(Finite) Shape Satisfiability w.r.t. Constraints. Based on the aforementioned 2ATA construction, we can now derive the first set of complexity results.

Theorem 8.19. *Deciding shape satisfiability w.r.t constraints for $\mathcal{ALC}\mathcal{IO}_S$ is EXPTIME-complete, under the well-founded semantics.*

Proof. As noted by Sattler and Vardi [SV01], emptiness of $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ can be decided in exponential time in the number of states. Moreover, the amount of guesses is also bounded by the amount of connections and guess lists, which means there are at most exponentially many automata $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ that need to be tested for emptiness, thus shape satisfiability w.r.t. constraints can be decided in EXPTIME.

EXPTIME-hardness follows for instance from reducing document implication to document satisfiability (Proposition 8.4), which is an EXPTIME-complete problem for $\mathcal{ALC}\mathcal{IO}_S$ (Proposition 7.6). ♣

For a more detailed coverage of restricted settings that are already EXPTIME-hard, see the previous chapter.

For the finite version of the above problem we do not provide tight bounds for full $\mathcal{ALC}\mathcal{IO}_S$, but decidability already follows from finite satisfiability of full hybrid μ -calculus being decidable. For $\mathcal{ALC}\mathcal{O}_S$ we do find tight bounds, via its finite model property under the well-founded semantics.

Proposition 8.20. *$\mathcal{ALC}\mathcal{O}_S$ has the finite model property, $\mathcal{ALC}\mathcal{I}_S$ does not, under the well-founded semantics.*

Proof. Given the result of Theorem 8.14, it follows that SHACL documents expressible in $\mathcal{ALC}\mathcal{O}_S$ can be reduced to hybrid μ -calculus, that is, the full hybrid μ -calculus without inverse programs. Tamura shows that this fragment has the finite model property [Tam15], thus, by our reduction, we find that also $\mathcal{ALC}\mathcal{O}_S$ has the finite model property.

To see that $\mathcal{ALC}\mathcal{I}_S$ does not have the finite model property, consider the following set of (stratified) constraints: $\mathcal{C} = \{s \leftarrow \neg s', s' \leftarrow \forall r. s' \wedge \neg s'', s'' \leftarrow \forall r^-. s''\}$. Satisfiability of s against $\mathcal{C}$ corresponds to satisfiability of $cln(tr_{\emptyset, \mathcal{C}}^+(s)) = \nu X_{s'}.\langle r \rangle X_{s'} \wedge \mu Y_{s''}.[r^-]Y_{s''}$, which is a well-known example of a modal μ -calculus formula that only has infinite models. ♣

Theorem 8.21. *Deciding finite shape satisfiability w.r.t. constraints for $\mathcal{ALC}\mathcal{IO}_S$ SHACL is decidable in 3-EXPTIME, under the well-founded semantics.*

Proof. Given the result of Theorem 8.14, the finite satisfiability of the full hybrid μ -calculus being decidable suffices. The latter follows from the finite satisfiability of guarded fixed point logic, which naturally extends the modal μ -calculus with backwards modalities [BB12]. In guarded fixed point logics, given that there is no bound on the arity of predicates, we can add a prefix $x_1, \dots, x_n$ to every predicate, where n is the amount of individuals used, and use x_i in place of the i -th individual, to model the usage of nominals [TCF05].

For the upper bound, recall the translation of SHACL into μ -calculus may come at the cost of a formula of size exponential in the size of the constraint set. Thus, we can combine finite satisfiability results for different fragments of the full hybrid μ -calculus by adding one exponential. As noted above, full hybrid μ -calculus is expressible in guarded fixed point logic with unbounded arity. As finite satisfiability is 2-EXPTIME-complete for this fragment (Theorem 1 in [BB12]), we find that finite shape satisfiability w.r.t. constraints for $\mathcal{ALC}\mathcal{IO}_S$ SHACL is decidable in 3-EXPTIME. ♣

For $\mathcal{ALC}\mathcal{I}_S$, 2-EXPTIME-membership for deciding finite satisfiability follows from adding the one exponential of the translation of SHACL into μ -calculus to the EXPTIME-completeness of deciding finite satisfiability of the μ -calculus with backwards modalities as proven by Bojanczyk

Fragment	Finite	Unrestricted
$\mathcal{ALCI}_S$	in 2-EXPTIME	EXPTIME-complete
$\mathcal{ALCO}_S$	EXPTIME-complete	EXPTIME-complete
$\mathcal{ALCIO}_S$	in 3-EXPTIME	EXPTIME-complete

Table 8.1: Complexity of deciding (finite) shape satisfiability w.r.t. constraints, document satisfiability and document containment under the well-founded semantics.

[Boj02]. We have no reason to believe these upper bounds are tight: they are obtained via multiple translations that cause one or more exponential blow-ups, which could well be avoidable in a more direct approach.

(Finite) Satisfiability and Implication of Documents. With a small update of the automaton construction, the EXPTIME complexity results transfer to the problem of document satisfiability.

Theorem 8.22. *Deciding document satisfiability for $\mathcal{ALCIO}_S$ is EXPTIME-complete, under the well-founded semantics. Finite document satisfiability for $\mathcal{ALCIO}_S$ is decidable in 3-EXPTIME.*

Proof. The main idea is to take the 2ATA $\mathbf{A}(\mathcal{C}, s, \mathbf{G})$ for some $(c, s) \in \mathcal{G}$, and update the transition function δ in the following way. For every $(D, s) \in \mathcal{G}$, with $D \in N_C \cup \overline{N}_R$, if $D \in \sigma$, the transition function for the state that is traversing the whole tree already, q' , is updated with an extra conjunct: $(0, tr^+(s))$. For the individuals $a_i \in N_I$, this can directly be taken care of by adding $(i, tr^+(s))$ as a conjunct to $\delta(q'_0, \sigma)$, where q'_0 is the state that already ensures all nominals in the graph are assigned the right states.

More specifically, taking the exact same automaton as in the proof of Proposition 8.18, with the following being redefined as described below, suffices.

$$\begin{aligned}
\Gamma(i) &= \begin{cases} (i, \perp) & \text{if } \gamma_i = \perp \\ \bigwedge_{\varphi \in \gamma_i} (i, tr^+(\varphi)) \wedge \bigwedge_{(a_i, s) \in \mathcal{G}} (i, tr^+(\varphi)) & \text{if } \gamma_i \subseteq sub(\mathcal{C}) \cup \{\neg s \mid s \in sub(\mathcal{C})\} \end{cases} \\
\delta(q', \sigma) &= \bigwedge_{\xrightarrow{r^-} a_i \in \sigma, \forall r. s. s \in \gamma_{f(i)}} (0, tr^+(s)) \wedge \bigwedge_{(D, s) \in \mathcal{G}, D \in (N_C \cup \overline{N}_R) \cap \sigma} (0, tr^+(s)) \\
&\quad \wedge \bigwedge_{\xrightarrow{r^-} a_i \in \sigma, \{\exists r. s, \neg s'\} \subseteq \gamma_{f(i)}, s' \leftarrow \exists r. s \in \mathcal{C}} (0, tr^-(s)) \wedge \bigwedge_{i=1}^k ((i, q') \vee (i, \perp))
\end{aligned}$$

For finite satisfiability, decidability follows from combining different results presented before. That is, because of Proposition 8.4 we can reduce to finite implication, at a polynomial cost. Then, because of Theorem 8.16 this can be reduced to finite satisfiability of a single full hybrid μ -calculus formula at the cost of one exponential. As argued in the proof of Theorem 8.21, finite satisfiability for the full hybrid μ -calculus is decidable in 2-EXPTIME, which provides a 3-EXPTIME algorithm for deciding finite document satisfiability for $\mathcal{ALCIO}_S$. ♣

A direct consequence of this result is that for $\mathcal{ALCO}_S$ deciding finite satisfiability w.r.t. constraints under the well-founded semantics is EXPTIME-complete, since the fragment has the finite model property; see Proposition 8.20. For $\mathcal{ALCI}_S$, similar reasoning as before applies to find that finite document satisfiability is decidable in 2-EXPTIME.

As the complement of an 2ATA can be constructed in polynomial time [MS87], it follows that deciding whether $(\mathcal{C}, \mathcal{G})$ implies $(\mathcal{C}', \mathcal{G}')$ under the well-founded semantics for $\mathcal{ALCIO}$ SHACL can be decided by taking the conjunction of the automaton checking document satisfiability for $(\mathcal{C}, \mathcal{G})$ and the complement of the automaton recognising document satisfiability for $(\mathcal{C}', \mathcal{G}')$.

Theorem 8.23. *Deciding document implication for $\mathcal{ALCIO}_S$ is EXPTIME-complete, under the well-founded semantics. Finite document implication for $\mathcal{ALCIO}_S$ is decidable in 3-EXPTIME.*

Note that in the finite case, again the EXPTIME-completeness, 2-EXPTIME and 3-EXPTIME results follow for $\mathcal{ALCO}_S$, $\mathcal{ALCI}_S$ resp. $\mathcal{ALCIO}_S$: by Theorem 8.16, (finite) document implication is reducible to (finite) satisfiability of a hybrid μ -calculus, μ -calculus with backwards modalities, resp. full hybrid μ -calculus formula at the cost of at most one exponential.

Least Fixed Point Semantics. Apart from the supported, stable model and well-founded semantics for recursive SHACL, we have also addressed the least fixed point semantics for stratified SHACL in Section 4.3. As the name already suggests, this is a semantics only using a least fixed point in its definition, making it conceptually easier. Nonetheless, this suffices to compute either the well-founded or stable model semantics over stratified constraints; if $\mathcal{C}$ is stratified, these three semantics coincide. In particular, this means the discussed static analysis problems for least fixed points semantics can be simply copied from the well-founded setting discussed above. Moreover, the translation of stratified SHACL, and thus also of SHACL under the least fixed point semantics, would fall in the alternation free μ -calculus. To see this, in a nutshell, note that in a μ -calculus formula with alternation like $\mu X_s. \varphi$ such that $\nu X_{s'}. \varphi(X_s) \in \text{sub}(\varphi)$ we find s' depending negatively on s ; because of the ν -formula appearing inside the μ -formula, and s depending negatively on s' ; because of X_s appearing inside the ν -subformula. Clearly, this makes the constraint set unstratified. We are optimistic that this can be exploited to achieve more scalable static analysis results for the least fixed point semantics.

8.6 Conclusion

In this chapter, by establishing a connection to the full hybrid μ -calculus, we provide the first decidability and complexity results for basic static analysis problems in recursive SHACL under the well-founded semantics. These results contrast sharply with the undecidability results we show for the supported and stable model semantics. A key observation is that the implication problem becomes computationally unmanageable when the underlying semantics can assign multiple shape assignments to a graph. The well-founded semantics yields a unique shape assignment for every graph, which forms one of the bases under our decidability result.

The results in this chapter clearly speak in favour of the well-founded semantics for SHACL. This means that next to deciding validation, which for full recursive SHACL is only tractable for the well-founded semantics, now also the deciding implication is decidable only for this case.

There are several directions for future work. First, in the context of practical applications, it is relevant to understand static analysis in the presence of a richer syntax for shape expressions. For instance, adding numeric restrictions (such as qualified number restrictions in DLs) to SHACL under the well-founded semantics is a natural next step. This requires understanding how our algorithmic methods can be adapted to it. Second, our translation from SHACL under the well-founded semantics into the full hybrid μ -calculus raises the question of whether a similar translation exists for Datalog with negation under the well-founded semantics into Fixpoint Logic. A natural follow-up question is whether decidability results for Guarded Fixpoint Logic can yield new positive results for restricted Datalog fragments under the well-founded semantics.

8.7 Related Work

To our knowledge, document and shape satisfiability are the only problems presented in this chapter that have been studied before, except for one work that studied implication (sometimes called containment) for SHACL under the supported model semantics, but in a more restricted setting; Leinberger et al. ([LSR⁺20]) studied satisfiability and implication problem for shape expressions alone, with no constraints. Their problem corresponds to deciding whether $(\{s \leftarrow \varphi_1\}, s)$ is satisfiable, and whether $(\{s_1 \leftarrow \varphi_1\}, s_1)$ implies $(\{s_2 \leftarrow \varphi_1\}, s_2)$ for given φ_1, φ_2 . However, as discussed in and around Example 4 in [BJVdB22b], there are issues with their translation, sometimes leading to wrong validation results.

Naturally, the related work of Chapter 7 also applies here. Specifically, note that undecidability results for satisfiability transfer to implication, which means that undecidability results for the supported model semantics are presented and can be derived from the work by Pareti et al. [PKM22], although for significantly more expressive SHACL fragments than the results presented in this chapter. In general, our understanding of satisfiability and implication in recursive SHACL is still very limited. In fact, to the best of our knowledge, no other attempts have been made to encode document implication directly into decidable logic formalisms, or solve it by other means.

The well-founded semantics for SHACL was originally defined by Okulmus and Šimkus [OŠ24], inspired by the well-founded semantics for general logic programs [VGRS91]. Next to providing the semantics, Okulmus and Šimkus built a tool, shaWell, that handles recursion in SHACL in a principled way.

The connection between the well-founded semantics and the modal μ -calculus is, to our knowledge, novel and of independent interest. We note that some works on the well-founded semantics for logic programs have already established a connection for logics with fixed points [VGRS91, VG93]. The setting discussed there is different: their monotonic operator is composed of two anti-monotonic fixed point operators. That is, the first operator produces an overestimate of the set of negative conclusions, whereas the second is an underestimate. This alternating, estimating behaviour is absent in this chapter, in the sense that we translate (directly) into (fragments of) the

full hybrid μ -calculus. We rely on the insights of our more constructive translation to derive the tight complexity bounds.

For SHACL under multiple semantics, Bogaerts and Jakubowski [BJ21] have considered the connection to approximation fixed point theory. As this captures all the main semantics for recursive SHACL, this also does not provide the specific insights in the well-founded semantics for SHACL as was of interest in this chapter.

We are not aware of any works considering satisfiability and containment under the stable or well-founded semantics, but an EXPTIME upper bound for shape satisfiability w.r.t. constraints under the stable semantics for the $\mathcal{ALCI}$ fragment can be inferred from the work of Di Stefano and Simkus [DSŠ24] on DL terminologies under the stable (a.k.a. *equilibrium*) semantics.

Conclusion

In this work, we have looked at two main aspects of advanced reasoning for recursive SHACL; how validation can work in presence of ontologies, and for which SHACL fragments certain static analysis problems are decidable (and how complex they are). The first half of the thesis was dedicated to the first question, whereas the last two chapters discussed the latter. Moreover, we also provided an overview of SHACL as a logic.

The first half is based on the research question *How can we combine SHACL and OWL?* and starts with arguing for a semantics based on an *austere* universal model: a model in which axioms are satisfied minimally, in the sense of introducing the least amount of successor nodes, while remaining a universal model. We provided explicit ways to build this model for $DL\text{-}Lite_{\mathcal{R}}$, $\mathcal{ELHI}$ and Horn- $\mathcal{ALCHIQ}$ in Chapters 4 and 5. It turns out that the austere canonical model in fact coincides with the unique core universal model of any given knowledge base that is consistent.

The semantics we argue for exists of producing the austere or core universal model for the given knowledge base, the combination of data graph and ontology, and evaluating the given SHACL document over this extension of the data graph or ABox. Given the access SHACL constraints have to negation, this appears to be the most intuitive, and also coincides with the argued for semantics in literature for answering queries with negation over knowledge bases [BF19, EKM22]. This provides an answer to Q1: *What are adequate semantics for validating SHACL in presence of ontologies?* for all knowledge bases for which we know there exists a unique core universal model, which contains at least the ones expressible in Horn- $\mathcal{ALCHIQ}$. As is discussed at the end of Chapter 4, already moving to Horn- $\mathcal{SHIQ}$ removes this property: logics allowing for transitive roles do not need a lot of expressivity to produce multiple core universal models, or none at all, depending on how the definition of ‘core’ is extended to infinite structures. That is, we provided an answer to Q1 for relative expressive fragments, but interesting problems remain. Simple extensions already provide new challenges: does it make sense to validate SHACL over all core universal models of Horn- $\mathcal{SHIQ}$ knowledge bases? Is it enough if one of them has a satisfactory assignment? And if we push it even further, to non-Horn logics, which do not have a universal model at all, what happens then?

However, just providing the above semantics does not answer Q3: *How can we validate these semantics?* In fact, building the universal core model might be infeasible, as it will be infinite in practice. To circumvent this, we alternatively realise validation via a rewriting of SHACL and produced an updated set of SHACL constraints, based on the axioms in the TBox or ontology. This rewriting relies on the finite representation of universal core models called the *good successor configuration*. In this way, we can simulate the construction of a least fixed point shape assignment of the full model in multiple smaller pieces of the model. Because of its modularity, this also allows for addressing the recursion in SHACL. Finally, stratified negation can be taken care of by first reaching a least fixed point, followed up by taking the complement of the information not derived for the modular pieces of the model. These are new techniques that have potential to be extended to other settings containing stratified negation in a query or constraint setting over ontologies.

This rewriting technique also directly provides one bound to answer Q2, namely how complex this semantics is: the rewriting produces a SHACL document that is exponentially bigger than the original one, but only in size of the TBox. As plain SHACL validation is much simpler, this means that (simple) SHACL validation in presence of Horn-*ALCHIQ* ontologies can be decided in EXPTIME in combined complexity. This turns out to be an optimal bound as it is possible to run an alternating Turing machines in SHACL validation over KBs. If however we only consider the data as a parameter, we find PTIME-completeness, as the exponential blow-up is not taking in the data size as a variable.

Nevertheless, finding EXPTIME-completeness does not mean a best case exponential procedure is necessary. In Chapter 5, we explore how taking into account data-awareness can improve the situation. And indeed, this algorithm is not always producing a exponential size rewriting. This is a first step towards an implementation, but a lot of work remains to be done in this aspect. In some recent work, some fragments are addressed [KZA24, OGA26], but a lot of work remains to be done in this area, in particular given that of all implementations of SHACL, only shaWell handles recursion in SHACL in a principled way [OŠ24].

In the results discussed above, the SHACL fragment considered was always a simplified form of SHACL. One of the features that had not received any attention yet in Chapters 4 and 5 is counting further than one over regular expressions, in particular in presence of ontologies. In Chapter 6 we study this, and conclude that it is very easy to make reasoning hard (intractable). As semantics, we consider here the least and maximal amounts of nodes reachable over all models of a given knowledge base, as a way of instantiating the certain answer semantics. We find that any type of inverses, either in the ontology or in the regular expression, directly leads to intractability, as does any logic that can access the concept names its neighbours participates in and which has express some form of disjointness. That is, almost every logic in the DL family is too expressive. To circumvent these constraints, we restrict *DL-Lite_R* and remove any interaction between roles in different directions. It turns out that this is sufficient to provide a polynomial time algorithm for deciding the MINCOUNT problem, which corresponds to deciding the minimal answer. This procedure is based on identifying a small set of models over which the minimal answer corresponds to the minimal answer over all models. As building all these models would be tedious, we furthermore provide an approach that simulates how these models would look like,

and brings the question whether there exist at least k answers over all models of a knowledge base back to satisfiability of a single logical formula. The other problem, deciding the maximal amount, turns out to be trivial and not very interesting for simple DLs, as intuitively follows from the notion of making more copies of reachable nodes.

For future work, it will be interesting to see to which extent the ideas and techniques used in the last chapter of this part can be embedded in providing semantics for non-Horn logics, in particular to provide rewriting techniques that consider a set of models instead of a single one.

In the second half of this thesis, we focussed on static analysis for recursive SHACL, in the form of Q4: *Can we find decidable instances of (recursive) SHACL satisfiability and containment or implication?* First, we explored the connection between SHACL and DLs further, and concluded that this can be exploited to obtain many new complexity results for SHACL satisfiability. Moreover, this chapter also provided some lack of finite model property results that were not yet considered in the DL literature. In the unrestricted cases, complexity of the satisfiability problem for SHACL can be derived for almost every fragment considered. However, for the finite case, many instances remain open.

In Chapter 8, we take a closer look at SHACL implication, and conclude that for both the supported as the stable model semantics, shape implication w.r.t. constraints is undecidable. However, for the third main semantics used for recursive SHACL, the well-founded semantics, we find various implication and satisfiability problems to be EXPTIME-complete, in particular for $\mathcal{ALCOI}_S$. To see this, we push the relation to various logics further by exploring the connections between the full hybrid μ -calculus and SHACL under the well-founded semantics. We provide a direct translation that takes any SHACL document and translates it into a single full hybrid μ -calculus formula that evaluates to true in any data graph that has a satisfying well-founded shape assignment. Next to providing insight into how the well-founded semantics can be expressed directly into fixed point logics, this also gives upper bounds on how complex SHACL document containment and implication can be. However, this translation of SHACL into μ -calculus may come at an exponential cost; this can be expected when translating recursive constraints into a single, linear, formula. Nevertheless, we show how the exponential blow-up can be circumvented by directly translating SHACL documents into two-way alternating parity tree automata. In this way, we can derive optimal, EXPTIME-complete, complexity bounds.

This means that for $\mathcal{ALCOI}_S$, the picture looks quite complete: except for satisfiability under the stable model semantics,¹ the complexity of satisfiability and implication is known. However, for the rest, there mostly remains a lot of open problems. For $\mathcal{ALCOQ}_S$ or $\mathcal{ALCTQ}_S$ under the well-founded semantics, it appears to be rather straightforward to translate these languages into decidable fragments of the graded full hybrid μ -calculus. Finite satisfiability and containment problems, however, might need another type of reduction to be settled.

In general, this thesis provides substantive answers to all the research questions posed. Nonetheless, maybe even more open and interesting questions appeared in trying to answer them.

¹An EXPTIME upper bound for $\mathcal{ALCT}_S$ satisfiability under the stable model semantics can be derived from the work of Di Stefano and Šimkus [DSŠ24].

Overview of Generative AI Tools Used

ChatGPT was used to proofread an earlier version of Chapter 6 and assisted in translating the abstract into German, followed by careful human editing. For the rest, no Generative AI Tools were used.

List of Figures

2.1	Graph representation of the ABox presented in Example 2.7.	14
2.2	(Strong) homomorphisms: there does not exist a strong homomorphism from $\mathcal{I}$ (left) into $\mathcal{I}'$ (right).	19
2.3	The interpretation $\mathcal{I}$ as described in Example 2.22 (left) and the core of $\mathcal{A}_{\mathcal{I}}$ (right)	20
3.1	Evaluating shape expressions	26
3.2	Evaluating shape expressions: upper and lower bounds.	29
4.1	Result of oblivious or Skolem chase (left) and austere universal model (right). We use hasWP and hasP as a shorthand for hasWingedPet and hasPet	36
4.2	Given the TBox $\mathcal{T}$ as in Example 5.2, the good successor configurations for the sets of 2-types $F = \{(\{B_0, B_1\}, \{r_1\}, \{A_2\}), (\{B_0, B_1\}, \{r_1\}, \{A_2\})\}$ and $F' = \{(\{B_0, B_1\}, \{r_1, r_2\}, \{A_2\})\}$, are $\text{succ}_{\mathcal{T}}(F) = \{(\{r_0, r_1\}, \{A_0, A_1\}), (\{r_2\}, \{A_2\})\}$ and $\text{succ}_{\mathcal{T}}(F') = \{(\{r_0, r_1\}, \{A_0, A_1\})\}$	38
4.3	Evaluating shape expressions	47
4.4	Austere Universal Model $\text{can}(\mathcal{T}, \mathcal{A})$, for $\mathcal{A} = \{A(a), p(a, b)\}$ and $\mathcal{T} = \{A \sqsubseteq \exists p.B, B \sqsubseteq \exists q.C\}$	56
4.5	Evaluation of SHACL ^b shape expressions.	67
4.6	Two core universal models (after taking the transitive closure of p) of one knowledge base. Two of the non-surjective homomorphisms are denoted by the dashed, grey arrows.	73
5.1	Evaluation of semi-positive SHACL shape expressions.	83
6.1	ABox construction for $\mathcal{G} = (\{1, 2, 3, 4\}, \{\{1, 2\}, \{1, 3\}, \{1, 4\}\})$. Black arrows represent the role <i>edge</i> , grey arrows <i>aux</i> , and dashed black arrows <i>hasCol</i> . Furthermore, $w_1 = \text{red}$, $w_2 = \text{red} \cdot \text{GreenWord} \cdot \text{BlueWord} \cdot \text{aux} \cdot \text{hasCol}$, $w_3 = \text{RedWord} \cdot \text{green}$, $w_4 = \text{green} \cdot \text{BlueWord} \cdot \text{aux} \cdot \text{hasCol}$, $w_5 = \text{RedWord} \cdot \text{GreenWord} \cdot \text{blue}$, $w_6 = \text{blue} \cdot \text{aux} \cdot \text{hasCol}$ and $w_7 = \text{RedWord} \cdot \text{GreenWord} \cdot \text{BlueWord}$	92
6.2	ABox construction: extension of roles.	93
7.1	Translation functions mapping $\mathcal{ALCOIQ}$ concepts into $\mathcal{ALCOIQ}_S$ shape expressions, and vice versa.	112
7.2	Infinite grid that, after adding s_f and s_g as label to every node, shows satisfiability of s . The red diagonal arrows denote the role d	114
		153

7.3	Decidability and complexity of SHACL fragments under the supported model semantics. Ellipse-shaped nodes denote (finite) satisfiability is decidable in EXPTIME (yellow border), or NEXPTIME (green border). Squared-shaped nodes indicate satisfiability is undecidable. A yellow filling indicates the presence of the finite model property, whereas a red filling stands for the lack of it. Arrows indicate subsumption of fragments.	115
7.4	Infinite grid that can be enforced in $\mathcal{ALCN}(*, \cup)_S$	116
8.1	Infinite grid that, after adding s as label to every node, shows undecidability of shape implication w.r.t. constraints under the supported model semantics. The grey dashed arrows indicate the extension of the role l	123
8.2	Semantics of μ -calculus formulas.	126
8.3	Cleaning function.	126
8.4	Translation functions, with emphasis on the negated translation of a negation, $tr_{S,C}^+(\neg s)$: the well-founded semantics does not treat positive and negative information equally, which is captured by only keeping the positive shape names in the set S : $pos(S) := S \setminus \{\bar{s} \in S\}$	128

List of Tables

7.1	Fragments of SHACL following the DL naming convention, extended with counting over regular paths.	111
8.1	Complexity of deciding (finite) shape satisfiability w.r.t. constraints, document satisfiability and document containment under the well-founded semantics.	143

Bibliography

- [ABW88] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, 1988.
- [ACO⁺20] Medina Andresel, Julien Corman, Magdalena Ortiz, Juan L. Reutter, Ognjen Savkovic, and Mantas Šimkus. Stable model semantics for recursive SHACL. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *Proc. of WWW 2020*, pages 1570–1580. ACM / IW3C2, 2020.
- [AGP18] Marcelo Arenas, Georg Gottlob, and Andreas Pieris. Expressive languages for querying the semantic web. *ACM Trans. Database Syst.*, 43(3):13:1–13:45, 2018.
- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [Alb01] Luca Alberucci. Strictness of the modal μ -calculus hierarchy. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*, pages 185–201. Springer, 2001.
- [AN01] André Arnold and Damian Niwinski. Rudiments of μ -calculus. *Studies in Logic and the Foundations of Mathematics*, 146, 2001.
- [AOS⁺23] Shqiponja Ahmetaj, Magdalena Ortiz, Anouk M. Oudshoorn, and Mantas Šimkus. Reconciling SHACL and ontologies: Semantics and validation via rewriting. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Radulescu, editors, *Proc. of ECAI 2023*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 27–35. IOS Press, 2023.
- [AOS20] Shqiponja Ahmetaj, Magdalena Ortiz, and Mantas Simkus. Polynomial rewritings from expressive description logics with closed predicates to variants of datalog. *Artif. Intell.*, 280:103220, 2020.
- [Bau95] Bruce L. Bauslaugh. Core-like properties of infinite graphs and structures. *Discret. Math.*, 138(1-3):101–111, 1995.

- [Bau96] Bruce L. Bauslaugh. Cores and compactness of infinite directed graphs. *J. Comb. Theory, Ser. B*, 68(2):255–276, 1996.
- [BB12] Vince Bárány and Mikolaj Bojanczyk. Finite satisfiability for guarded fixpoint logic. *Inf. Process. Lett.*, 112(10):371–375, 2012.
- [BCM⁺03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [Ber66] Robert Berger. *The undecidability of the domino problem*. Number 66. American Mathematical Soc., 1966.
- [BF19] Stefan Borgwardt and Walter Forkel. Closed-world semantics for conjunctive queries with negation over ELH-bottom ontologies. In Sarit Kraus, editor, *Proc. of IJCAI 2019*, pages 6131–6135. ijcai.org, 2019.
- [BHLS17] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.
- [BJ21] Bart Bogaerts and Maxime Jakubowski. Fixpoint semantics for recursive SHACL. In Andrea Formisano, Yanhong Annie Liu, Bart Bogaerts, Alex Brik, Verónica Dahl, Carmine Dodaro, Paul Fodor, Gian Luca Pozzato, Joost Vennekens, and Neng-Fa Zhou, editors, *Proc. of ICLP 2021*, volume 345 of *EPTCS*, pages 41–47, 2021.
- [BJVdB22a] Bart Bogaerts, Maxime Jakubowski, and Jan Van den Bussche. Expressiveness of SHACL features. In Dan Olteanu and Nils Vortmeier, editors, *Proc. of ICDT 2022*, volume 220 of *LIPICs*, pages 15:1–15:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [BJVdB22b] Bart Bogaerts, Maxime Jakubowski, and Jan Van den Bussche. SHACL: A description logic in disguise. In Georg Gottlob, Daniela Inclezan, and Marco Maratea, editors, *Proc. of LPNMR 2022*, volume 13416 of *Lecture Notes in Computer Science*, pages 75–88. Springer, 2022.
- [BJVdB24] Bart Bogaerts, Maxime Jakubowski, and Jan Van den Bussche. Expressiveness of SHACL features and extensions for full equality and disjointness tests. *Log. Methods Comput. Sci.*, 20(1), 2024.
- [BK22] Bartosz Bednarczyk and Emanuel Kieronski. Finite entailment of local queries in the Z family of description logics. In *Proc. of AAAI 2022*, pages 5487–5494. AAAI Press, 2022.
- [BLB08] Franz Baader, Carsten Lutz, and Sebastian Brandt. Pushing the EL envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proc. of OWLED 2008*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

- [BLMV08] Piero A. Bonatti, Carsten Lutz, Aniello Murano, and Moshe Y. Vardi. The complexity of enriched mu-calculi. *Log. Methods Comput. Sci.*, 4(3), 2008.
- [BMT20] Meghyn Bienvenu, Quentin Manière, and Michaël Thomazo. Answering counting queries over DL-Lite ontologies. In Christian Bessière, editor, *Proc. of IJCAI 2020*, pages 1608–1614. ijcai.org, 2020.
- [BMT22] Meghyn Bienvenu, Quentin Manière, and Michaël Thomazo. Counting queries over $\mathcal{ELHI}_\perp$ ontologies. In *Proc. of KR 2022*, pages 53–62, 2022.
- [BO15] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In Wolfgang Faber and Adrian Paschke, editors, *Reasoning Web 2015, Tutorial Lectures*, volume 9203 of *Lecture Notes in Computer Science*, pages 218–307. Springer, 2015.
- [Boj02] Mikolaj Bojanczyk. Two-way alternating automata and finite models. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan J. Eidenbenz, and Ricardo Conejo, editors, *Proc. of ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 833–844. Springer, 2002.
- [BS96] Franz Baader and Ulrike Sattler. Number restrictions on complex roles in description logics: A preliminary report. In *Proc. of KR 1996*, pages 328–339, 1996.
- [BS99] Franz Baader and Ulrike Sattler. Expressive number restrictions in description logics. *J. Log. Comput.*, 9(3):319–350, 1999.
- [BtCLW14] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.
- [CCK⁺17] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471–487, 2017.
- [CCLR20] Diego Calvanese, Julien Corman, Davide Lanti, and Simon Razniewski. Counting query answers over a DL-Lite knowledge base. In Christian Bessière, editor, *Proc. of IJCAI 2020*, pages 1658–1666. ijcai.org, 2020.
- [CEO09] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In Craig Boutilier, editor, *Proc. of IJCAI 2009*, pages 714–720, 2009.
- [CGK13] Andrea Cali, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

- [CGL⁺07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.*, 39(3):385–429, 2007.
- [CGL⁺11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.
- [CKM⁺18] David Carral, Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Sebastian Rudolph. Preserving constraints with the stable chase. In Benny Kimelfeld and Yael Amerdamer, editors, *Proc. of ICDT 2018*, volume 98 of *LIPICs*, pages 12:1–12:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- [CKNT08] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Camilo Thorne. Aggregate queries over ontologies. In Ramez Elmasri, Martin Doerr, Mathias Brochhausen, and Hyoil Han, editors, *Proc. of ONISW 2008*, pages 97–104. ACM, 2008.
- [CRS18] Julien Corman, Juan L. Reutter, and Ognjen Savkovic. Semantics and validation of recursive SHACL. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *Proc. of ISWC 2018*, volume 11136 of *Lecture Notes in Computer Science*, pages 318–336. Springer, 2018.
- [DNR08] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In Maurizio Lenzerini and Domenico Lembo, editors, *Proc. of PODS 2008*, pages 149–158. ACM, 2008.
- [DSŠ24] Federica Di Stefano and Mantas Šimkus. Stable model semantics for description logic terminologies. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Proc. of AAI 2024*, pages 10484–10492. AAI Press, 2024.
- [EKM22] Stefan Ellmauthaler, Markus Krötzsch, and Stephan Mennicke. Answering queries with negation over existential rules. In *Proc. of AAI 2022*, pages 5626–5633. AAI Press, 2022.
- [EOŠ⁺12] Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for horn-shiq plus rules. In Jörg Hoffmann and Bart Selman, editors, *Proc. of AAI 2012*, pages 726–733. AAI Press, 2012.
- [FG10] Wenfei Fan and Floris Geerts. Relative information completeness. *ACM Trans. Database Syst.*, 35(4):27:1–27:44, 2010.
- [FGG22] Daniel Fernández-Álvarez, José Emilio Labra Gayo, and Daniel Gayo-Avello. Automatic extraction of shapes using shexer. *Knowl. Based Syst.*, 238:107975, 2022.

- [FLP21] Cristina Feier, Carsten Lutz, and Marcin Przybylko. Answer counting under guarded tgds. In Ke Yi and Zhewei Wei, editors, *Proc. of ICDT 2021*, volume 186 of *LIPICs*, pages 11:1–11:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [GHKP12] Birte Glimm, Aidan Hogan, Markus Krötzsch, and Axel Polleres. OWL: yet to arrive on the web of data? In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proc. of WWW 2012*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
- [GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proc. of Logic Programming 1988*, pages 1070–1080. MIT Press, 1988.
- [GLL09] Stefan Göller, Markus Lohrey, and Carsten Lutz. PDL with intersection and converse: satisfiability and infinite-state model checking. *J. Symb. Log.*, 74(1):279–314, 2009.
- [GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
- [HL711] HL7. FHIR specification (v5.0.0). 2011. <http://www.hl7.org/fhir/>.
- [HMS07] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reason.*, 39(3):351–384, 2007.
- [HN92] Pavol Hell and Jaroslav Nešetřil. The core of a graph. *Discret. Math.*, 109(1-3):117–126, 1992.
- [HST00] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Log. J. IGPL*, 8(3):239–263, 2000.
- [HSW01] Juraj Hromkovic, Sebastian Seibert, and Thomas Wilke. Translating regular expressions into small ϵ -free nondeterministic finite automata. *J. Comput. Syst. Sci.*, 62(4):565–588, 2001.
- [KAH11] Matthias Knorr, José Júlio Alferes, and Pascal Hitzler. Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.*, 175(9-10):1528–1554, 2011.
- [Kaz09] Yevgeny Kazakov. Consequence-driven reasoning for horn SHIQ ontologies. In Craig Boutilier, editor, *Proc. of IJCAI 2009*, pages 2040–2045, 2009.
- [KR15] Egor V Kostylev and Juan L Reutter. Complexity of answering counting aggregate queries over DL-Lite. *Journal of Web Semantics*, 33:94–111, 2015.

- [KRH07] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for horn description logics. In *Proc. of AAAI 2007*, pages 452–457. AAAI Press, 2007.
- [Krö12] Markus Krötzsch. OWL 2 profiles: An introduction to lightweight ontology languages. In Thomas Eiter and Thomas Krennwallner, editors, *Proc. of Reasoning Web 2012*, volume 7487 of *Lecture Notes in Computer Science*, pages 112–183. Springer, 2012.
- [KZA24] Jin Ke, Zenon G. Zacouris, and Maribel Acosta. Efficient validation of SHACL shapes with reasoning. *Proc. VLDB Endow.*, 17(11):3589–3601, 2024.
- [LAHS05] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *J. Artif. Intell. Res.*, 23:667–726, 2005.
- [LSR⁺20] Martin Leinberger, Philipp Seifer, Tjitze Rienstra, Ralf Lämmel, and Steffen Staab. Deciding SHACL shape containment through description logics reasoning. In Jeff Z. Pan, Valentina Tamma, Claudia d’Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *Proc. of ISWC 2020*, volume 12506 of *Lecture Notes in Computer Science*, pages 366–383. Springer, 2020.
- [LST05] Carsten Lutz, Ulrike Sattler, and Lidia Tendera. The complexity of finite model reasoning in description logics. *Inf. Comput.*, 199(1-2):132–171, 2005.
- [MHS07a] Boris Motik, Ian Horrocks, and Ulrike Sattler. Adding integrity constraints to OWL. In Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors, *Proc. of OWLED 2007*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
- [MHS07b] Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the gap between OWL and relational databases. In *Proc. of WWW 2007*. ACM, 2007.
- [MS87] David E. Muller and Paul E. Schupp. Alternating automata on infinite trees. *Theor. Comput. Sci.*, 54:267–276, 1987.
- [NKK⁺19] Charalampos Nikolaou, Egor V. Kostylev, George Konstantinidis, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. Foundations of ontology-based data access under bag semantics. *Artif. Intell.*, 274:91–132, 2019.
- [OGA26] Anouk M. Oudshoorn, Piotr Gorczyca, and Dörthe Arndt. Rewrite once, validate anywhere: Producing OWL-aware SHACL constraints. In *Proc. of ISWC 2026*, 2026.
- [OOŠ24] Anouk M. Oudshoorn, Magdalena Ortiz, and Mantas Šimkus. Reasoning with the core chase: the case of SHACL validation over $\mathcal{ELHI}$ knowledge bases. In Laura Giordano, Jean Christoph Jung, and Ana Ozaki, editors, *Proc. of DL 2024*, volume 3739 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024.

- [OŠ26a] Anouk M. Oudshoorn, Magdalena Ortiz, and Mantas Šimkus. SHACL validation in the presence of ontologies: Semantics and rewriting techniques. *Artif. Intell.*, 352:104483, 2026.
- [OOS26b] Anouk M. Oudshoorn, Magdalena Ortiz, and Mantas Simkus. Static analysis of recursive SHACL. In Renata Wassermann, Marie-Laure Mugnier, and Franz Baader, editors, *Proc. of KR 2026*, 2026.
- [ORŠ11] Magdalena Ortiz, Sebastian Rudolph, and Mantas Šimkus. Query answering in the horn fragments of the description logics SHOIQ and SROIQ. In Toby Walsh, editor, *Proc. of IJCAI 2011*, pages 1039–1044. IJCAI/AAAI, 2011.
- [Ort23] Magdalena Ortiz. A short introduction to SHACL for logicians. In Helle Hvid Hansen, Andre Scedrov, and Ruy J. G. B. de Queiroz, editors, *Proc. of WoLLIC 2023*, volume 13923 of *Lecture Notes in Computer Science*, pages 19–32. Springer, 2023.
- [OŠ12] Magdalena Ortiz and Mantas Šimkus. Reasoning and query answering in description logics. In *Proc. of Reasoning Web 2012*, volume 7487 of *Lecture Notes in Computer Science*, pages 1–53. Springer, 2012.
- [OŠ24] Cem Okulmus and Mantas Šimkus. SHACL validation under the well-founded semantics. In Pierre Marquis, Magdalena Ortiz, and Maurice Pagnucco, editors, *Proc. of KR 2024*, 2024.
- [OTMH23] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio José Rodríguez Méndez, and Armin Haller. Learning SHACL shapes from knowledge graphs. *Semantic Web*, 14(1):101–121, 2023.
- [Oud25] Anouk M. Oudshoorn. SHACL satisfiability: What can we learn from DLs? In *Proc. of DL 2025*, volume 4091 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2025.
- [Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
- [Par24] Paolo Pareti. SHACL2FOL: an FOL toolkit for SHACL decision problems. *CoRR*, abs/2406.08018, 2024.
- [PKM22] Paolo Pareti, George Konstantinidis, and Fabio Mogavero. Satisfiability and containment of recursive SHACL. *J. Web Semant.*, 74:100721, 2022.
- [PKMN20] Paolo Pareti, George Konstantinidis, Fabio Mogavero, and Timothy J. Norman. SHACL satisfiability and containment. In Jeff Z. Pan, Valentina Tamma, Claudia d’Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *Proc. of ISWC 2020*, volume 12506 of *Lecture Notes in Computer Science*, pages 474–493. Springer, 2020.

- [PMH10] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable query answering and rewriting under description logic constraints. *J. Appl. Log.*, 8(2):186–209, 2010.
- [Pra05] Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *J. Log. Lang. Inf.*, 14(3):369–395, 2005.
- [PSH13] Eric Prud’hommeaux, Andy Seaborne, and Steve Harris. SPARQL 1.1 Query Language. W3C Recommendation, 2013.
- [RLH23] Kashif Rabbani, Matteo Lissandrini, and Katja Hose. Extraction of validating shapes from very large knowledge graphs. *Proc. VLDB Endow.*, 16(5):1023–1032, 2023.
- [Rud11] Sebastian Rudolph. Foundations of description logics. In Axel Polleres, Claudia d’Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider, editors, *Reasoning Web 2011, Tutorial Lectures*, volume 6848 of *Lecture Notes in Computer Science*, pages 76–136. Springer, 2011.
- [Sch89] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *Proc. of KR 1989*, pages 421–431. Morgan Kaufmann, 1989.
- [SKL19] Ognjen Savkovic, Evgeny Kharlamov, and Steffen Lamparter. Validation of SHACL constraints over kgs with OWL 2 QL ontologies via rewriting. In Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J. G. Gray, Vanessa López, Armin Haller, and Karl Hammar, editors, *Proc. of ESWC 2019*, volume 11503 of *Lecture Notes in Computer Science*, pages 314–329. Springer, 2019.
- [SV01] Ulrike Sattler and Moshe Y. Vardi. The hybrid μ -calculus. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proc. of IJCAR 2001*, volume 2083 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2001.
- [Tam15] Kuniaki Tamura. A small model theorem for the hybrid μ -calculus. *J. Log. Comput.*, 25(2):405–441, 2015.
- [TCF05] Balder Ten Cate and Massimo Franceschet. Guarded fragments with constants. *J. Log. Lang. Inf.*, 14(3):281–288, 2005.
- [Tob00] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Intell. Res.*, 12:199–217, 2000.
- [Tob01] Stephan Tobies. *Complexity results and practical algorithms for logics in knowledge representation*. PhD thesis, RWTH Aachen University, Germany, 2001.

- [Var82] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proc. of Theory of Computing 1982*, pages 137–146. ACM, 1982.
- [VG93] Allen Van Gelder. The alternating fixpoint of logic programs with negation. *J. Comput. Syst. Sci.*, 47(1):185–221, 1993.
- [VGRS91] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.
- [W3C12] W3C. Owl 2 web ontology language. *Technical Report.*, 2012.
- [W3C17] W3C. Shape constraint language (SHACL). *Technical Report.*, 2017.