# Consequence-Based Reasoning for the Description Logic $\mathcal{SROIQ}$




David Jaime Tena Cucala

Hertford College

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Michaelmas 2019

# Abstract

Consequence-based (CB) reasoners combine ideas from resolution and (hyper)tableau calculi to solve the problem of ontology classification in Description Logics (DLs). Existing CB reasoners, however, are only capable of handling DLs without nominals (such as $\mathcal{SRIQ}$), or DLs without disjunction (such as Horn-$\mathcal{SROIQ}$). In this thesis, we present a novel CB calculus for concept subsumption in the DL $\mathcal{ALCHOIQ}^+$, a logic that extends $\mathcal{ALC}$ with role hierarchies, inverse roles, number restrictions, and nominals. To the best of our knowledge, ours is the first CB calculus for an NExpTime-complete DL. By using standard transformations, our calculus extends to $\mathcal{SROIQ}$, which covers all of OWL 2 DL except for datatypes. A key feature of our calculus is its pay-as-you-go behaviour: the calculus is worst-case optimal for many well-known fragments of $\mathcal{ALCHOIQ}^+$, including $\mathcal{ELH}$, Horn-$\mathcal{ALCHOIQ}^+$, and $\mathcal{ALCHIQ}^+$. Furthermore, the calculus is worst-case time exponential for the full logic $\mathcal{ALCHOIQ}^+$, except for cases where nominals, inverse roles, and number restrictions interact simultaneously, which are very rare in practice. Our calculus can also decide DL reasoning problems other than subsumption, such as ontology classification, instance retrieval, and realisation. We have implemented our calculus as an extension of the reasoner Sequoia, which previously supported ontology classification for $\mathcal{SRIQ}$. Our implementation includes novel optimisation techniques that overcome some of the performance limitations affecting the previous version. We have carried out an empirical evaluation of our implementation which shows that the performance of Sequoia is competitive with other state-of-the-art systems. Our results also show that Sequoia nicely complements tableau-based reasoners, as it can more easily classify some ontologies. Thus, the calculus and implementation presented in this thesis provide an important addition to the repertoire of reasoning techniques and practical systems for expressive DLs.

# Acknowledgements

My gratitude goes first and foremost to Professor Bernardo Cuenca Grau and Professor Ian Horrocks. From the first day of my D.Phil, their unsparing advice has helped me navigate the program, learn the ropes, and avoid pitfalls. Furthermore, their kind words of encouragement have rekindled my confidence when I most needed it. I thank in particular Prof. Cuenca Grau for unfailingly meeting with me every week and regularly giving feedback on my progress. Prof. Cuenca Grau has patiently taught me how to improve my writing skills and how to deal efficiently with red tape. I also thank especially Prof. Horrocks for helping me keep in mind the big picture and for his wise advice on how to prioritise tasks.

I am very grateful to the members of the KRR group in the Department of Computer Science, both current and former. Our daily lunches, cake parties in office 309, and profound or silly conversations over coffee breaks have been a staple of my D.Phil life. A special mention to Dr. Robert Piro, who helped me hit the ground running; Prof. Boris Motik, who patiently taught me good practices for working in tandem; Dr. Babis Nikolaou, who was a steadfast source of support on both personal and academic matters, and Dr. Andrew Bate, whose systematically documented code made my life much easier. Last, but not least, a shot out to my D.Phil colleagues: Alessandro Ronca, Alina Petrova, Pan Hu, Temitope Ajileye, and Federico Igne; their invaluable companionship made every day a little bit brighter.

This work would not have been possible without the backing of Dr. Edward Grefenstette and Hertford College, who jointly funded my studies through the Oxford – Mary Jane Grefenstette Scholarship.

I express my sincere appreciation for my friends at Hertford College, Balliol College, and in other places all over the world: many of my fondest memories of these years have been thanks to you. I am also indebted to Let's Dub Project, Mr. Oda, and Rooster Teeth, who provided a much-needed means of escape for me every now and then.

Finally, my biggest thanks are for my family and Maria, for their boundless love, support, and encouragement. This work is dedicated to you.

# Contents

**1 Introduction** **1**
1.1 Contributions . . . . . 5
1.2 Structure and Outline . . . . . 7

**I Foundations** **11**

**2 Preliminaries** **12**
2.1 Many-Sorted Clausal Equational Logic . . . . . 12
2.2 Orderings . . . . . 14
2.3 Rewrite systems . . . . . 15
2.4 Description Logics . . . . . 16
2.4.1 Syntax . . . . . 17
2.4.2 Semantics . . . . . 19
2.4.3 Inference Problems . . . . . 20
2.5 Clausification . . . . . 23

**3 Consequence-Based Reasoning** **26**
3.1 Evolution of CB Reasoning . . . . . 26
3.1.1 First CB Algorithms: the $\mathcal{EL}$ Family . . . . . 26
3.1.2 Extending CB Reasoning to Horn-$\mathcal{SHIQ}$ . . . . . 28
3.1.3 Reasoning with Nominals: Horn-$\mathcal{SROIQ}$ and $\mathcal{ELHO}$ . . . . . 30
3.1.4 Moving Beyond Horn DLs . . . . . 32
3.2 The CB Calculus for $\mathcal{ALCHIQ}^{+}$ . . . . . 34
3.3 Hybrid Methods . . . . . 37

**II Calculus and Applications** **39**

**4 Extending Consequence-Based Reasoning to $\mathcal{ALCHOIQ}^+$** **40**
4.1 Key Features of CB Reasoning . . . . . 40
4.1.1 "Materialisation" of Logical Consequences . . . . . 40
4.1.2 One-Pass Decision of Multiple Subsumptions . . . . . 43
4.1.3 Focus on Local Consequences . . . . . 44
4.1.4 Goal-Oriented Behaviour . . . . . 45
4.2 Reconciling Locality with Nominals . . . . . 47
4.3 Interaction of $\mathcal{I}$, $\mathcal{Q}$, and $\mathcal{O}$ . . . . . 49

**5 The Consequence-Based Calculus for $\mathcal{ALCHOIQ}^+$** **51**
5.1 Context Structure . . . . . 51
5.2 Inference Rules . . . . . 54
5.3 Soundness and Completeness . . . . . 58
5.4 Variant of the Calculus for Horn Ontologies . . . . . 61
5.5 Examples . . . . . 63

**6 Key Properties of the Calculus** **69**
6.1 Existence of Context Orders . . . . . 69
6.2 Proof of Soundness . . . . . 72
6.3 Proof of Completeness . . . . . 76
6.3.1 Proof Overview . . . . . 77
6.3.2 The Model Fragment $R_t^*$ . . . . . 80
6.3.3 Naming of Nominal-like Elements . . . . . 99
6.3.4 The Composite Model . . . . . 116
6.4 Proofs for the Variant of the Calculus . . . . . 130

**7 Solving DL Reasoning Problems Using the Calculus** **134**
7.1 General Algorithm for Subsumption Checking . . . . . 135
7.2 Modified Algorithm for Horn Ontologies and Single Subsumptions . . 139
7.3 Discussion . . . . . 141
7.3.1 Relation to CB Calculi for $\mathcal{ELHO}$ and Horn-$\mathcal{SROIQ}$ . . . . . 141
7.3.2 Relation to Tableau and Resolution Calculi for $\mathcal{SHOIQ}$ . . . 143
7.4 Proofs of Termination & Pay-as-you-go Behaviour . . . . . 144

**III Implementation** **148**

**8 System Description** **149**
8.1 Overview of Sequoia v.0.6 . . . . . 149
8.2 Implementing Support for Nominals . . . . . 153
8.2.1 Representation and Ordering of Terms and Literals . . . . . 153
8.2.2 Fragmentation of the Root Context . . . . . 154
8.2.3 Implementation of Inference Rules . . . . . 155
8.3 Optimisations . . . . . 156
8.3.1 Ordering of Query Atoms . . . . . 156
8.3.2 Controlled Propagation of the ABox . . . . . 157

**9 Evaluation** **159**
9.1 Methodology . . . . . 159
9.2 Results and Discussion . . . . . 162

**10 Conclusions and Future Work** **166**
10.1 Summary . . . . . 166
10.2 Future Work . . . . . 168

**A Survey and Comparison of Consequence-Based Calculi** **171**
A.1 The Calculi for the $\mathcal{EL}$ Family . . . . . 171
A.2 The Calculus for Horn-$\mathcal{SHIQ}$ . . . . . 174
A.3 The Calculus for $\mathcal{ELHO}$ . . . . . 176
A.4 The Calculus for Horn-$\mathcal{SROIQ}$ . . . . . 178
A.5 The Calculus for $\mathcal{ALCH}$ . . . . . 182
A.6 The Calculus for $\mathcal{ALCHI}$ . . . . . 183

**Bibliography** **190**

# Chapter 1

# Introduction

Description Logics (DLs) [5] are a prominent family of knowledge representation formalisms corresponding to decidable fragments of first-order logic. A DL *ontology* describes a domain of interest in terms of *concepts* (sets of entities), *roles* (relations between entities), and *individuals* (named entities).

DLs derive from early theories about how intelligent systems acquire, store, and use knowledge [103]. In the 60's and 70's, the prevalent view was that intelligent systems hold internal symbolic representations of real-world facts and entities. Inspired by this idea, two prominent paradigms emerged: *semantic networks* [110] and *frames* [91]. Both approaches propose symbolic formalisms that describe an actual or imagined reality in terms of classes of individuals and their relations. Early proposals lacked formal semantics, however, which was perceived as a shortcoming. Later iterations focused on closing this gap, and eventually both approaches converged to KL-ONE [26], a system that is considered to be the precursor of DLs.

In the last three decades, interest has shifted away from the general question of how intelligent systems represent knowledge, towards the design of practical languages for capturing real-world information in current software systems. Throughout this evolution, DLs have stayed at the forefront of knowledge representation research. Presently, they provide a rich repertoire of languages for ontology design and management, which offer different trade-offs between complexity and expressive power. DLs have been applied in many fields, including agriculture [129], astronomy [38], defence [81], medicine [141], and mining [77]. They also provide the formal underpinning for the Web Ontology Language (OWL 2) [108], a family of languages established as a Semantic Web standard by the World Wide Web Consortium (W3C). For instance, the highly expressive DL $\mathcal{SROIQ}$ [55] corresponds to OWL 2 DL [106], one of the most general languages in this family.

*Ontology reasoners* provide automated support for ontology-based modelling and query answering. Ontologies are hand-crafted by experts or learned automatically [83, 143, 29, 114], and reasoning algorithms can be used to check their consistency and repair any contradictions or undesired consequences [10]. Furthermore, reasoners can be used to uncover implicit knowledge [9] and produce a taxonomy of the concept names mentioned in the ontology. DL reasoning also has applications in data management [64]. For instance, ontologies can be used to enrich data by encoding global complex constraints that augment the explicit assertions in a database [32]. DLs can also be used to facilitate access to data, by providing a user-friendly query language that can interface with underlying relational databases [112, 78]. Finally, the fact that DLs provide a vocabulary with formally, well-defined semantics makes them ideal for projects of data integration [30, 31].

A key reasoning service for DLs is *concept subsumption*: given an ontology $\mathcal{O}$ and concepts $C$ and $D$, determine whether each instance of $C$ is also an instance of $D$ in every model of $\mathcal{O}$. *Ontology classification* is the problem of computing all subsumption relationships between pairs of concept names in an ontology. Concept subsumption in $\mathcal{SROIQ}$ is an N2ExpTime-complete problem; however, despite such high complexity, practical calculi for this logic and other related DLs have been developed and successfully implemented.

*Tableau* and *hypertableau* calculi [21] underpin many of the currently available ontology reasoners, including FaCT++ [139], RacerPro [50], Pellet [126], and HermiT [121]. To check whether a subsumption is entailed, these calculi attempt to construct a finite representation of a countermodel – that is, a model of the ontology in which the subsumption does not hold. The first tableau-based algorithm was devised by Schmidt-Schauß and Smolka for the DL $\mathcal{ALC}$ [118], and it was quickly followed by the work of Donini et al. [39], who noted the similarities between this approach and the so-called *tableau* methods for first-order logic [128].

Tableau-based algorithms face two main sources of complexity [97]: a need for frequent backtracking due to the non-determinism introduced by disjunctive statements (*or-branching*), and the generation of very large models due to the need to satisfy existential restrictions (*and-branching*). These problems are exacerbated in tasks such as classification, where a large number of subsumptions must be checked. Notwithstanding these limitations, tableau-based calculi tend to perform well in practice, and the study of practical tableau-based calculi has been a very active area of research. Tableau algorithms have been developed for the DL $\mathcal{ALC}$ with ABoxes [52], $\mathcal{ALCN}$ [53, 13], $\mathcal{SHIQ}$ [61], $\mathcal{SHOQ}$ [56], and finally $\mathcal{SHOIQ}$ [60]. The last of these

calculi was further developed into a *hypertableau* procedure [97], which is based on the hypertableau calculus of Baumgartner [22]. Compared with tableau calculi, the hypertableau approach reduces the inefficiencies of non-determinism associated with disjunction, and this often translates into better performance.

Tableau-based algorithms are typically not worst-case optimal, but they often include several techniques and optimisations to reduce the impact of and-branching and or-branching. Furthermore, these calculi attempt to reduce the number of subsumption checks during classification using methods such as the *enhanced traversal* algorithm [6, 46, 145], or modular decomposition [2, 88]. The technique of *global caching* has been used to ensure worst-case optimal behaviour in ExpTime complete DLs [98, 99, 100]; to the best of our knowledge, however, this technique has not been applied to DLs that simultaneously support disjunction, number restrictions, inverse roles, and nominals. Attempts to devise a worst-case optimal tableau algorithm for $\mathcal{SHOIQ}$ remain incomplete [82]. Despite these optimisations, the aforementioned sources of complexity can still compromise the performance of tableau-based reasoners on complex ontologies.

Techniques based on *first-order resolution* constitute another category of DL reasoning calculi. These algorithms transform an ontology into a set of first-order clauses, and then attempt to derive the empty clause using a first-order resolution-based inference system [15]. DLs have been closely related to first-order logic since their inception; this is because their original architects drew inspiration from first-order logic to define their semantics [24]. Later, it was shown that many expressive DLs can be seen as a fragment of the two-variable fragment $\mathcal{FO}^2$ of first-order logic [23], which is known to be NExpTime [92], or the fragment $\mathcal{C}^2$, which extends $\mathcal{L}^2$ with counting quantifiers – in the case of DLs with number restrictions; this language is also decidable, even with binary encoding of numbers [109]. Similarly, Schmidt [116] and Areces [1] showed how to reduce DLs to decidable fragments of first-order logic for which theorem provers can be used to decide satisfiability.

A standard problem which emerges when translating DLs to fragments of first-order logic is that the target language tends to have a higher complexity than is required. For instance, satisfiability in $\mathcal{L}^2$ is NExpTime-complete, which is much higher than the PSpace-complete complexity of $\mathcal{ALC}$. This obstacle requires most resolution-based calculi for DLs to define additional constraints on the inference rules to avoid deriving too many clauses that are irrelevant for the satisfiability proof. A common approach to ensuring termination and worst-case optimal behaviour while preserving completeness is constraining the application of resolution inferences using

carefully defined term orderings and selection functions [36, 67, 117, 65, 66, 43, 75, 63]. The reasoner KAON2 [94] implements this approach for the DL $\mathcal{SHIQ}$ [93], and it was later extended to $\mathcal{SHOIQ}$ [76]. Resolution has also been used to simulate model-building (hyper)tableau techniques [62], including blocking methods which ensure termination [45].

It has been observed that resolution-based procedures, even with carefully defined selection functions and orderings, may still generate a large number of clauses which are irrelevant for deriving the empty clause in some cases [20]. This is a significant source of performance problems, especially for large and complex ontologies. For instance, the KAON2 reasoner works well in ontologies with relatively simple TBoxes, even when coupled with large ABoxes [96], as it relies on a reduction of expressive knowledge base to a Disjunctive Datalog program [63]. In ontologies with large TBoxes, however, this reduction can lead to an infeasible blow-up in the size of the TBox, which in turn creates a situation where most inferred clauses are irrelevant, and hence the method becomes effectively non-terminating in many cases. In general, resolution-based methods have not achieved the same widespread use as their tableau counterparts [54].

*Consequence-based* (CB) calculi emerged as a promising approach to DL reasoning combining ideas from both tableau and resolution. On the one hand, similarly to resolution, CB calculi derive formulae entailed by the ontology, thus avoiding the explicit construction of large models, and they are typically worst-case optimal. On the other hand, clauses are organised into "contexts" arranged in a graph structure reminiscent of that used in (hyper)tableau algorithms for representing a model; this yields a nice goal-oriented behaviour and prevents CB calculi from drawing many unnecessary inferences. Furthermore, in contrast to both resolution and (hyper)tableau, CB calculi can verify a large number of subsumptions in a single execution, allowing for one-pass classification. CB calculi are very effective in practice, and systems based on this approach, such as ELK [74] and Snorocket [89], have shown outstanding performance.

The first CB calculi were developed for the lightweight $\mathcal{EL}$ family of DLs [27, 4, 72]. They originally received the name of "consequence-based" because, in contrast to the dominant tableaux-based paradigm, CB calculi proceed by deriving logical consequences of the ontology. As we have seen above, however, this is not their only defining feature. CB calculi have been extended to the more expressive logics Horn-$\mathcal{SHIQ}$ [69], Horn-$\mathcal{SROIQ}$ [105], $\mathcal{ALCH}$ [123], and $\mathcal{SRIQ}$ [20], and they have also been used in combination with other reasoning methods. For instance, the CB calculus by

Vlasenko et al. [142] for the DL $\mathcal{ELQ}$ relies on an Integer Linear Programming solver to efficiently deal with number restrictions; this approach has been more recently extended to the DL $\mathcal{SHOQ}$ in [68] and implemented in the reasoner Avalanche [142], with encouraging results. The $\mathcal{SROIQ}$ reasoner Konclude [132] is primarily based on a tableau calculus, but it exploits an incomplete CB calculus to efficiently derive as many entailments as possible. The CB calculus for $\mathcal{ALCHI}$ by Kazakov et al. [71] introduces a non-deterministic rule to handle disjunction, which is similar to the disjunction rule used in tableaux. The reasoner MORe [2] exploits module extraction techniques [35] to divide the workload of ontology classification between a CB reasoner for a lightweight DL, and a tableau-based reasoner for $\mathcal{SROIQ}$.

To the best of our knowledge, however, no CB calculus currently handles DLs supporting simultaneously all Boolean connectives, inverse roles, number restrictions, and nominals. Furthermore, it is well known that the interaction of these features causes a complexity jump from ExpTime [86, 51, 138] to NExpTime [85] for subsumption checking, and poses major practical challenges for reasoning algorithms [59, 76]. Furthermore, the presence of nominals in the ontology creates additional difficulties that are specific to the CB approach: nominals can be used to express global constraints on models of the ontology, which are difficult to represent within the CB framework [72].

## 1.1 Contributions

The main contribution of this thesis is the development and implementation of the first CB calculus for deciding subsumption in the DL $\mathcal{SHOIQ}$, which supports all Boolean connectives, inverse roles, number restrictions, and nominals. By encoding role inclusion axioms using well-known methods [55, 122], our calculus can be extended to the DL $\mathcal{SROIQ}$ [55], which covers all of OWL 2 DL except for datatypes, albeit the input can suffer an exponential blow-up in size. Key features of this calculus are:

- **Multiple Queries in One Pass**: similarly to other CB procedures, the calculus presented in this thesis can decide multiple subsumption queries in a single run; in particular, the calculus is capable of one-pass classification.
- **Reusability**: the calculus can re-use the consequences that it has derived while answering a query to reduce the computation time of any subsequent queries. In some cases, it may even directly provide the answer without performing

additional computations. This holds even when the ontology is extended with new axioms between one query and the next.

- **"Pay-as-you-go" behaviour**: the calculus is worst-case optimal for proper fragments of the logic $\mathcal{SHOIQ}$; furthermore, for some of these logics the calculus simulates previously existing CB procedures. Although the calculus is not worst-case optimal for the full logic, it remains worst-case exponential time, except for ontologies where nominals simultaneously interact with number restrictions, inverse roles, and disjunction. These ontologies are assumed to be rare in practice, however, and as we will see in Chapter 9, no such interaction was found during our extensive evaluation.

- **Practicality**: the calculus is amenable to practical implementation. The combination of features from (hyper)tableau and resolution produces compact representations of models and a nice goal-oriented behaviour. Furthermore, the calculus supports ordered resolution, which reduces the number of inferences required to reach saturation; the calculus also uses a notion of redundancy to eliminate irrelevant clauses. Finally, the derived consequences are distributed across *contexts*; many inference rules of the calculus can be applied to each context independently of the state of other contexts, which makes the calculus suitable for parallel reasoning.

The second contribution of this thesis is the practical evaluation of our calculus, which we have implemented as an extension of the reasoner Sequoia [20], originally introduced to handle classification for $\mathcal{SRIQ}$ ontologies. We extend Sequoia to support nominals, and we implement additional DL reasoning tasks, such as consistency checking, instance retrieval, and realisation. Our results show that the reasoner offers competitive performance, and we find that there is little overlap between the set of ontologies where Sequoia times out, and the corresponding sets for tableau reasoners like HermiT. Therefore, Sequoia provides an important addition to the repertoire of practical implementation techniques for DL reasoning, and it nicely complements previously existing approaches.

Another contribution of this thesis is a collection of algorithmic optimisations that address some of the performance limitations of Sequoia. It has been shown, for instance, that the previous version of Sequoia struggles with ontology clauses that contain a large number of disjuncts [136]. Furthermore, the implementation of support for nominals also poses challenges for the performance of the reasoner. To

ameliorate these issues, the thesis presents a series of optimisations which have been implemented in the latest version of the reasoner.

Other contributions are:

- We present an analysis of the defining features of CB reasoning and we illustrate how they induce the particular advantages of this approach. This fills a gap in the literature, since CB calculi have historically been presented as extensions of the original procedure by Baader et al. [4], with little to no emphasis on the underlying philosophy of the approach.
- We show how to apply CB reasoning to the DL reasoning problems of instance retrieval and realisation; thus, we take the first steps towards a query-answering CB reasoning procedure.
- We carry out an empirical test to detect the number of ontologies in the Oxford Ontology repository that contain simultaneous interactions of nominals, inverse roles, and number restrictions. Our results confirm the widely-held belief that such occurrences are extremely rare.
- We present a step-by-step comparison of existing CB calculi, and we show the correspondences between inference rules of different calculi.

## 1.2 Structure and Outline

The thesis is structured in three parts: a review of preliminary literature, the presentation of our calculus for $\mathcal{ALCHOIQ}^+$, and the description and evaluation of its implementation. An additional chapter concludes the thesis.

In Part I, we lay the foundations for the contributions presented in this thesis. We recapitulate key definitions and results from prior literature, and we give an overview of the evolution and state-of-the-art of CB reasoning.

- In Chapter 2, we restate essential definitions and preliminary results for this thesis. Following Bate et al. [19], our calculus encodes derived consequences as clauses of two-sorted logic where equality is the only predicate; thus, the chapter starts with a summary of key notions of many-sorted clausal first-order logic, followed by a method to encode ordinary clauses in equational form. Next, we introduce Herbrand equality interpretations, which we will use as models of the ontology. We represent these interpretations as rewrite systems,

since this allows us to exploit techniques from paramodulation-based theorem proving to show completeness and establish complexity bounds; we therefore include next a section describing the formalism of rewrite systems and their basic properties. Afterwards, we recapitulate standard definitions of term orderings, which are useful for parameterising the calculus with ordered resolution and proving termination of rewrite systems. Finally, we define the DL $\mathcal{SROIQ}$ and give a procedure to normalise and clausify ontologies into a suitable input for our calculus.

- In Chapter 3, we present an in-depth review of the CB approach to DL reasoning. We outline the original CB calculus for $\mathcal{ELH}$ in [27], followed by a recounting of its subsequent extensions to more expressive logics like Horn-$\mathcal{SHIQ}$ [69], Horn-$\mathcal{SROIQ}$ [105], $\mathcal{ALCH}$ [124], $\mathcal{ALCHIQ}$ [18]; $\mathcal{ALCHOI}$, and $\mathcal{ALCHOQ}$ [68]. We also give a detailed overview of the CB calculus for $\mathcal{ALCHIQ}^+$ by Bate et al. [20], which serves as the basis for our calculus; this also allows us to introduce key elements of the framework for CB reasoning that we use in the remainder of the thesis.

In Part II, we introduce our calculus for $\mathcal{ALCHOIQ}^+$. We prove its soundness, completeness, termination, and its pay-as-you-go property, and we illustrate how it can be applied to solve various DL reasoning problems.

- In Chapter 4, we discuss the challenges of extending CB reasoning to a DL supporting all Boolean connectives, nominals, inverse roles, and number restrictions. We start the chapter with an analysis of the features which we believe to be essential of any CB calculus, and we explain why they are necessary to produce the advantageous properties of the CB approach. Next, we identify and discuss two main challenges which emerge when attempting to devise a calculus for $\mathcal{ALCHOIQ}^+$ which preserves those features. The first obstacle lies in combining the "local" nature of CB reasoning with the global constraints on ontology models that may be imposed by the use of nominals. The second obstacle is caused by the possibility of interactions between inverse roles, number restrictions, and nominals.

- In Chapter 5, we present our CB calculus for the DL $\mathcal{ALCHOIQ}^+$. The calculus is designed so that it seamlessly extends the previous procedure by Bate et al. [18]. To handle nominals, our calculus allows for ground atoms in derived clauses, it includes several new inference rules, and it introduces a new *root*

*context* where inferences on ground atoms are performed. We also introduce a variant of our calculus to achieve worst-case optimality in Horn ontologies with nominals. The chapter ends with two examples that demonstrate in detail how our calculus addresses the challenges described in the previous chapter.

- In Chapter 6, we present the proofs of the soundness and completeness claims from the previous chapter. Soundness is proved for each of the inference rules of the calculus, and it follows from the soundness of the hyperresolution rule. For completeness, we show that for every subsumption not appearing in a suitably initialised and saturated context structure, we can build a countermodel. To construct the model, we unfold the context structure into a forest-like Herbrand equality interpretation.

- In Chapter 7, we introduce a subsumption algorithm based on our calculus, and we show how it can solve consistency checking, general concept subsumption, classification, instance retrieval, and realisation. We also illustrate how variants of the calculus and the algorithm based on it can achieve worst-case optimal performance for proper fragments of $\mathcal{ALCHOIQ}^+$. In particular, we give deterministic exponential time procedures for all of the DLs $\mathcal{ALCHOI}$, $\mathcal{ALCHOQ}$, $\mathcal{ALCHIQ}^+$ and Horn-$\mathcal{ALCHOIQ}^+$, and polynomial procedures for the lightweight DLs $\mathcal{ELH}$ and $\mathcal{ELHO}$. We conclude the chapter with a discussion highlighting the similarities and differences between our calculus (and the algorithms based on it), and other CB calculi for DLs with nominals, as well as with resolution and tableau calculi for $\mathcal{SHOIQ}$.

In Part III, we describe the implementation of our calculus as an extension of the $\mathcal{SRIQ}$ CB reasoner Sequoia [20]. We also present a collection of algorithmic optimisations used in the latest version of the reasoner, and we report on our evaluation of the system performance.

- In Chapter 8, we discuss in detail all modifications and novel features with respect to the previous version of Sequoia. The newest version supports nominals and reasoning tasks other than classification. We also present a collection of algorithmic optimisations that improve the performance of the system. Some of the optimisations address the limitations already present in the previous version of Sequoia, while others are critical for performance on ontologies with nominals.

- In Chapter 9, we describe the results of our extensive experimental evaluation of Sequoia on a large corpus of real-world ontologies. We compare the ontology classification times of Sequoia with those of state-of-the-art reasoners Pellet, FaCT++, HermiT, and Konclude. Our experiments show that Sequoia offers competitive performance. Furthermore, we observe that Sequoia and HermiT time out on sets of ontologies that overlap very little with each other. This shows that Sequoia is not only competitive, but it nicely complements systems based on tableau algorithms. We also give empirical evidence of the effect of the optimisations described in the previous chapter, and we discuss variations of performance across some relevant partitions of the corpus, e.g. ontologies with and without nominals.

We conclude the thesis in Chapter 10, where we recapitulate our key contributions, discuss their significance, and outline possible avenues of research for future work.

Much of the content presented in this thesis has already been published in different outlets, which are listed as follows.

- The first CB calculus for a DL with disjunction, inverse roles, and nominals was presented in DL 2017 [134].
- The full $\mathcal{ALCHOIQ}^+$ calculus was presented in IJCAI 2018 [135].
- The implementation of the calculus, together with its evaluation, was exhibited in DL 2019 [34].
- An extension of the results presented in the three publications listed above is currently under review by AIJ.
- A thorough review of the literature in CB reasoning and the analysis of common features has been included in a chapter of the book *Description Logic, Theory Combination, and All That*, published in 2019 [136].
- Several evaluation methods and proofs techniques are derived from the author's contribution to the JAIR article *Consequence-Based Reasoning for Description Logics with Disjunctions and Number Restrictions* [20].

# Part I

# Foundations

# Chapter 2

# Preliminaries

In this chapter, we recapitulate essential definitions and preliminary results that will be used in the remainder of this thesis. In Section 2.1, we recount basic notions of many-sorted first-order equational logic; in Section 2.2, we provide basic definitions related to term orders, and in Section 2.3, we introduce rewrite systems. Next, in Section 2.4, we overview the basics of Description Logics and their inference problems. Finally, in Section 2.5, we define a clausal form for DL ontologies, which we will use as input for the calculus.

## 2.1 Many-Sorted Clausal Equational Logic

In this section we review the basic notions of many-sorted clausal equational logic. Following Bate et al. [19], we assume that equality is the only predicate; this is without loss of generality since first-order atoms with arbitrary predicates can be transformed into equality atoms in a way that preserves satisfiability and entailment [101, 102].

A *many-sorted* signature $\Sigma$ is a pair $(\Sigma^S, \Sigma^F)$, where $\Sigma^S$ is a non-empty set of *sorts*, and $\Sigma^F$ is a countable set of *function symbols*, where each function symbol $f \in \Sigma^F$ is associated to a *symbol type*: an expression of the form $\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}$, where $n \in \mathbb{N}_0$ is the *arity* of $f$, $\mathsf{s_i} \in \Sigma^S$ for each $1 \leq i \leq n$, and $\mathsf{s} \in \Sigma^S$ is the *sort* of $f$. Function symbols of arity 0 are called *constants*. For the remainder of this section, we fix an arbitrary many-sorted signature $\Sigma = (\Sigma^S, \Sigma^F)$, which we will use in all definitions.

For each sort $\mathsf{s} \in \Sigma^S$, let $\mathcal{X}_\mathsf{s}$ be a countably infinite set of *variables* of sort $\mathsf{s}$, and suppose that sets of variables of different sorts are disjoint. If $\mathsf{s} \in \Sigma^S$, the set of *terms* of sort $\mathsf{s}$ is the smallest set containing (i) each variable in $\mathcal{X}_\mathsf{s}$ and (ii) each expression $f(t_1, \ldots, t_n)$, where $n \in \mathbb{N}_0$, $t_i$ is a term of sort $\mathsf{s}_i$ for $1 \leq i \leq n$, and $f$ is a function symbol of type $\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}$. A term of sort $\mathsf{s}$ is also called an $\mathsf{s}$-*term*. A *position* $p$

of a term is a finite sequence of positive integers; we represent the empty position as $\epsilon$ and a non-empty position as $i_1 \cdot i_2 \cdot \ldots \cdot i_n$, where $i_j$ is the $j$-th element in the sequence, for $1 \leq j \leq n$. A position $p$ is *proper* if $p \neq \epsilon$. Given a term $t = f(t_1, \cdots, t_n)$, the *subterm of* $t$ *at position* $p$, represented $t|_p$, is defined inductively as follows: if $p = \epsilon$, then $t|_p = t$; otherwise, $p$ is of the form $i \cdot p'$, where $i \in \mathbb{N}$ and $p'$ is a position, and $t|_p$ is defined as $t_i|_{p'}$ if such subterm exists, and is undefined otherwise. If $t_2$ is a subterm of $t_1$ at some position $p$, we say that $t_1$ *mentions* or *has an occurrence of* $t_2$. A term is *ground* if it does not mention any variable. If $s_1$ is an $\mathsf{s}$-subterm of term $t$ at position $p$, and $s_2$ is an $\mathsf{s}$-term, then $t[s_2]_p$ denotes the term obtained when replacing $s_1$ by $s_2$ at position $p$ of $t$. Furthermore, we denote by $t[s_1/s_2]$ the result of replacing every occurrence of $s_1$ in $t$ by $s_2$.

An *equality* is an expression of the form $t_1 \approx t_2$, where $t_1$ and $t_2$ are terms of the same sort. An *inequality* $t_1 \not\approx t_2$ is the negation of an equality $t_1 \approx t_2$. Equalities and inequalities are symmetric; that is, expression $t_1 \bowtie t_2$ is identical to $t_2 \bowtie t_1$ for $\bowtie \in \{\approx, \not\approx\}$. A *literal* is either an equality or an inequality. A literal, a conjunction of literals, or a disjunction of literals is *ground* if each of its terms is ground.

We can express ordinary first-order logic atoms in a signature where equality is the only predicate by assuming existence of a distinguished sort $\mathsf{p} \in \Sigma^S$ such that, for every function symbol $A \in \Sigma^F$ of type $\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{p}$, we have $\mathsf{s}_i \neq \mathsf{p}$ for each $i$ with $1 \leq i \leq n$; furthermore, we assume that $\mathsf{true}$ is a $\mathsf{p}$-constant. We refer to function symbols of sort $\mathsf{p}$ as *predicates*; equalities of the form $A(t_1, \ldots, t_n) \approx \mathsf{true}$ then play a role analogous to first-order atoms $A(t_1, \ldots, t_n)$. Given an arbitrary formula in standard first-order logic, replacing standard atoms by equality atoms in this way preserves entailment and satisfiability [102]. When the meaning is clear from the context, we abbreviate an equality $A(t_1, \ldots, t_n) \approx \mathsf{true}$, where $A$ is a predicate, to the expression $A(t_1, \ldots, t_n)$.

A *clause* is an expression of the form $\forall x_1, \ldots, x_n. (\Gamma \to \Delta)$ with $n \in \mathbb{N}_0$, where the *body* of the clause $\Gamma$ is a finite conjunction of equalities, the *head* of the clause $\Delta$ is a finite disjunction of literals, and expression $x_1, \ldots, x_n$ contains all variables occurring in $\Gamma \to \Delta$. The quantifier $\forall x_1, \ldots, x_n$ in clauses is omitted for brevity. Conjunctions and disjunctions are treated as sets (we assume that they are unordered and without repetition), and we use set operations to manipulate them; note that this is a simpler representation of clauses than the standard bag representation in resolution theorem proving [14]. The empty conjunction is represented as $\top$, and the empty disjunction is represented as $\bot$. A clause is *ground* if the body and the head of the clause are ground.

A set of clauses $N$ *contains a clause* $\Gamma \to \Delta$ *up to redundancy* (abbreviated u.t.r.) if and only if (i) there is a term $s$ such that $s \approx s \in \Delta$, (ii) there is a pair of terms $s, t$ such that $\{s \approx t, s \not\approx t\} \subseteq \Delta$, or (iii) there exists a clause $\Gamma' \to \Delta' \in N$ such that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$. If $N$ contains $\Gamma \to \Delta$ u.t.r., we write $\Gamma \to \Delta \,\hat{\in}\, N$. This notion of redundancy is taken from [20]. Given a clause $\Gamma \to \Delta$, if either (i) or (ii) hold, we say that $\Delta$ *contains a tautology*, and we refer to $s \approx s$ or $s \approx t \vee s \not\approx t$ (respectively) as a *tautology* of $\Delta$. Furthermore, we say that a clause $\Gamma' \to \Delta'$ *subsumes* $\Gamma \to \Delta$ if $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$, and we say that $\Gamma' \to \Delta'$ is a clause *subsuming* $\Gamma \to \Delta$.

A *substitution* is a mapping $\sigma$ from variables to terms such that each variable $x \in \mathcal{X}_{\mathsf{s}}$ is mapped to a term of sort $\mathsf{s}$, and all but finitely many variables are mapped to themselves by $\sigma$. The image of variable $x$ by $\sigma$ is represented as $x\sigma$. Substitution $\sigma$ is often represented as the set $\{x_1 \mapsto t_1, \cdots, x_n \mapsto t_n\}$, where $t_i = x_i\sigma$ for each $1 \leq i \leq n$, and the *domain* $\{x_1, \ldots, x_n\}$ of $\sigma$ consists of all variables not mapped to themselves by $\sigma$. If $\alpha$ is either a term, a conjunction, a disjunction, or a clause, then $\alpha\sigma$ is the result of replacing every occurrence of $x$ in $\alpha$ by $x\sigma$. Finally, a substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is *ground* if each $t_i$ is a ground term.

The *Herbrand Universe* HU is the smallest set containing all ground terms of sort $\mathsf{s}$ for each $\mathsf{s} \in \Sigma^S$. We assume that for each $\mathsf{s} \in \Sigma^S$ there is at least one ground $\mathsf{s}$-term. A *Herbrand equality interpretation* $\mathcal{I}$ is a set of equalities between terms of HU satisfying the following properties: (i) reflexivity, (ii) symmetry, (iii) transitivity, and (iv) if $t|_p \approx s \in \mathcal{I}$, then $t \approx t[s]_p \in \mathcal{I}$. Let $\alpha$ be a ground conjunction, a ground disjunction, a (not necessarily ground) clause, or any set thereof; we say that $\mathcal{I}$ *satisfies* $\alpha$ if it does so according to the usual criteria, but assuming that each universally quantified variable of sort $\mathsf{s} \in \Sigma^S$ ranges only over $\mathsf{s}$-terms in HU. We say that $\mathcal{I}$ is a *model* of $\alpha$, written $\mathcal{I} \models \alpha$, if $\mathcal{I}$ satisfies $\alpha$. A set of clauses $\alpha$ *entails* a clause $\Gamma \to \Delta$, written $\alpha \models \Gamma \to \Delta$, if each model of $\alpha$ satisfies $\Gamma \to \Delta$.

## 2.2 Orderings

A *strict (partial) order* on a set $X$ is a transitive and irreflexive binary relation on $X$. A strict order is *well-founded* if, for each non-empty subset $Y \subseteq X$, there exists $a \in Y$ such that no $b \in Y$ satisfies $a \succ b$. A strict order $\succ$ induces a *non-strict (partial) order* $\succeq$ by taking the reflexive closure of $\succ$. A *total* order $>$ on $X$ is a strict order on $X$ such that, given any two distinct elements $a, b \in X$, either $a > b$ or $b > a$. For any $\circ \in \{\succ, \succeq, >\}$, we have that if $a \in X$ and $Y \subseteq X$, then $a \circ Y$ if and only if $a \circ b$

for every $b \in Y$. It is well-known that every partial order can be extended to a total order.

A *multiset* over a set $X$ is a function $\mu : X \to \mathbb{N}_0$. A multiset $\mu$ is *finite* if there is a finite $Y \subseteq X$ such that $a \notin Y$ implies $\mu(a) = 0$, for each $a \in X$. The *difference* $\mu_1 \backslash \mu_2$ between multisets $\mu_1$ and $\mu_2$ over $X$ is the multiset defined as $\mu_1 \backslash \mu_2(a) = \max(0, \mu_1(a) - \mu_2(a))$. The *multiset extension* of an order $\circ \in \{\succ, \succeq, >\}$ on $X$ is the order $\circ_{mul}$ on all multisets over $X$ defined as: $\mu_1 \circ \mu_2$ if and only if $\circ \in \{\succ, >\}$ implies $\mu_1 \neq \mu_2$, and for every $a \in \mu_2 \backslash \mu_1$ there exists $b \in \mu_1 \backslash \mu_2$ with $b \circ a$.

A *term order* is an order $\circ \in \{\succ, \succeq, >\}$ on HU. A term order $\circ$ induces an order (also represented as $\circ$) on literals by treating each literal $t_1 \approx t_2$ as the multiset $\{t_1, t_2\}$, each literal $t_1 \not\approx t_2$ as the multiset $\{t_1, t_1, t_2, t_2\}$, and comparing literals using the multiset extension of $\circ$. A strict term order $\succ$ is *monotonic* if, for each pair $s_1 \succ s_2$, each term $t \in \mathrm{HU}$, and each proper position $p$ in $t$, we have $t[s_1]_p \succ t[s_2]_p$. A monotonic order is *stable* under substitutions if $s_1 \succ s_2$ implies $s_1\sigma \succ s_2\sigma$ for any substitution $\sigma$. A *rewrite* order is a monotonic order that is stable under substitutions. A *reduction* order is a well-founded rewrite order. A strict term order has the *subterm property* if, for each term $s \in \mathrm{HU}$ and each proper position $p$ in $s$, we have $s \succ s|_p$. A *simplification* order is a reduction order with the subterm property.

Let $\succ$ be an ordering on a set of function symbols. The *lexicographic path ordering* $\succ_{lpo}$ induced by $\succ$ is defined as follows. Given terms $s$ and $t$ of the form $f(s_1, \ldots, s_m)$ and $g(t_1, \ldots, t_n)$, respectively, we have $s \succ_{lpo} t$ if and only if:

1. $f \succ g$ and $s \succ_{lpo} t_i$ for all $i$ with $1 \leq i \leq n$; or
2. $f = g$, and there is $j$ such that $s_i = t_i$ for all $i$ with $1 \leq i \leq j-1$, $s_j \succ_{lpo} t_j$, and $s \succ_{lpo} t_k$ for all $k$ with $j \leq k \leq n$; or
3. $s_j \succeq_{lpo} t$ for some $j$ with $1 \leq j \leq m$.

It is easy to show that if $\succ$ is well-founded, then $\succ_{lpo}$ is a simplification ordering, and if $\succ$ is total, then $\succ_{lpo}$ is total [102].

## 2.3 Rewrite systems

This section introduces basic notions of rewrite systems. We borrow the notation from [7] and represent an ordered pair $(s_1, s_2)$ in a binary relation $\circ$ as $s_1 \circ s_2$. All definitions and theorems are formulated with respect to HU.

A *rewrite system* is a binary relation $R$ on HU. Each ordered pair $s_1 \Rightarrow s_2$ in $R$ is called a *rewrite rule*. Given a rewrite system $R$, we define the *rewrite relation* for $R$, represented as $\rightarrow_R$, as the smallest binary relation on HU which contains all pairs of the form $t[s_1]_p \rightarrow_R t[s_2]_p$, where $s_1 \Rightarrow s_2 \in R$, $t \in \text{HU}$, and $p$ is a position of $t$. We let $\xrightarrow{*}_R$ be the reflexive–transitive closure of $\rightarrow_R$, and $\xleftrightarrow{*}_R$ be the symmetric closure of $\xrightarrow{*}_R$.

We say that a term $s \in \text{HU}$ is *irreducible by a rewrite system* $R$ if and only if there is no term $t \in \text{HU}$ such that $s \rightarrow_R t$. We extend the definition of irreducibility to literals and say that a literal $s_1 \bowtie s_2$ is irreducible by $R$ if and only both $s_1$ and $s_2$ are irreducible by $R$. Finally, we say that a term $t$ is a *normal form* of a term $s$ *with respect to* $R$ if and only if $t$ is irreducible by $R$ and $s \xleftrightarrow{*}_R t$. Given a rewrite system $R$, the *Herbrand equality interpretation induced by* $R$ is the Herbrand equality interpretation $R^*$ such that for every pair of terms $s, t \in \text{HU}$, we have

$$s \approx t \in R^* \text{ if and only if } s \xleftrightarrow{*}_R t.$$

Let $R$ be a rewrite system. We say that $R$ is *terminating* if and only if there is no infinite sequence of terms $s_1, s_2, \cdots$ such that $s_i \rightarrow_R s_{i+1}$ for every $i \in \mathbb{N}$. Next, we say that $R$ is *left-reduced* if and only if for each $s \Rightarrow t \in R$, the term $s$ is irreducible by $R \backslash \{s \Rightarrow t\}$. Finally, we say that $R$ is *Church-Rosser* if and only if every for every pair $s_1 \xleftrightarrow{*}_R s_2$, there exists a term $t$ such that $s_1 \xrightarrow{*}_R t$ and $s_2 \xrightarrow{*}_R t$. With these definitions, we have the following three results:[1]

**Lemma 1.** *Given a rewrite system $R$, if there exists a simplification order $\succ$ on* HU *such that for each $s \Rightarrow t \in R$ we have $s \succ t$, then $R$ is terminating and $s \rightarrow_R t$ implies $s \succ t$.*

**Lemma 2.** *If $R$ is terminating and left-reduced, then it is Church-Rosser.*

**Lemma 3.** *If $R$ is Church-Rosser, then each term $s \in$ HU has a unique normal form $s'$ with respect to $R$ such that $s \xrightarrow{*}_R s'$.*

If $\succ$ satisfies the condition in Lemma 1 with respect to a rewrite system $R$, we say that $\succ$ *embeds* the rules of $R$.

## 2.4 Description Logics

In this section we recapitulate the syntax and semantics of the DL $\mathcal{SROIQ}$ and the fragments of this logic that are relevant for this thesis. Furthermore, we describe the reasoning problems that our calculus aims to solve.

[1] We refer the reader to [7] for proofs of these results.

### 2.4.1 Syntax

A *DL signature* is a triple of the form $\langle \mathbf{B}, \mathbf{S}, \mathbf{I} \rangle$, where $\mathbf{B}$, $\mathbf{S}$, and $\mathbf{I}$ are three disjoint, countable sets of *concept names*, *role names*, and *individual names*, respectively. We assume that there exists a subset $\mathbf{S}' \subseteq \mathbf{S}$ of *simple* role names. We define the set $\mathbf{R}$ of *roles* as:

$$\mathbf{S} \cup \{S^- \mid S \in \mathbf{S}\}.$$

Similarly, we define the set of *simple roles* $\mathbf{R}'$ as:

$$\mathbf{S}' \cup \{S^- \mid S \in \mathbf{S}'\}.$$

Notice that $\mathbf{R}' \subseteq \mathbf{R}$. Furthermore, for every $S \in \mathbf{S}$, we define $\mathsf{inv}(S) = S^-$ and $\mathsf{inv}(S^-) = S$.

The set of *concepts* $\mathbf{C}$ is the minimal set containing:

- Each concept name $B \in \mathbf{B}$;
- the *top concept* symbol $\top$ and the *bottom concept* symbol $\bot$;
- the concept *negation* $\neg C$ for every concept $C \in \mathbf{C}$;
- the concept *intersection* $C_1 \sqcap C_2$ and concept *disjunction* $C_1 \sqcup C_2$ for each pair of concepts $C_1, C_2 \in \mathbf{C}$;
- the *existential restriction* $\exists R.C$ and the *universal restriction* $\forall R.C$ for every role $R \in \mathbf{R}$ and concept $C \in \mathbf{C}$;
- the *at-least restriction* $\geqslant n\, R.C$ and the *at-most restriction* $\leqslant n\, R.C$ for each $n \in \mathbb{N}_0$, role $R \in \mathbf{R}'$, and concept $C \in \mathbf{C}$;
- the *nominal* $\{o\}$ for each individual name $o \in \mathbf{I}$;
- the *local reflexivity* concept $\exists\, R.\mathsf{Self}$ for each role $R \in \mathbf{R}'$.

For each concept $C$, we say that both $C$ and every concept mentioned in the definition of $C$ are *subconcepts* of $C$. A *general concept inclusion* (GCI) is an expression of the form $C_1 \sqsubseteq C_2$, where $C_1, C_2 \in \mathbf{C}$. A *TBox* is a finite set of GCIs. An *individual assertion* is either:

- a *concept assertion* $C(o)$, with $C \in \mathbf{C}$ and $o \in \mathbf{I}$; or
- a *role assertion* $R(o_1, o_2)$, with $R \in \mathbf{R}$, $o_1, o_2 \in \mathbf{I}$; or

- a *negated role assertion* $\neg R(o_1, o_2)$, with $R \in \mathbf{R}$, $o_1, o_2 \in \mathbf{I}$; or
- an *equality assertion* $o_1 \approx o_2$, with $o_1, o_2 \in \mathbf{I}$; or
- an *inequality assertion* $o_1 \not\approx o_2$, with $o_1, o_2 \in \mathbf{I}$.

An *ABox* is a finite set of individual assertions. A *role inclusion axiom* (RIA) is an expression of the form

$$R_1 \circ \cdots \circ R_n \sqsubseteq R,$$

where $R_1, \cdots, R_n, R \in \mathbf{R}$, and we have $R \in \mathbf{R}'$ if and only if $n = 1$ and $R_1 \in \mathbf{R}'$. A *role hierarchy* is a finite set of RIAs. A *role assertion* is either:

- a *symmtery assertion* $\mathsf{Sym}(R)$, or a *reflexivity assertion* $\mathsf{Ref}(R)$, for $R \in \mathbf{R}$; or
- an *asymmtery assertion* $\mathsf{Asym}(R)$, or an *irreflexivity assertion* $\mathsf{Irref}(R)$, or a *transitivity assertion* $\mathsf{Trans}(R)$, for $R \in \mathbf{R}'$; or
- a *disjointness assertion* $\mathsf{Disj}(R_1, R_2)$, where $R_1, R_2 \in \mathbf{R}'$.

An *RBox* is the disjoint union of a role hierarchy and a finite set of role assertions.

A $\mathcal{SROIQ}$ *knowledge base* $\mathcal{K}$ is a triple $\langle \mathcal{T}, \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{T}$ is a *TBox*, $\mathcal{A}$ is an *ABox*, and $\mathcal{R}$ is an *RBox*. We next define several fragments of $\mathcal{SROIQ}$ that will be relevant for this thesis.

A knowledge base is $\mathcal{ALCHOIQ}^+$ if there are no occurrences of complex roles $R$ or transitivity assertions in $\mathcal{K}$; notice that this involves ensuring that $n = 1$ for all RIAs. A $\mathcal{SROIQ}$ knowledge base can be transformed into an $\mathcal{ALCHOIQ}^+$ knowledge base while preserving satisfiability and entailment, although this may generate an exponential increase in the size of the knowledge base [37, 55, 122], where size is measured in the number of symbols required to encode the knowledge base on the input tape of a Turing machine [58]. For many reasoning tasks, the encoding away of complex roles can be combined with any worst-case optimal decision procedure for $\mathcal{ALCHOIQ}^+$ to obtain a worst-case optimal procedure for $\mathcal{SROIQ}$. Furthermore, every $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$ can be *normalised* using a standardised procedure [97] in order to produce a new knowledge base $\mathcal{K}'$ where every element is of the form given in the second column of Table 2.4. The normalisation procedure is a variant of the *structural transformation* [104] used in first-order theorem proving, and it may incur in a polynomial increase in the size of the knowledge base. The normalised knowledge base $\mathcal{K}'$ is equisatisfiable and preserves entailment w.r.t. $\mathcal{K}$.

A $\mathcal{SROIQ}$ knowledge base is in $\mathcal{SRIQ}$ if it does not contain any nominal i.e. if no concept in a GCI or concept assertion has a subconcept of the form $\{o\}$ with $o \in \mathbf{I}$. An $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$ is:

- $\mathcal{ALCHIQ}^+$ if it does not contain any nominal.
- $\mathcal{ALCHOQ}$ if it does not contain any inverse role, that is, if $\mathbf{R} = \mathbf{S}$, and the RBox contains only RIAs.
- $\mathcal{ALCHOI}$ if it does not contain any at-most or at-least restriction, that is, if no concept in a GCI or concept assertion has a subconcept of the form $\geqslant n\, R.C$ or $\leqslant n\, R.C$ for $n \in \mathbb{N}_0$, $R \in \mathbf{R}$, $C \in \mathbf{C}$, and the RBox contains only RIAs. Furthermore, an $\mathcal{ALCHOI}$ knowledge based is $\mathcal{ALCHI}$ if it does not contain nominals, and $\mathcal{ALCH}$ if it contains neither nominals nor inverse roles.

An $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$ is Horn-$\mathcal{ALCHOIQ}^+$ if no concept in a GCI or concept assertion has a subconcept of the form $C_1 \sqcup C_2$, $\neg C$, or $\leqslant n\, R.C$ with $n \geq 2$, $C_1, C_2, C \in \mathbf{C}$, $R \in \mathbf{R}$, and no concept in the left-hand side of a GCI has a subconcept of the form $\geqslant n\, R.C$ or $\bot$. A Horn-$\mathcal{ALCHOIQ}^+$ knowledge base corresponds to the maximal fragment of an $\mathcal{ALCHOIQ}^+$ knowledge base which can be transformed into first-order Horn clauses. Finally, an $\mathcal{ALCHOIQ}^+$ knowledge base is $\mathcal{ELHO}$ if $\mathbf{C}$ does not contain $\bot$, or any concept of the form $\neg C$, $C_1 \sqcup C_2$, $\forall R.C$, $\geqslant n\, R.C$, $\leqslant n\, R.C$, or $\exists R.\mathsf{Self}$; where $C, C_1, C_2 \in \mathbf{C}$, $R \in \mathbf{R}$, and $n \in \mathbb{N}_0$, and the RBox contains only RIAs. An $\mathcal{ELHO}$ knowledge base is $\mathcal{ELH}$ if does not contain any nominal. An $\mathcal{ELH}$ knowledge base is $\mathcal{EL}$ if its RBox is empty.

### 2.4.2 Semantics

Given a DL signature $\Sigma$, an *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set, and the *interpretation function* $\cdot^{\mathcal{I}}$ is defined for each element in $\mathbf{B}$, $\mathbf{S}$, and $\mathbf{I}$, so that:

- For each concept name $B \in \mathbf{B}$, we have $B^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$.
- For each role name $S \in \mathbf{S}$, we have $S^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.
- For each individual name $o \in \mathbf{I}$, we have $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

The interpretation function is inductively extended to roles as follows:

$$(S^-)^{\mathcal{I}} = \{\langle y, x \rangle \mid \langle x, y \rangle \in S^{\mathcal{I}}\},$$

| | Syntax | Semantics |
|---|---|---|
| top concept | $\top$ | $\Delta^{\mathcal{I}}$ |
| bottom concept | $\bot$ | $\emptyset$ |
| negation | $\neg C$ | $\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$ |
| intersection | $C_1 \sqcap C_2$ | $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ |
| disjunction | $C_1 \sqcup C_2$ | $C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$ |
| existential restriction | $\exists R.C$ | $\left\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\right\}$ |
| universal restriction | $\forall R.C$ | $\left\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\right\}$ |
| at-least restriction | $\geqslant n\, R.C$ | $\left\{x \in \Delta^{\mathcal{I}} \mid \#\left\{y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\right\} \geq n\right\}$ |
| at-most restriction | $\leqslant n\, R.C$ | $\left\{x \in \Delta^{\mathcal{I}} \mid \#\left\{y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\right\} \leq n\right\}$ |
| local reflexivity | $\exists R.\mathsf{Self}$ | $\{x \in \Delta^{\mathcal{I}} \mid \langle x, x\rangle \in R^{\mathcal{I}}\}$ |

Table 2.1: Extension of the interpretation function $\cdot^{\mathcal{I}}$ to the set of concepts $\mathbf{C}$.

and to concepts as described in Table 2.1. Furthermore, the interpretation function defines a satisfaction condition for each element in the TBox, ABox, or RBox, as specified in Table 2.2. If we have $a \in C^{\mathcal{I}}$ for some interpretation $\mathcal{I}$ and concept $C$, we say that $a$ is an *instance of* $C$ in $\mathcal{I}$. Similarly, if we have $\langle a, b\rangle \in R^{\mathcal{I}}$ for some interpretation $\mathcal{I}$ and role $R$, we say that $b$ is an *R-filler* of $a$ in $\mathcal{I}$.

### 2.4.3 Inference Problems

The following key inference problems for DLs will be relevant in this thesis:

- *Consistency checking:* given a knowledge base $\mathcal{K}$, decide whether there exists an interpretation $\mathcal{I}$ that satisfies $\mathcal{K}$.
- *Subsumption:* given a knowledge base $\mathcal{K}$ and concepts $C$ and $D$, decide whether each instance of $C$ is also an instance of $D$ in all models of $\mathcal{K}$.
- *Classification:* given a knowledge base $\mathcal{K}$, decide subsumption for all pairs of concept names $B_1, B_2 \in \mathbf{B}$ mentioned in $\mathcal{K}$.
- *Instance retrieval:* given a knowledge base $\mathcal{K}$ and a concept $C$, find every individual name $o \in \mathbf{I}$ mentioned in $\mathcal{K}$ such that $o$ is interpreted as an instance of $C$ in all models of $\mathcal{K}$.
- *Realisation:* given a knowledge base $\mathcal{K}$, for each individual name $o \in \mathbf{I}$ mentioned in $\mathcal{K}$, compute the minimal subset of $\mathbf{B}$ containing every concept name $B$ mentioned in $\mathcal{K}$ such that $o$ is interpreted as an instance of $B$ in all models

| | Expression | Satisfaction Condition |
|---|---|---|
| *TBox*: | | |
| general concept inclusion | $C_1 \sqsubseteq C_2$ | $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ |
| *ABox*: | | |
| concept assertion | $C(o)$ | $o^{\mathcal{I}} \in C^{\mathcal{I}}$ |
| role assertion | $R(o_1, o_2)$ | $\langle o_1^{\mathcal{I}}, o_2^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ |
| neg. role assertion | $\neg R(o_1, o_2)$ | $\langle o_1^{\mathcal{I}}, o_2^{\mathcal{I}} \rangle \notin R^{\mathcal{I}}$ |
| equality assertion | $o_1 \approx o_2$ | $o_1^{\mathcal{I}} = o_2^{\mathcal{I}}$ |
| inequality assertion | $o_1 \not\approx o_2$ | $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$ |
| *RBox*: | | |
| symmetry | $\mathsf{Sym}(R)$ | $\langle x, y \rangle \in R^{\mathcal{I}} \rightarrow \langle y, x \rangle \in R^{\mathcal{I}}$ |
| reflexivity | $\mathsf{Ref}(R)$ | $x \in \Delta^{\mathcal{I}} \rightarrow \langle x, x \rangle \in R^{\mathcal{I}}$ |
| asymmetry | $\mathsf{Asym}(R)$ | $\langle x, y \rangle \in R^{\mathcal{I}} \rightarrow \langle y, x \rangle \notin R^{\mathcal{I}}$ |
| irreflexivity | $\mathsf{Irref}(R)$ | $x \in \Delta^{\mathcal{I}} \rightarrow \langle x, x \rangle \notin R^{\mathcal{I}}$ |
| transitivity | $\mathsf{Trans}(R)$ | $\langle x, y \rangle \in R^{\mathcal{I}} \wedge \langle y, z \rangle \in R^{\mathcal{I}} \rightarrow \langle x, z \rangle \in R^{\mathcal{I}}$ |
| disjointness | $\mathsf{Disj}(R_1, R_2)$ | $R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset$ |
| role inclusion axiom | $R_1 \circ \cdots \circ R_n \subseteq R$ | $\bigwedge_{1 \leq i \leq n} \langle x_{i-1}, x_i \rangle \in R_i^{\mathcal{I}} \rightarrow \langle x_0, x_n \rangle \in R^{\mathcal{I}}$ |

Table 2.2: Satisfaction condition for the components of the TBox, ABox, and RBox. Variables $x, y, z$, and $x_i$ for $i \in \mathbb{N}_0$ are universally quantified over $\Delta^{\mathcal{I}}$.

| Logic | Complexity | Sources |
|---|---|---|
| $\mathcal{SROIQ}$ | N2ExpTime-hard | Hard: [70] |
| $\mathcal{SRIQ}$ | 2ExpTime-complete | Hard: [70], Memb: [20] |
| $\mathcal{ALCHOIQ}^+$ | NExpTime-complete | Hard: [137], Memb: [138] |
| $\mathcal{ALCHIQ}^+$ | ExpTime-complete | Hard: [87], Memb: [138] |
| $\mathcal{ALCHOI}$ | ExpTime-complete | Hard: [87], Memb: [113] |
| $\mathcal{ALCHOQ}$ | ExpTime-complete | Hard: [87], Memb: [56] |
| Horn-$\mathcal{ALCHOIQ}^+$ | ExpTime-complete | Hard: [80], Memb: [105] |
| $\mathcal{ALCH}$ | ExpTime-complete | Hard: [115], Memb: [40] |
| $\mathcal{ELHO}$ | PTime-complete | Hard: [4], Memb: [72] |
| $\mathcal{EL}$ | PTime-complete | Hard: [4], Memb: [4] |

Table 2.3: Complexity for relevant inference problems and DLs.

of $\mathcal{K}$. This problem is sometimes defined with the additional requirement that the output for each $o$ cannot contain a pair of concept names $B_1, B_2$ such that $\mathcal{K} \models B_1 \sqsubseteq B_2$.

Let $\succ$ be a strict partial order on $\mathbf{R}$; we say that $\succ$ is *regular* if and only if $R_1 \succ R_2$ implies $R_1 \succ \mathsf{inv}(R_2)$ for every $R_1, R_2 \in \mathbf{R}$. With this, we characterise an RBox as *regular* if and only if there exists a strict partial order $\succ$ on $\mathbf{R}$ which is regular and every RIA in the RBox is of the form:

(i) $R \circ R \sqsubseteq R$; or

(ii) $\mathsf{inv}(R) \sqsubseteq R$; or

(iii) $R \circ R_1 \circ \cdots \circ R_n \sqsubseteq R$; or

(iv) $R_1 \circ \cdots \circ R_n \circ R \sqsubseteq R$;

where $R \succ R_i$ for every $i$ with $1 \leq i \leq n$ whenever $R_i \in \mathbf{R} \backslash \mathbf{R}'$. In order to guarantee decidability in $\mathcal{SROIQ}$ of the reasoning problems listed above, it is a crucial requirement that the RBox is regular [57].

If we assume regular RBoxes, then for each DL in the left column of Table 2.3, the inference problems listed above have the same computational complexity. Complexities are summarised in the second column of Table 2.3; numbers are assumed to be encoded in unary.

## 2.5 Clausification

In this section, we show how to transform $\mathcal{ALCHOIQ}^+$ knowledge bases into sets of equational clauses as defined in Section 2.1.

A *DL signature* consists of a two-sorted signature $\Sigma = \{\Sigma^S, \Sigma^F\}$, where $\Sigma^S$ contains a sort $\mathsf{a}$ corresponding to standard, single-sorted first-order terms, and a sort $\mathsf{p}$ defined as in Section 2.1, representing standard first-order atoms. The set of function symbols $\Sigma^F$ is the disjoint union of the following countable sets: (i) a set $\Sigma_f$ of *successor function* symbols of type $\mathsf{a} \to \mathsf{a}$, (ii) a set $\Sigma_A$ of *unary predicate* symbols of type $\mathsf{a} \to \mathsf{p}$, (iii) a set $\Sigma_S$ of *binary predicate* symbols of type $\mathsf{a} \times \mathsf{a} \to \mathsf{p}$, and (iv) a set $\Sigma_c$ of *constant* symbols of type $\to \mathsf{a}$. In the remainder of this section, we assume that $\Sigma = \{\{\mathsf{a}, \mathsf{p}\}, \{\Sigma_f \sqcup \Sigma_A \sqcup \Sigma_S \sqcup \Sigma_c\}\}$ is a fixed DL signature, and that notions defined in Section 2.1 refer implicitly to $\Sigma$.

For each $f \in \Sigma_f$, the *$f$-successor* of a term $t$ is the term $f(t)$; we then say that $t$ is the *predecessor* of $f(t)$. We consider a *central variable* $x \in \mathcal{X}_\mathsf{a}$, and a set of *neighbour variables* $\{z_i \mid i \in \mathbb{N}\} \subset \mathcal{X}_\mathsf{a}$.

A *DL-$\mathsf{a}$-term* is a term of the form $x$, $z_i$ for some $i \in \mathbb{N}$, $f(x)$ for some $f \in \Sigma_f$, or $o \in \Sigma_c$. A *DL-$\mathsf{p}$-term* is a term of the form $B(z_i)$, $B(x)$, $B(f(x))$, $B(o)$, $S(z_i, x)$, $S(x, z_i)$, $S(x, x)$, $S(x, f(x))$ or $S(f(x), x)$; where $f \in \Sigma_f$, $B \in \Sigma_A$, $S \in \Sigma_S$, and $o \in \Sigma_c$. A *DL-atom* is an equality of the form $A \approx \mathsf{true}$, where $A$ is a DL-$\mathsf{p}$-term. A *DL-literal* is a DL-atom or an equality or inequality between DL-$\mathsf{a}$-terms. A *DL-clause* is a clause which contains only DL-atoms of the form $B(x)$, $S(x, z_i)$, $S(z_i, x)$, or $S(x, x)$ in the body, and only DL-literals in the head.

Following the well-known correspondence between Description Logics and first-order logic, we have that a normalised $\mathcal{ALCHOIQ}^+$ knowledge base corresponds to DL-clauses as specified in Table 2.4. We assume $\mathbf{B} \subseteq \Sigma_A$, $\mathbf{S} \subseteq \Sigma_S$, and $\mathbf{I} \subseteq \Sigma_c$. Note that GCIs of the form DL4 would normally be translated as:

$$B_1(x) \wedge \bigwedge_{1 \le i \le n+1} (S(x, z_i) \wedge B_2(z_i)) \to \bigvee_{1 \le i < j \le n+1} z_i \approx z_j. \tag{1}$$

This is, however, not a DL clause, due to the presence of atoms $B_2(z_i)$ for $1 \le i \le n+1$ in the body. Our translation introduces fresh predicates $S_{B_2}$ for each pair of $S$ and $B_2$, then it adds the clause $S(z_1, x) \wedge B_2(x) \to S_{B_2}(z_1, x)$, and finally it replaces conjunction $S(x, z_i) \wedge B_2(z_i)$ in clause (1) with literal $S_{B_2}(x, z_i)$, for each $1 \le i \le n+1$.

We define an *ontology* $\mathcal{O}$ as a set of DL-clauses. An ontology is $\mathcal{ALCHOIQ}^+$ if each DL-clause in $\mathcal{O}$ is of one of the forms given in the rightmost column of Table 2.4. An ontology is $\mathcal{ALCHIQ}^+$ if it does not contain clauses of the form DL10 or DL11.

| | | | | | |
|---|---|---|---|---|---|
| DL1 | $\bigsqcap_{1\leq i\leq n} B_i \sqsubseteq \bigsqcup_{n+1\leq i\leq m} B_i$ | $\rightsquigarrow$ | $\bigwedge_{1\leq i\leq n} B_i(x)$ | $\rightarrow$ | $\bigvee_{n+1\leq i\leq m} B_i(x)$ |
| DL2 | $B_1 \sqsubseteq \geqslant n\, S.B_2$ | $\rightsquigarrow$ | $\{B_1(x)$ | $\rightarrow$ | $B_2(f_i(x)) \mid 1 \leq i \leq n\}$, |
| | | | $\{B_1(x)$ | $\rightarrow$ | $S(x, f_i(x)) \mid 1 \leq i \leq n\}$, |
| | | | $\{B_1(x)$ | $\rightarrow$ | $f_i(x) \not\approx f_j(x) \mid 1 \leq i < j \leq n\}$ |
| DL3 | $\exists S.B_1 \sqsubseteq B_2$ | $\rightsquigarrow$ | $S(z_1, x) \wedge B_1(x)$ | $\rightarrow$ | $B_2(z_1)$ |
| DL4 | $B_1 \sqsubseteq \leqslant n\, S.B_2$ | $\rightsquigarrow$ | $S(z_1, x) \wedge B_2(x)$ | $\rightarrow$ | $S_{B_2}(z_1, x)$ |
| | | | $B_1(x) \wedge \bigwedge_{1\leq i\leq n+1} S_{B_2}(x, z_i)$ | $\rightarrow$ | $\bigvee_{1\leq i<j\leq n+1} z_i \approx z_j$ |
| DL5 | $B \sqsubseteq \exists S.\mathsf{Self}$ | $\rightsquigarrow$ | $B(x)$ | $\rightarrow$ | $S(x, x)$ |
| DL6 | $\exists S.\mathsf{Self} \sqsubseteq B$ | $\rightsquigarrow$ | $S(x, x)$ | $\rightarrow$ | $B(x)$ |
| DL7 | $S_1 \sqsubseteq S_2$ | $\rightsquigarrow$ | $S_1(z_1, x)$ | $\rightarrow$ | $S_2(z_1, x)$ |
| DL8 | $S_1 \sqsubseteq S_2^-$ | $\rightsquigarrow$ | $S_1(z_1, x)$ | $\rightarrow$ | $S_2(x, z_1)$ |
| DL9 | $\mathsf{Disjoint}(S_1, S_2)$ | $\rightsquigarrow$ | $S_1(z_1, x) \wedge S_2(z_1, x)$ | $\rightarrow$ | $\bot$ |
| DL10 | $\{o\} \sqsubseteq B$ | $\rightsquigarrow$ | $\top$ | $\rightarrow$ | $B(o)$ |
| DL11 | $B \sqsubseteq \{o\}$ | $\rightsquigarrow$ | $B(x)$ | $\rightarrow$ | $x \approx o$ |

Table 2.4: Left: normalised $\mathcal{ALCHOIQ}^+$ GCIs, assertions, and elements of the RBox; to the right, their transformation to DL-clauses. Each $B_{(i)}$ is a concept name in $\Sigma_A$; each $S_{(i)}$ is a role name in $\Sigma_S$; each $f_i$ is a fresh successor function symbol in $\Sigma_f$; symbol $o$ is a constant in $\Sigma_c$. Roles $S_{B_2}$ in DL4 are fresh, contained in $\Sigma_S$, and determined by $S$ and $B_2$.

An ontology is $\mathcal{ALCHOQ}$ if it does not contain clauses of the form DL5, DL6, DL8, or DL9. An ontology is $\mathcal{ALCHOI}$ if it does not contain clauses of the forms DL4 to DL6, or DL9, and if for every clause of the form DL2 we have $n = 1$. An ontology is *Horn* if it does not contain clauses with more than one literal in the head. Finally, an ontology is $\mathcal{ELHO}$ if it is Horn and contains only clauses of the form DL1, DL2 with $n = 1$, DL3, DL7, DL10, and DL11; such an ontology is $\mathcal{ELH}$ if it does not contain clauses of the form DL10 or DL11.

Due to the correspondence between DLs and first-order logic, any $\mathcal{ALCHOQ}$, $\mathcal{ALCHOI}$, Horn-$\mathcal{ALCHOIQ}^+$ or $\mathcal{ELHO}$ knowledge base (defined using DL notation in Section 2.4.1) can be transformed into an $\mathcal{ALCHOQ}$, $\mathcal{ALCHOI}$, Horn-$\mathcal{ALCHOIQ}^+$, or $\mathcal{ELHO}$ ontology (consisting of DL-clauses), respectively, while preserving satisfiability and entailment. Our definitions of $\mathcal{ALCHIQ}^+$ and $\mathcal{ELH}$ ontologies, however, require that both $\mathcal{ALCHIQ}^+$ and $\mathcal{ELH}$ knowledge bases must have an empty *ABox* in order to be converted, respectively, into $\mathcal{ALCHIQ}^+$ and $\mathcal{ELH}$ ontologies in a way that preserves satisfiability and entailment.

A *query clause* is a DL-clause $\Gamma_Q \rightarrow \Delta_Q$ where each atom is of the form $B(x)$,

with $B \in \Sigma_A$. Query clauses in the CB calculus for $\mathcal{SRIQ}$ [20] also allow atoms of the form $S(x, x)$ in $\Delta_Q$ with $S \in \Sigma_S$; however, each atom of this form can be replaced in the query clause with a fresh atom $B_S(x)$ if $\mathcal{O}$ is extended with a clause $S(x, x) \rightarrow B_S(x)$ of the form DL6; this transformation preserves query entailment.

# Chapter 3

# Consequence-Based Reasoning

In this chapter, we introduce the basics of CB reasoning. In Section 3.1, we review and compare the CB reasoning algorithms that have been developed for DLs since the early 2000s. In Section 3.2, we summarise the CB calculus for $\mathcal{ALCHIQ}^+$ by Bate et al., which we will extend to handle nominals in Part II of this thesis. Finally, in Section 3.3, we review combined approaches to reasoning that use CB algorithms in conjunction with other methods.

## 3.1 Evolution of CB Reasoning

In this section we provide a historical perspective of CB reasoning, starting from the appearance of the first such algorithm in 2004. This section is accompanied by Appendix A, which provides a more detailed description and comparison between CB calculi in the literature.

### 3.1.1 First CB Algorithms: the $\mathcal{EL}$ Family

Early work in CB reasoning focused on the $\mathcal{EL}$ family of DLs. These languages received special interest in the early 2000s, after they were shown to provide a good balance between expressive power and complexity of reasoning tasks. In particular, the problem of concept subsumption was shown to be in PTime for increasingly expressive languages in the $\mathcal{EL}$ family of DLs [8, 27, 4]. Furthermore, and in contrast to earlier assumptions, the logics in the $\mathcal{EL}$ family proved to be expressive enough to capture many relevant real-world ontologies. The logic $\mathcal{EL}^{+,+}$ which includes several additional expressive features with respect to $\mathcal{ELH}$, such as the concept $\bot$ and RIAs, eventually became the basis for the OWL 2 EL profile [107] of the ontology language

OWL 2, and was used to write the SNOMED ontology, which is widely used in the medical sector [119].

The first CB calculi was presented in 2004 by Brandt et al. [27] in order to prove decidability of subsumption for the DL $\mathcal{ELH}$ in polynomial time. The algorithm was extended shortly afterwards by Baader et al. [4] to the DL $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$, which extends $\mathcal{EL}^{++}$ with concrete domains. The introduction of CB calculi to prove tractability of subsumption problems marked a sharp contrast with previous techniques, which were based on structural subsumption [11, 8]. Furthermore, the algorithms were shown to be capable of fully classifying an ontology in polynomial time, in contrast to other existing approaches at the time. We next summarise a simplified version of these calculi in order to introduce the basics of consequence-based reasoning and lay the groundwork for further discussion.

To classify an ontology, the calculus first creates a data structure consisting of a pair of sets, denominated $\mathcal{S}$ and $\mathcal{E}$. The set $\mathcal{S}$ contains a set $\mathcal{S}(A)$ for each concept name $A$ mentioned in $\mathcal{O}$, and the set $\mathcal{E}$ contains a set $\mathcal{E}(R)$ for each role name $R$ mentioned in $\mathcal{O}$. For each concept name $A$, the set $\mathcal{S}(A)$ represents the concept names that are known to subsume $A$ in each model of the ontology. Thus, if $B \in \mathcal{S}(A)$ for some concept name $B$, then we have that $\mathcal{O} \models A \sqsubseteq B$. Conversely, every time the calculus proves $\mathcal{O} \models A \sqsubseteq C$ for some concept name $C$, then $C$ is added to $\mathcal{S}(A)$. Finally, for each role name $R$, the set $\mathcal{E}(R)$ represents every pair of concept names $(A, B)$ such that $\mathcal{O} \models A \sqsubseteq \exists R.B$. The sets are initialised as follows:

- $\mathcal{S}(A) = \{A, \top\}$ for each $A \in \mathbf{B}$
- $\mathcal{E}(R) = \emptyset$ for each $R \in \mathbf{S}$.

The calculus requires that the input ontology be normalised into a particular form. For instance, in the case of $\mathcal{EL}$ ontologies, the TBox may only contain GCIs of the form $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists R.B$, and $\exists R.A \sqsubseteq B$, for $A, A_1, A_2, B$ concept names, and $R$ a role name. The normalisation procedure is similar to previously existing methods for other logics [84]; it can be carried out in polynomial time, and it may result in a polynomial increase in the size of the ontology.

Inference rules in Table 3.1 are used to derive new subsumptions and expand sets $\mathcal{S}$ and $\mathcal{E}$. For instance, inference rule $\mathsf{CR1}$ takes a set $\mathcal{S}(A)$, with $B \in \mathcal{S}(A)$ for some concept name $B$, and looks for a GCI of the form $B \sqsubseteq C$, where $C$ is a concept name, in the TBox of $\mathcal{O}$. If it finds it, then it adds $C$ to $\mathcal{S}(A)$. The complete set of inference rules for the original (non-simplified) $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ calculus [4] is presented in Appendix A, Figure A.1. When sets $\mathcal{S}$ and $\mathcal{E}$ become saturated by the inference rules

| | |
|---|---|
| CR1 | If $B \in \mathcal{S}(A)$ and $B \sqsubseteq C \in \mathcal{O}$, then add $C$ to $\mathcal{S}(A)$. |
| CR2 | If $B, C \in \mathcal{S}(A)$, $B \sqcap C \sqsubseteq D \in \mathcal{O}$, and $D \notin \mathcal{S}(A)$ then add $D$ to $\mathcal{S}(A)$. |
| CR3 | If $B \in \mathcal{S}(A)$ and $B \sqsubseteq \exists R.C \in \mathcal{O}$, then add $(A, C)$ to $\mathcal{E}(R)$. |
| CR4 | If $(A, B) \in \mathcal{E}(R)$, $B \in \mathcal{S}(C)$, and $\exists R.C \sqsubseteq D \in \mathcal{O}$, then add $D$ to $\mathcal{S}(A)$. |

Table 3.1: Rules of the simplified $\mathcal{EL}$ procedure, based on [27] and [4].

from Table 3.1, and if we assume that $\mathcal{O}$ is $\mathcal{EL}$, then all subsumptions of the form $A \sqsubseteq B$ and $A \sqsubseteq \exists R.B$ that are entailed by $\mathcal{O}$ can be found in these sets. Hence, the calculus achieves *one-pass* classification.

Although none of these calculi were originally referred to as "consequence-based," they already display key defining features of the CB approach. For instance, they operate by saturating sets of logical consequences of the ontology, where each set contains inferences relevant to all instances of a particular concept name; subsequent CB calculi will refer to these sets as "contexts."

The $\mathcal{EL}^{++}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ calculus was implemented in the reasoner CEL [12], and experiments with this system on life-science ontologies showed that efficient reasoning was feasible even for very large ontologies. Their elegance, simplicity, and practical success would go on to inspire similar methods for more expressive DLs. Furthermore, interest was sparked into developing more efficient implementations of CB calculi for the $\mathcal{EL}$ family of DLs. Research in this area has covered diverse topics such as the development of faster CB calculi for $\mathcal{EL}$ and its variants, the creation of efficient strategies for saturation using tailored data structures and indices, or incremental reasoning. Reasoners such as ELK [74] and Snorocket [90] draw upon this work in order to provide highly efficient, robust, and scalable implementations for lightweight extensions of $\mathcal{EL}$. The improved performance of these systems illustrates the importance of algorithmic optimisations in shifting from the high-level description of a calculus to practical applications.

### 3.1.2 Extending CB Reasoning to Horn-$\mathcal{SHIQ}$

In 2009, Kazakov [69] extended the $\mathcal{ELH}$ fragment of the calculi in the previous section to the DL Horn-$\mathcal{SHIQ}$, which allows for transitive roles, universal restrictions ($\forall R.A$), inverse roles ($\mathcal{I}$), and number restrictions of the form $\geqslant n\, R.C$ and $\leqslant 1\, R.C$ in positive occurrences. The calculus also introduced a new, simplified notation which replaced sets $\mathcal{S}(A)$ and $\mathcal{E}(R)$ by an explicit list of subsumptions of the form $A \sqsubseteq B$

$$\textsf{CR1'}\ \frac{A \sqsubseteq B \qquad B \sqsubseteq C \in \mathcal{O}}{A \sqsubseteq C}$$

$$\textsf{CR2'}\ \frac{A \sqsubseteq B \qquad A \sqsubseteq C \qquad B \sqcap C \sqsubseteq D \in \mathcal{O}}{A \sqsubseteq D}$$

$$\textsf{CR3'}\ \frac{A \sqsubseteq B \qquad B \sqsubseteq \exists R.C \in \mathcal{O}}{A \sqsubseteq \exists R.C}$$

$$\textsf{CR4'}\ \frac{A \sqsubseteq \exists R.B \qquad B \sqsubseteq C \qquad \exists R.C \sqsubseteq D \in \mathcal{O}}{A \sqsubseteq D}$$

Table 3.2: Inference rules for $\mathcal{EL}$ – notation by Kazakov (2009). In each rule, premises appear over the horizontal line, and the derived conclusion appears under it.

and $B_1 \sqsubseteq \exists R.B_2$, respectively, where $B$, $B_1$, and $B_2$ are concept names. Using this notation, the inference rules in Table 3.1 would look as shown in Table 3.2.

The Horn-$\mathcal{SHIQ}$ calculus encodes away transitive roles using well-known methods [55] which do not affect the computational complexity of the overall procedure. However, in order to deal with the remaining expressive features of Horn-$\mathcal{SHIQ}$, the calculus needs to derive consequences of a more general form. In the calculi from Section 3.1.1, derived consequences for $\mathcal{EL}$ ontologies are either of the form $A \sqsubseteq B$ or $A \sqsubseteq R.B$, where $A$ and $B$ are atomic names, and $R$ is a role name. By contrast, in Horn-$\mathcal{SHIQ}$, a derived logical consequence of the from $A \sqsubseteq \exists R.B$ can interact with a GCI of the form $A \sqsubseteq \forall R.C$ to produce a new consequence of the form $A \sqsubseteq R.(B \sqcap C)$. Thus, the Horn-$\mathcal{SHIQ}$ calculus allows the derivation of consequences of the form:

$$\prod_{i=1}^{n} A_i \sqsubseteq B \qquad \text{and} \qquad \prod_{i=1}^{n} A_i \sqsubseteq \exists R. \left( \prod_{j=1}^{m} B_j \right),$$

where $A_i$, $B_j$ are atomic concepts for $1 \leq i \leq n$ and $1 \leq j \leq m$, $R$ is an atomic role, and $n$ and $m$ may be as large as the total number of concept names in the ontology. This syntax is also useful to represent consequences of GCIs involving number restrictions, as we can see in the following inference rule of the calculus:

$$\textsf{R5}\ \frac{\begin{array}{llll} M \sqsubseteq \exists R_1.N_1 & N_1 \sqsubseteq B & R_1 \sqsubseteq S \in \mathcal{O} & \\ M \sqsubseteq \exists R_2.N_2 & N_2 \sqsubseteq B & R_2 \sqsubseteq S \in \mathcal{O} & N_1 \sqsubseteq \leqslant 1\, S.B \in \mathcal{O} \end{array}}{M \sqsubseteq \exists R_1(N_1 \sqcap N_2)}$$

where $M, N_1, N_2$ are conjunctions of concept names, $R_1, R_2, S$ are roles, and $B$ is a concept name.

| | | | |
|---|---|---|---|
| CR1″ | $\mathsf{imp}(\{A\},\{B\}),\ \mathsf{imp}(\{B\},\{C\})$ | $\rightarrow$ | $\mathsf{imp}(\{A\},\{C\})$ |
| CR2″ | $\mathsf{imp}(\{A\},\{B\}),\ \mathsf{imp}(\{A\},\{C\})$<br>$\mathsf{imp}(\{B,C\},\{D\})$ | $\rightarrow$ | $\mathsf{imp}(\{A\},\{D\})$ |
| CR3″ | $\mathsf{imp}(\{A\},\{B\}),\ \mathsf{enf}(\{B\},\{R\},\{C\})$ | $\rightarrow$ | $\mathsf{enf}(\{A\},\{R\},\{C\})$ |
| CR4″ | $\mathsf{enf}(\{A\},\{R\},\{C\}),\ \ R' \in \{R\}$<br>$C' \in \{C\},\ \ \exists R'.C' \sqsubseteq D \in \mathcal{O}$ | $\rightarrow$ | $\mathsf{imp}(\{A\},\{D\})$ |

Table 3.3: Simplified version for $\mathcal{EL}$ of the inference rules by Ortiz et al. (2010).

The calculus shares many desirable properties with the algorithms from the previous section: it is worst-case optimal (it works in exponential time for a logic that is ExpTime-complete) and it allows for one-pass classification. Furthermore, it displays a pay-as-you-go behaviour, as it mimics the calculi from the previous section on $\mathcal{ELH}$ ontologies, and therefore it is worst-case time polynomial in such cases. This behaviour is very convenient in practical applications, because the calculus can deal very effectively with ontologies that are "mostly" $\mathcal{ELH}$, that is, ontologies that have only a small number of GCIs that are not $\mathcal{ELH}$. This was empirically demonstrated after the calculus was implemented in a reasoner called CB [69], which was the first system to classify the full version of the well-known GALEN ontology [130].

### 3.1.3 Reasoning with Nominals: Horn-$\mathcal{SROIQ}$ and $\mathcal{ELHO}$

Although the original calculus in [4] for $\mathcal{EL}^{++}$ included nominals, it was later found to be incomplete for handling them [73]. Complete CB calculi for logics involving nominals were later developed for Horn-$\mathcal{SROIQ}$ [105] and $\mathcal{ELHO}$ [73].

The $\mathcal{ELHO}$ calculus uses a similar notation to the Horn-$\mathcal{SHIQ}$ calculus from the previous section. By contrast, the Horn-$\mathcal{SROIQ}$ calculus uses a Datalog-inspired formalism [41] to represent the ontology, the inference rules, and the consequences. In particular, concept names, conjunctions of concept names, and sets of roles are treated as constants, and their properties and relations are represented as predicates. For instance, subsumption $\mathcal{O} \models A \sqsubseteq B$ for concept names $A$ and $B$ is represented in the calculus as the fact $\mathsf{imp}(\{A\},\{B\})$, where $\mathsf{imp}$ is a special predicate representing subsumption. Similarly, $\mathcal{O} \models A \sqsubseteq \exists R.B$, for $R$ a role name, is represented by the fact $\mathsf{enf}(\{A\},\{R\},\{B\})$, where $\mathsf{enf}$ is a ternary predicate such that $\mathsf{enf}(X,Y,Z)$ represents that in every model of $\mathcal{O}$, every instance of $X$ is related to an instance of $Z$ via every role in $Y$, where $X$ and $Z$ are conjunctions of concept names, and $Y$ is a set of roles.

The inference rules for $\mathcal{EL}$ from Table 3.1 could be rewritten, using the notation of the Horn-$\mathcal{SROIQ}$ calculus by Ortiz and colleagues, in the way presented in Table 3.3.

In order to ensure completeness in the presence of nominals, both the $\mathcal{ELHO}$ and the Horn-$\mathcal{SROIQ}$ calculi keep track of "conditional" consequences, that is, consequences that hold only in models where some atomic concepts are non-empty. To see why this may be necessary, observe that a GCI of the form $C \sqsubseteq \{o\}$ in an ontology splits the models of the ontology into two kinds: those in which $C$ is interpreted as the empty set and those where $C$ is interpreted as a singleton; this also affects concepts subsumed by $C$.

Both the Horn-$\mathcal{SROIQ}$ and the $\mathcal{ELHO}$ calculi introduce additional syntax to keep track of such consequences. The Horn-$\mathcal{SROIQ}$ calculus uses a unary predicate $\mathsf{rel}(X)$ which represents the claim that the conjunction $X$ of concept names is non-empty in a model. On the other hand, the $\mathcal{ELHO}$ calculus introduces a new type of consequences which can be written as $G : C \sqsubseteq D$; this represents the fact that inclusion $C \sqsubseteq D$ holds in any model of the ontology where concept $G$ is not empty. Furthermore, the calculus also uses the dependency $G \rightsquigarrow H$ to represent that concept $H$ is non-empty in a model where concept $G$ is also non-empty. Notice that such consequences are not always "local," in the sense that they involve model elements that may be arbitrarily far apart. In general, the presence of nominals in ontologies is difficult to marry with a focus on deriving local clauses, which we consider to be a key feature of CB reasoning, as we explain in Section 4.1. This difficulty is paramount in developing our CB calculus for $\mathcal{ALCHOIQ}^+$, as we show in Section 4.2.

Another interesting aspect of the Horn-$\mathcal{SROIQ}$ method is the introduction of a binary predicate $\mathsf{equal}(X, Y)$ to represent equality between individuals $X$ and $Y$, together with inference rules that axiomatise this predicate as a congruence relation. Later calculi for DLs with number restrictions and/or nominals also require the use of equalities, as we show in Section 3.1.4; these calculi, however, will use paramodulation-based inference rules [144] instead of axiomatising equality.

Finally, the Horn-$\mathcal{SROIQ}$ calculus also introduces a new inference rule to handle the "terrible trifecta" of nominals, inverse roles, and functional restrictions interacting together simultaneously. The rule ensures that if we have two concepts (or conjunctions of concepts) $C$ and $D$, and a role $R$ such that:

- in every model, all instances of $C$ and $D$ have the same named individual as an $R$-filler, that is, there exists some individual $o$ such that for every model $\mathcal{I}$ and domain element $c$, having $c \in C^{\mathcal{I}}$ or $c \in D^{\mathcal{I}}$ implies $(c, o^{\mathcal{I}}) \in R^{\mathcal{I}}$, and

- $\mathsf{inv}(R)$ is functionally restricted, that is, for every model $\mathcal{I}$ and domain element $c$, there exists at most one domain element $d$ such that $(d, c) \in R^{\mathcal{I}}$,

then the interpretation of $C$ and $D$ is equal in every model to a single, unique domain element. This is enforced by writing $\mathsf{same}(C, D)$, where the special predicate $\mathsf{same}$ has the following meaning: $\mathsf{same}(X, Y)$ represents that conjunctions $X$ and $Y$ of concept names have a unique instance in every model, and this instance is the same for both. Similar strategies have been used in other reasoning approaches for $\mathcal{SHOIQ}$, such as the resolution-based calculus in [76] or the tableau calculus in [59]; we discuss these in detail in Section 7.3, where we compare them to our own approach.

To conclude this section, we highlight the fact that both the Horn-$\mathcal{SROIQ}$ and the $\mathcal{ELHO}$ calculi are worst-case optimal, although they do not achieve one-pass classification; the reason for this is further analysed in Section 7.3. In addition, the $\mathcal{ELHO}$ algorithm displays pay-as-you-go behaviour, in the sense that it reduces to the standard $\mathcal{ELH}$ calculus in the absence of nominals.

### 3.1.4 Moving Beyond Horn DLs

The introduction of disjunction in the CB paradigm leads to new difficulties. In previous calculi for Horn DLs without nominals, a canonical model can be built from the saturated set of consequences in order to disprove any subsumption $A \sqsubseteq B$ that has not been derived by the calculus. In non-Horn DLs without nominals, however, such a model may not exist.

The CB calculus for $\mathcal{ALCH}$ by Simančík et al., presented in [124], was the first to support concept disjunction. The calculus uses an explicit representation of disjunction which is similar to that used in resolution calculi for fragments of first-order logic. The calculus also introduces ordering, which dramatically reduces the number of inferences to consider [111]. The approach to disjunction pioneered by this calculus has been adopted by more recent calculi [125, 18], including the one presented in this thesis.

Much like calculi for Horn DLs, the $\mathcal{ALCH}$ CB calculus transforms the input ontology into a normal form, which allows for conjunctions of concepts of the form $A$, $\exists R.A$ and $\forall R.A$, with $A$ a concept name or its negation, and $R$ a role name. Due to the presence of disjunction in the ontology, the consequences derived by the calculus have a more general form:

$$\prod_{i=1}^{n} L_i \sqsubseteq \bigsqcup_{i=1}^{m} A_i \sqcup \exists R. \left( \prod_{i=n+1}^{k} L_i \right),$$

where each expression $L_i$ for $1 \leq i \leq k$ is an atomic concept or its negation, each $A_i$ is an atomic concept for $1 \leq i \leq m$, and $R$ is a role name. Once again, the calculus is worst-case optimal and pay-as-you-go, as it simulates the Horn-$\mathcal{ALCH}$ and the $\mathcal{ELH}$ approaches for ontologies of the corresponding expressivity. One-pass classification is retained.

In 2014, Simančík and colleagues [125] extended this calculus to $\mathcal{ALCHI}$ while ensuring worst-case optimality, pay-as-you-go behaviour, and one-pass classification. They also introduced a unifying framework for CB reasoning, which formalised explicitly many of the features that were tacitly present in previous calculi, and offered the possibility of mimicking these calculi by choosing appropriate parameters. The framework was later used by Bate and colleagues to extend CB reasoning to the DL $\mathcal{ALCHIQ}^+$ [20]. We illustrate this framework in Section 3.2, where we give an overview of that CB calculus for $\mathcal{ALCHIQ}^+$.

The $\mathcal{ALCHIQ}^+$ calculus also introduced a novel feature with respect to the framework by Simančík and colleagues: it abandoned DL syntax for first-order logic syntax with equality, which it used to represent both ontology axioms and derived consequences. This choice was motivated by the need for representing more general consequences, stemming from the interactions between number restrictions and inverse roles. Equalities participate in inferences that implement well-known paramodulation rules for equality reasoning [144]. The $\mathcal{ALCHIQ}^+$ calculus is also worst-case optimal and it enjoys a pay-as-you-go property, in the sense that it mimics the earlier $\mathcal{ALCHI}$ and $\mathcal{ELH}$ calculi when the input ontology is in these DLs; furthermore, it retains the one-pass classification property. The calculus has been implemented in the reasoner Sequoia [20], which has proved to be competitive with state-of-the-art reasoners despite being an early prototype, and has shown evidence of practical pay-as-you-go behaviour.

Experiments have shown that CB calculi based on first-order logic notation occasionally face the challenge of having to generate very long disjunctive clauses, which slow down execution [20, 136]. Furthermore, the problem is exacerbated when the input ontology uses number restrictions with large numbers. This issue has motivated the introduction of some of the "hybrid" approaches discussed in Section 3.3, which combine CB reasoning with other techniques in order to ameliorate the problem. In Chapter 8, we show how these problems are reflected in our calculus, and then we introduce several algorithmic optimisations that address them.

## 3.2 The CB Calculus for $\mathcal{ALCHIQ}^+$

The calculus for $\mathcal{ALCHIQ}^+$ proposed in [20] aims at deciding whether $\mathcal{O} \models Q$, for $\mathcal{O}$ an ontology and $Q$ a query clause, that is, a clause of the form:

$$\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{i=1}^{m} B_i(x), \tag{1}$$

for $A_i$ and $B_i$ concept names. The calculus is based on the framework for CB reasoning introduced in [125], where each derived consequence is assigned to a particular *context*. Each context $v$ is represented as a node in a graph called a *context structure*, and it is associated with a *core*, denoted $\mathsf{core}_v$, which consists of a conjunction of atoms with free variables $x$ and (optionally) $y$, where $y$ always appears in atoms of the form $R(x, y)$ or $R(y, x)$.

For each model $\mathcal{I}$ of $\mathcal{O}$, a context $v$ can be seen as representing every element $c$ in the domain of $\mathcal{I}$ such that $\mathcal{I} \models \exists y.\mathsf{core}_v\{x \mapsto c\}$. For example, if we consider context $v$ with $\mathsf{core}_v = \{A(x), R(x, y)\}$ and an arbitrary model $\mathcal{I}$ of $\mathcal{O}$, then $v$ intuitively represents all instances of $A$ in $\mathcal{I}$ that are related via $R$ to some arbitrary domain element.

Consequences in the calculus are represented as constant-free *context clauses*, which contain only variables $x$ and $y$, and functional terms of the form $f(x)$. The framework requires that each context clause $\Gamma \rightarrow \Delta$ must be assigned to a context $v$ such that $\mathsf{core}_v \subseteq \Gamma$; the clause is then written as $\Gamma' \rightarrow \Delta$, where $\Gamma = \mathsf{core}_v \wedge \Gamma'$. Variables have a special meaning in context clauses: variable $x$ is universally quantified over domain elements satisfying $\mathsf{core}_v$, while variable $y$ refers to a predecessor of $x$.

Context structures also contain directed edges between contexts. Each edge is labelled with a successor function symbol. An $f$-labelled edge from a context $u$ to a context $v$ with $\mathsf{core}_v \neq \top$ indicates that, in each model $\mathcal{I}$ of $\mathcal{O}$, every element represented by context $u$ has an $f$-successor which is represented by context $v$. If there is an edge from context $u$ to context $v$, then $u$ is a *predecessor context* of $v$, and $v$ is a *successor context* of $u$. A context structure may have more than one edge from $u$ to $v$, each labelled with a different function symbol. A context structure may also contain self-loops.

Similarly to ordered resolution calculi [15], each context $v$ is parametrised by an ordering $\succ_v$ on literals that determines which literals and context clauses can be selected by inference rules. Using this ordering prevents the derivation of many clauses that are unnecessary for deciding whether $Q$ is entailed by the ontology. If the calculus is used to decide multiple queries in a single run e.g., during classification,

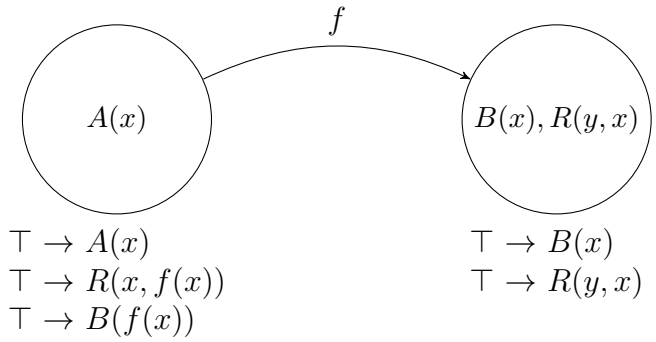



Figure 3.1: Example of a simple context structure. Context cores are represented inside each context, and context clauses for each context are written under it. The context structure represents that every instance of concept name $A$ in a model is related via role name $R$ to an instance of concept name $B$.

the ordering helps prevent the derivation of many clauses that are not relevant to deciding *any* of the target subsumptions.

To check whether $Q$ is entailed by $\mathcal{O}$, the calculus typically creates a fresh context $v$ where $\mathsf{core}_v$ is exactly the body of $Q$. Next, the rules of the calculus are applied until the context structure becomes saturated. Given an arbitrary context $v$, each of the following inference rules can be applied to $v$:

- Rule Core initialises a context $v$ by adding a context clause $\top \to A$ to $v$ for each $A \in \mathsf{core}_v$. For instance, if $\mathsf{core}_v = \{A(x), R(y, x)\}$, the rule adds context clauses $\top \to A(x)$ and $\top \to R(y, x)$ to $v$.

- Rule Hyper applies an ordered hyperresolution step [15] using context clauses in $v$ as side premises and a clause in $\mathcal{O}$ as the main premise. For instance, suppose that clauses $S(y, x) \to S(y, x)$ and $\top \to A(x) \vee B(x)$ appear in $v$, that $A(x) \not\succ_v B(x)$, where $\succ_v$ is the literal ordering on $v$, and that $S(z, x) \wedge B(x) \to C(z) \in \mathcal{O}$. Then, rule Hyper adds clause $S(y, x) \to A(x) \vee C(y)$ to $v$. Inferences by Hyper can only unify variable $x$ in $\mathcal{O}$ with variable $x$ in context clauses; this ensures that conclusions preserve the syntactic form of context clauses.

- Rule Eq uses an equality in the head of a context clause in $v$ to make a substitution in a literal in the head of another context clause in $v$. For instance, if we have context clauses $\top \to R(x, f(x))$ and $\top \to f(x) \approx g(x)$ in a context $v$, where $f(x) \succ_v g(x)$, rule Eq will derive $\top \to R(x, g(x))$ in $v$. This rule, together with the next two, implement a form of ordered paramodulation reasoning [14].

- Rule Ineq eliminates literals of the form $t \not\approx t$ from heads of context clauses in $v$. For instance, if a context clause $A(x) \rightarrow f(x) \not\approx f(x)$ appears in $v$, the rule will write $A(x) \rightarrow \bot$ in this context.

- Rule Factor replaces two equalities in the head of a context clause in $v$ by two equivalent equations, a step which is required for completeness. For instance, suppose that there exists a context $v$ with a context clause $\top \rightarrow f(x) \approx y \vee f(x) \approx g(x)$, and $f(x) \succ_v g(x)$; this rule will then add in $v$ the context clause $\top \rightarrow g(x) \not\approx y \vee f(x) \approx g(x)$.

- Rule Elim is used for redundancy elimination within a context. For instance, rule Elim will remove a clause $A(x) \rightarrow B(x)$ from $v$ if a clause $\top \rightarrow B(x)$ is added to $v$. This rule also eliminates clauses that contain tautologies in the head.

In addition to these, the calculus provides inference rules so that two neighbouring contexts can exchange information:

- Rule Succ identifies, in each context $v$, a "type" $K$ for which it must compute all relevant subsumptions of the form $K \rightarrow \Delta$, where $K$ is a conjunction of atoms of the same form as those in context cores, and $\Delta$ is a disjunction of literals. This occurs when a term $f(x)$ in a context clause of $v$ satisfies $K$; as we mentioned in the discussion of the Hyper rule, we cannot directly unify $f(x)$ with variable $x$ in an ontology clause, so we must instead look for a context that represents domain elements satisfying $K$. We say that the calculus must find a new context to *expand* $K$.

  The rule then selects or creates a context $w$ where all relevant subsumptions for $K$ can be computed. The policy for choosing $w$ (or deciding to create it) is a parameter of the calculus called the *expansion strategy*. To ensure soundness, the rule requires that $\mathsf{core}_w$ subsumes $K$, and to ensure completeness, the rule adds a clause $A \rightarrow A$ for each $A \in K \backslash \mathsf{core}_w$. Finally, it introduces an $f$-labelled edge from $v$ to $w$, if such edge does not already exist.

  To illustrate this rule, assume there is a context $v$ where a term $f(x)$ is mentioned exactly in two clauses: $\top \rightarrow B(f(x))$ and $\top \rightarrow R(x, f(x))$. Rule Succ identifies the "type" $\{B(x), R(y, x)\}$ as relevant; the rule can then select, for instance, a context $w$ with $\mathsf{core}_w = \{B(x), R(y, x)\}$, and add an $f$-labelled edge from $v$ to $w$. Figure 3.1 gives an example of a context structure that could be

generated after applying this inference rule. With a different expansion strategy, a context $w'$ with $\mathsf{core}_{w'} = B(x)$ could have been chosen instead. In that case, the rule would have added $R(y,x) \rightarrow R(y,x)$ to $w'$, because $w'$ represents instances of $B$ in a model, but not necessarily $R$-fillers. Since atom $R(y,x)$ in context clause $R(y,x) \rightarrow R(y,x)$ is readily available for applications of Hyper, the extra clause ensures that all relevant consequences for type $B(x) \wedge R(y,x)$ are computed in $w'$.[1]

- Rule Pred complements Succ by propagating back to context $v$ all relevant consequences for type $K$ that have been computed in context $w$. For this, the rule applies a hyperresolution step with clauses of $v$ as side premises and a clause of $w$ as main premise. Following our previous example for Succ, if clause $R(y,x) \rightarrow F(y)$ is derived in context $w$, the fact that context $v$ contains clauses $\top \rightarrow B(f(x))$ and $\top \rightarrow R(x,f(x))$, and an $f$-labelled edge $(v,w)$ allows the Pred rule to add clause $\top \rightarrow F(x)$ to $v$.

Once the context structure is saturated, soundness and completeness results ensure that the query holds if and only if a clause of the form $\top \rightarrow \Delta_Q$, where $\Delta_Q$ is the right-hand side of the query, has been generated in the starting context $v$. One-pass classification can be achieved by initialising the context structure with a context $v_{A_i}$ with $\mathsf{core}_{v_i} = A_i(x)$ for each unary predicate $A_i$ in $\mathcal{O}$.

## 3.3 Hybrid Methods

The CB approach has been successfully combined with other well-known reasoning methods to overcome some of the practical limitations mentioned in Section 3.1.4. For instance, it has been suggested that the problem of generating long disjunctions may be addressed by means of an algebraic reasoner [76]. The calculus in [142] for the DL $\mathcal{ELQ}$ incorporates calls to an external Integer Linear Programming component that is able to find solutions to algebraic constraint satisfaction problems based on the numeric restrictions appearing in derived consequences. This approach has been extended to the DL $\mathcal{SHOQ}$ in [68]. From a theoretical perspective, it remains unclear whether these calculi are worst-case optimal or whether they show pay-as-you-go behaviour. However, these systems have been implemented into the reasoner Avalanche [142], which shows promising results in ontologies with large number restrictions.

[1] Similarly, for $B(x)$, we have that the Core rule eventually derives $\top \rightarrow B(x)$ in $w'$.

A different kind of hybridisation is proposed in [133], which introduces the $\mathcal{SROIQ}$ reasoner Konclude. This system works very efficiently in a wide range of expressive DLs. In contrast to the calculi discussed in this chapter up to this point, Konclude is based on a tableau calculus, and therefore it attempts to answer subsumption queries by building a countermodel. However, Konclude uses an incomplete version of a CB calculus in order to generate as many (sound) consequences of the ontology as possible. These consequences are then used to aid in the construction of the tableau, and the result is a system that is highly competitive even in lightweight DLs such as $\mathcal{EL}$.

Another hybrid of tableau and CB calculus is presented in [71] for the Description Logic $\mathcal{ALCHI}$. Similarly to the calculus in Section 3.2, this calculus introduces context structures and proceeds by deriving clauses until saturation is reached. The main difference with the $\mathcal{ALCHI}$ calculus by Simančík et al. discussed in Section 3.1.4, however, lies in the treatment of disjunction. Instead of representing disjunction explicitly and using resolution-based inference rules, this algorithm makes non-deterministic choices during the saturation procedure which are reminiscent of those used by tableau-based algorithms. The calculus then introduces clauses which hold only in some models, but not necessarily in all models. The rules of the calculus ensure that some of these clauses can later be re-combined to derive sound logical consequences. Completeness of the calculus is guaranteed: all relevant subsumptions entailed by the ontology will eventually be derived as logical consequences. This mechanism prevents the generation of long disjunctions and often discovers shorter proofs of relevant entailments. The calculus is worst-case optimal and enjoys pay-as-you-go behaviour. A prototype implementing this calculus shows promising performance.

Finally, approaches that operate via black-box combination of different reasoners [140, 131, 3] often include systems based on CB reasoning for modules or fragments of ontologies in the $\mathcal{EL}$ family. For instance, the MORe [3] system addresses the problem of ontology classification by decomposing an input ontology into modules, and then classifying each module using a reasoner most suited for the language used in that module. MORe has shown that it can effectively classify ontologies using a CB reasoner on $\mathcal{ELH}$ modules, and a reasoner based on a tableau calculus, like HermiT [47], on the remaining modules. In general, this approach is particularly useful for ontologies where most GCIs belong to a lightweight DL, while a few of them belong to a much more expressive logic.

# Part II

# Calculus and Applications

# Chapter 4

# Extending Consequence-Based Reasoning to $\mathcal{ALCHOIQ}^+$

In this section we describe the key challenges in extending the CB calculus for $\mathcal{ALCHIQ}^+$ from [20] to $\mathcal{ALCHOIQ}^+$, and we give an informal overview of how our calculus approaches them. In Section 4.1 we discuss which features of CB reasoning are responsible for its desirable properties; we aim to preserve these features in our calculus. In Section 4.2 we examine a problem which emerges when trying to preserve the local nature of CB calculi, which is at odds with the possibility of imposing global constraints on ontology models using nominals; after our discussion, we sketch the intuitions underpinning our solution. Finally, in Section 4.3 we consider the challenges caused by the interaction between inverse roles, number restrictions, and nominals; we conclude by presenting an outline of the solution to this problem implemented in our calculus.

## 4.1 Key Features of CB Reasoning

Most CB calculi have been presented in the literature as "extensions" of previous CB procedures. This makes it challenging to precisely determine what qualifies as a CB reasoning procedure. In this section, we identify the properties which we consider to be distinctive of the CB approach, and we explain the advantages which they confer.

### 4.1.1 "Materialisation" of Logical Consequences

A key property of all CB calculi is that, similarly to resolution-based approaches, they proceed by deriving formulae entailed by the input ontology. This approach can have significant practical advantages over tableau and hypertableau calculi, which construct (a representation of) a model of the input ontology that can grow very

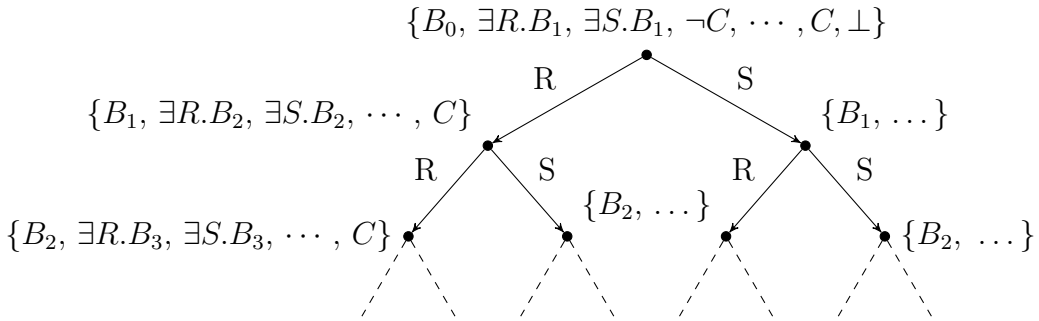



Figure 4.1: Example of a model built by a tableau-like procedure. All concepts in node labels are derived from the root downwards, except for $\bot$, which is derived last, and $C$, which is derived first at the leaves of the tree, and then propagated upwards.

large due to and-branching. We illustrate this with the following example: suppose we would like to check whether the subsumption $B_0 \sqsubseteq C$ is entailed by the following $\mathcal{EL}$ ontology:

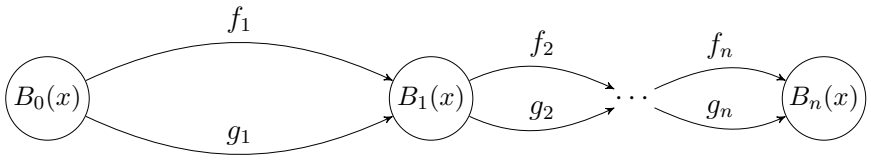
$$B_i \sqsubseteq \exists R.B_{i+1}, \text{ for } 0 \leq i \leq n-1 \qquad B_n \sqsubseteq C \qquad \exists R.C \sqsubseteq C$$

$$B_i \sqsubseteq \exists S.B_{i+1}, \text{ for } 0 \leq i \leq n-1 \qquad \exists S.C \sqsubseteq C$$

If we use a standard tableau algorithm for this task, such as the calculus for $\mathcal{ALC}$ in [10], the size of the generated model will depend on the order in which inference rules are applied. In particular, building the tableau in a breadth-first manner will lead to an exponentially large model, as illustrated in Figure 4.1.

Compare this with an application of the algorithm of Section 3.1.1, which proceeds in a linear number of steps. For each $0 \leq i \leq n$, concepts $B_i$ and $\top$ are added to $\mathcal{S}(B_i)$ at initialisation, and $C$ and $\top$ are added to $\mathcal{S}(C)$. Rule $\mathsf{CR1}$ would then add $C$ to $\mathcal{S}(C)$; afterwards, rule $\mathsf{CR3}$ would add $\langle B_i, B_{i+1}\rangle$ to at least one of $\mathcal{E}(S)$ or $\mathcal{E}(R)$ for each $0 \leq i \leq n$. Finally, rule $\mathsf{CR4}$ would add $C$ to $\mathcal{S}(B_i)$ for each $0 \leq i \leq n$, and this includes our target subsumption, since $C \in \mathcal{S}(B_0)$. For comparison, we show in Figure 4.2 how a saturated context structure for this example could look like in the calculus of Section 3.2, with an expansion strategy which, for each $1 \leq i \leq n$, uses the same context to expand types $\{R(y,x), B_i(x)\}$ and $\{S(y,x), B_i(x)\}$. Observe that the context structure is also linear in the size of the ontology.

A second desirable feature enabled by this property of CB calculi is the possibility of re-using derived consequences across queries. Since clauses derived by CB calculi are logical consequences of the ontology, they may be of help in answering




| | | | | | |
|---|---|---|---|---|---|
| $\top \to B_0(x)$ | (1) | $\top \to B_1(x)$ | (7) | $\top \to B_n(x)$ | (17) |
| $\top \to R(x, f_1(x))$ | (2) | $R(y,x) \to R(y,x)$ | (8) | $R(y,x) \to R(y,x)$ | (18) |
| $\top \to B_1(f_1(x))$ | (3) | $S(y,x) \to S(y,x)$ | (9) | $S(y,x) \to S(y,x)$ | (19) |
| $\top \to S(x, g_1(x))$ | (4) | $\top \to R(x, f_2(x))$ | (10) | $\top \to C(x)$ | (20) |
| $\top \to B_1(g_1(x))$ | (5) | $\top \to B_2(f_2(x))$ | (11) | $R(y,x) \to C(y)$ | (21) |
| $\top \to C(x)$ | (6) | $\top \to S(x, g_2(x))$ | (12) | $S(y,x) \to C(y)$ | (22) |
| | | $\top \to B_2(g_2(x))$ | (13) | | |
| | | $\top \to C(x)$ | (14) | | |
| | | $R(y,x) \to C(y)$ | (15) | | |
| | | $S(y,x) \to C(y)$ | (16) | | |

Figure 4.2: Saturated context structure for the ontology in the example of Section 4.1.1. Cores of contexts are written inside each context, and clauses attached to each context appear under it. Clauses (1), (7) and (17) are produced by Core; clauses (3) through (6), (10) through (13) and (20) are produced by Hyper from the previous clauses; clauses (8), (9), (18) and (19) are propagated by Succ from the corresponding predecessor contexts; clauses (6) and (14) are propagated by Pred from the corresponding successor contexts, and clauses (15), (16), (21) and (22) are generated by Hyper from the previous clauses.

future queries on the same ontology. In fact, CB calculi are purposefully built to take advantage of consequences that have already been derived. For instance, in the example above, any consequences generated while checking whether $B_i \sqsubseteq C$ for some $0 < i < n$ can be later reused if we would like to determine whether $B_0 \sqsubseteq C$. For a similar reason, consequences can also be reused when the input ontology $\mathcal{O}$ is extended with new clauses. This behaviour stands in contrast with calculi that make non-deterministic choices, such as tableau-based calculi, in which answering each query generally requires building a new model.

### 4.1.2 One-Pass Decision of Multiple Subsumptions

CB calculi have been developed with the goal of solving ontology classification problems. As a result, these calculi are typically able to decide entailment for a large number of subsumptions in a single execution. The exact set of subsumptions that are decided by a CB calculus is determined by the choice of initial parameters during execution. For instance, by initialising the calculus from Section 3.1.1 with $A$ and $\top$ in $\mathcal{S}(A)$ for every concept name $A$ in the input ontology, this calculus decides all subsumptions of the form $A \sqsubseteq B$, with $B$ a concept name in the input ontology. We have already seen that, as a result, most CB calculi are capable of "one-pass" classification. This feature distinguishes the CB approach from tableaux or resolution, which are typically complete only for refutation.

It is important to observe that in spite of their focus on classification, CB reasoners can also be used to check whether an ontology is consistent. For instance, if a given DL contains $\mathcal{EL}$ as a fragment, the problem of ontology consistency checking can be reduced to the problem of subsumption. This reduction requires a transformation of the input, which can be carried out without incurring in any significant complexity penalty [10]. For example, in the case of the $\mathcal{ALCHI}$ calculus mentioned in Section 3.1.4, one can solve consistency checking by testing entailment of the subsumption $\top \rightarrow \bot$, since this clause is entailed by an ontology $\mathcal{O}$ if and only if $\mathcal{O}$ is inconsistent.

The one-pass classification approach stands in contrast with *traversal*-based methods [47], which check independently all the subsumptions of the form $A \sqsubseteq B$ that are necessary to produce a classification of the ontology. Naively, one might check the quadratically many subsumptions of this form. However, traversal-based methods build a concept name hierarchy iteratively, starting from top concept $\top$ and advancing downwards, and starting from concept $\bot$ and moving upwards. These techniques

exploit the transitivity of the subsumption relation in the hierarchy to avoid checking unnecessary subsumptions. For instance, if it is known that $\mathcal{O} \models B \sqsubseteq C$ and $\mathcal{O} \not\models A \sqsubseteq C$ for input ontology $\mathcal{O}$ and concept names $A, B, C$, then it is not necessary to check whether $A \sqsubseteq B$ is entailed, since we already know that this is not the case. In practice, traversal-based methods are combined with some version of sound but incomplete CB reasoning to reduce the number of necessary checks, similarly to the methods discussed in Section 3.3.

### 4.1.3 Focus on Local Consequences

An essential ingredient of CB calculi is an emphasis on the derivation of *local* clauses. Indeed, in CB calculi for DLs without nominals, derived clauses can only be grounded to clauses that mention a single domain element and its neighbours. This can be explicitly seen in the $\mathcal{ALCHIQ}^+$ calculus from Section 3.2, where the only terms allowed in context clauses are variable $x$, terms of the form $f(x)$ for some successor function symbol $f$, and variable $y$, which is only allowed in context clauses where either the body of the clause or the context core have an atom of the form $R(x, y)$ or $R(y, x)$, thereby ensuring that $y$ is interpreted as a neighbour of the "central" element represented by $x$. This restriction prevents the calculus from generating deeper terms, such as $B(f(g(x)))$, or clauses quantified over arbitrarily distant domain elements, such as $A(x) \wedge B(y) \rightarrow C(x)$. Instead, as we have seen in Section 3.2, if the calculus derives in a context $v$ a clause of the form $A(x) \rightarrow B(f(x))$, and there is a clause of the form $B(x) \rightarrow C(x)$ in $\mathcal{O}$, the calculus will connect $v$ to a context $w$ where the relevant "type" $\{B(x)\}$ can be expanded. Similarly, other CB calculi have mechanisms to guarantee completeness while ensuring that only local clauses are derived. This has important benefits:

1. It prevents the calculus from repeating similar inferences. To see this, consider the following example, sketched using the $\mathcal{ALCHIQ}^+$ calculus from Section 3.2. Suppose we have a clause $\top \rightarrow B_0(f_i(x))$ in context $v_i$ for each $1 \leq i \leq n$. Additionally, suppose we have ontology clauses $\{B_j(x) \rightarrow B_{j+1}(x) \mid 0 \leq j < m\}$. If these were directly resolved together, we would derive $m \cdot n$ consequences of the form:

   $$\top \rightarrow B_j(f_i(x)). \tag{23}$$

   In particular, for each $1 \leq i \leq n$, $m$ clauses mentioning $f_i$ would be derived in context $v_i$.

Instead, the calculus can use the Succ rule to introduce an edge from $v_i$ to a context $w$ with core $B_0(x)$ and derive in $w$ clause $\top \to B_{j+1}(x)$ for each $1 \leq j \leq m$. Thus, the calculus performs $m$ inferences, instead of $m \cdot n$. The inferences in $w$ are a more abstract version of the $m \cdot n$ resolution inferences that would yield clauses of the form (23). Specifically, inferences in $w$ refer to *arbitrary* instances of $B_0$, whereas inferences in $v_i$ deal only with instances of $B_0$ that are $f_i$-successors of elements represented by $v_i$.

2. It contributes to the re-use of context clauses across different queries. For instance, if clause

$$A(x) \to B(f(x)) \tag{24}$$

is derived while checking subsumption $A(x) \to C(x)$, rule Succ will ensure that a clause of the form $B(x) \to B(x)$ is generated, possibly in a different context[1]. This clause can lead to further inferences of the form $B(x) \to \Delta$, which can later be re-used to answer subsumption queries of the form $B(x) \to D(x)$ on the same context structure. In contrast, if we had performed all relevant resolution inferences on the literal $B(f(x))$ in clause (24), this would not have been possible.

3. It ensures that the number of clauses that can be derived within each context is bounded, and hence it simplifies the termination and complexity arguments. The role of this is similar to the careful definition of selection functions and orderings in resolution-based methods for DLs, or the use of caching and blocking in tableau.

4. It is a pre-requisite for organising consequences in separate contexts, which enables goal-oriented behaviour, as we discuss in Section 4.1.4. This also makes CB calculi amenable to parallelisation, since inferences in each context can be carried out independently of other contexts.

### 4.1.4 Goal-Oriented Behaviour

The last essential ingredient of CB calculi is their goal-oriented behaviour. The general strategy of CB calculi consists in expanding the context structure in the minimal necessary way to derive all relevant subsumptions. This is typically achieved

[1]If the context includes $B(x)$ in the core, then the clause would be $\top \to B(x)$. To simplify the example, we assume that this is not the case.

by deriving all consequences necessary to build a countermodel for each input query $Q$ not entailed by $\mathcal{O}$. This behaviour manifests itself in up to three ways:

1. Contexts are initialised only for those "types" (conjunctions of atoms) that may be realised by some element of a countermodel. For each input query $Q = \Gamma_Q \to \Delta_Q$, CB algorithms typically introduce a context $v$ with $\mathsf{core}_v = \Gamma_Q$, because we need a model with at least one instance of $\Gamma_Q$ in order to disprove $Q$, and context $v$ represents exactly such instances. Afterwards, new contexts are created only when the $\mathsf{Succ}$ rule or its equivalent identify a new type $K$ to be expanded because some element of a countermodel can be an instance of $K$.

2. Most inference rules, including resolution and paramodulation, are only applied to clauses in the same context. Furthermore, CB inferences only resolve literals in the right-hand side of context clauses. This ensures that, in each context, derived consequences are relevant to the model elements represented by that context. On the other hand, resolution methods may resolve context clauses of the form $A_1(x) \to B_1(x)$ and $A_2(x) \to B_2(x)$ with an ontology clause $B_1(x) \wedge B_2(x) \to C(x)$ in order to produce a clause $A_1(x) \wedge A_2(x) \to C(x)$, even if expanding type $A_1(x) \wedge A_2(x)$ is unnecessary, because no element of a relevant countermodel is an instance of both $A_1$ and $A_2$.

3. CB calculi can use ordered resolution to reduce the number of inferences that are necessary to reach saturation. To preserve goal-oriented behaviour, however, atoms appearing in heads of queries should be made smallest in contexts introduced during initialisation. This is because the ordering determines how the countermodel is built. For instance, if the input query is $A_1(x) \to A_2(x)$ and we introduce a context $v$ with $\mathsf{core}_v = A_1(x)$, we can make $A_2(x)$ smaller than any other literal of the form $B(x)$ for $B \in \Sigma_A$ in $v$. This ensures that instances of $A_1$ represented by $v$ in the countermodel will not be interpreted as instances of $A_2$ unless $\top \to A_2(x)$ is derived in $v$. This is analogous to the *answer literal* technique in [49].

The combination of features 1 and 2 is reminiscent of forward chaining as it occurs in tableau-like algorithms. Indeed, to check a subsumption of the form $A(x) \to B(x)$, with $A$ and $B$ concept names, (hyper)tableau-based calculi typically introduce a fresh individual $c$, and define a model fragment which satisfies atom $A(c)$, and literal $\neg B(c)$. Then, they resolve $A(c)$ with ontology clauses in order to generate new ground facts. This approach ensures that the only resolution inferences carried out by the calculus




$$\top \to A_1(x) \quad (25)$$
$$C(x) \to C(x) \quad (26)$$




$$\top \to A_2(x) \quad (27)$$
$$\top \to B(x) \vee D(x) \quad (28)$$

Figure 4.3: Context structure for Section 4.2. Context $v_{A_i}$ has core $A_i(x)$.

are those relevant to the tableau. Thus, by initialising context structures minimally, adding contexts only on demand, and only applying resolution rules to clauses in the same context, this behaviour is imitated in CB calculi.

As a final remark, we notice that in the $\mathcal{ALCHIQ}^+$ calculus, each context $v$ is used to expand *each* conjunction of the form $\mathsf{core}_v \wedge A_1 \wedge \ldots A_n$, where for each $1 \leq i \leq n$, we have that clause $A_i \to A_i$ is in $v$. The use of a single context to expand multiple conjunctions responds to a trade-off between goal-orientedness and the desire to avoid repeated inferences. To see this, observe that if contexts $v$ and $w$ are such that $\mathsf{core}_v \subseteq \mathsf{core}_w$, then all clauses derived in $v$ will also be derived in $w$ [123]. By adding clause $A \to A$ to $v$ for each $A \in \mathsf{core}_w \backslash \mathsf{core}_v$, context $w$ is "absorbed" into $v$, and duplication of inferences is avoided. Naturally, the interaction between clauses of the form $A \to A$ in the same context can lead to irrelevant consequences. For instance, suppose clauses $B_1(x) \to C_1(x)$ and $B_2(x) \to C_2(x)$ exist in a context $v$, and let $\mathcal{O}$ contain a clause $C_1(x) \wedge C_2(x) \to D(x)$. A resolution step (via rule Hyper) will add $B_1(x) \wedge B_2(x) \to D(x)$ to $v$ even if expanding $B_1(x) \wedge B_2(x)$ is unnecessary. Thus, clause $B_1(x) \wedge B_2(x) \to D(x)$ and all clauses subsequently derived from it may be superfluous in the context structure.

## 4.2 Reconciling Locality with Nominals

Most features in the previous section can be easily preserved when extending the $\mathcal{ALCHIQ}^+$ calculus from Section 3.2 to handle nominals. There is a salient exception, however: the presence of nominals in the ontology may require the derivation of clauses that are no longer local.

To illustrate this, consider an $\mathcal{ALCHIQ}^+$ ontology $\mathcal{O}$, and suppose that the context structure from Figure 4.3 is generated when the calculus in Section 3.2 is used to classify the ontology. Next, suppose we extend $\mathcal{O}$ with clauses $C(x) \to x \approx o$ and $D(x) \to x \approx o$. Resolving clause (26) in $v_{A_1}$ with $C(x) \to x \approx o$ yields clause (29) in $v_{A_1}$. An analogous step in $v_{A_2}$ would lead to context clause (30):

$$C(x) \to x \approx o \quad (29) \qquad\qquad \top \to B(x) \vee x \approx o \quad (30)$$

Clauses (29) and (30) can interact with each other despite being in separate contexts. Indeed, given $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{O}$, clause (29) in context $v_{A_1}$ requires that the domain object interpreting individual $o$ is the only possible instance of both $A_1$ and $C$ in $\mathcal{I}$. Clause (30) in $v_{A_2}$ enforces that any instance of $A_2$ in $\mathcal{I}$ will either be an instance of $B$, or the object interpreting $o$. Now suppose $\mathcal{O}$ contains clause $A_1(x) \wedge A_2(x) \rightarrow \bot$. If the intersection of $A_1^{\mathcal{I}}$ and $C^{\mathcal{I}}$ is non-empty, then each instance of $A_2$ must be an instance of $B$. This dependency can only be represented by a non-local clause, such as the following one:

$$\forall x \, \forall y \, . \, A_1(x) \wedge C(x) \wedge A_2(y) \rightarrow B(y). \tag{31}$$

The loss of locality is highly problematic, as it leads to the loss of the advantages discussed in Section 4.1.3. In particular, the introduction of clauses such as (31) can easily compromise completeness and termination of the calculus.

Our solution to this issue hinges on the observation that non-local consequences which are necessary to prove target subsumptions can always be split into consequences involving ground atoms which satisfy a weaker form of locality. For example, we can split clause (31) into a clause $C(x) \wedge A_1(x) \rightarrow A_1(o)$ to be placed in $v_{A_1}$ (simplified as clause $C(x) \rightarrow A_1(o)$, because $A_1(x)$ is the core of $v_{A_1}$), and a clause $A(o) \wedge A_2(x) \rightarrow B(x)$ to be placed in context $v_{A_2}$ (simplified as $A_1(o) \rightarrow B(x)$). Clauses $C(x) \rightarrow A_1(o)$ and $A_1(o) \rightarrow B(x)$ are not strictly local, in the sense that an arbitrary grounding $x \mapsto t$ does not mention exclusively $t$ and its neighbours in $\mathcal{I}$; indeed, $t$ may be arbitrarily distant from $o$ in a model of $\mathcal{O}$. These clauses are, however, "local" in the sense that they are still quantified over elements of the model and their neighbours. This weaker guarantee is sufficient to recover the benefits discussed above. This solution has the additional benefit that, in the absence of nominals in $\mathcal{O}$, fully local forms are recovered.

After splitting a clause like (31), we need to ensure that ground atoms can be propagated between contexts. For instance, a ground atom $A$ appearing in a context $v$ can be propagated to a neighbouring context $w$ if a clause containing $A$ in its head participates in an inference by rule Pred as the main premise. Propagation of ground atoms across contexts may lead to the occurrence of two clauses of the form $\Gamma_1 \rightarrow \Delta_1 \vee A$ and $A \wedge \Gamma_2 \rightarrow \Delta_2$, respectively, in the same context. These clauses can then be "joined" via ground resolution on the atom $A$; this step will be implemented by a new inference rule in our calculus.

Finally, we must ensure that ground atoms are resolved with any relevant ontology clauses. While it is possible to directly resolve ground atoms with ontology clauses in




$$\top \to A_1(x) \quad (32) \qquad \top \to A_2(x) \quad (34)$$
$$D(x) \to C(x) \quad (33) \qquad \top \to B(x) \vee C(x) \quad (35)$$

Figure 4.4: Context structure for Section 4.3. Context $v_{A_i}$ has core $A_i(x)$

each context in which they appear, this can lead to a large number of repeated inferences across many contexts. To prevent this, our calculus will introduce a dedicated *root context*, in which all inferences involving ground atoms and ontology clauses are carried out.

## 4.3 Interaction of $\mathcal{I}$, $\mathcal{Q}$, and $\mathcal{O}$

Extending $\mathcal{ALCHIQ}$ with nominals presents additional difficulties caused by the interaction between nominals, inverse roles, and number restrictions [97, 60]. To illustrate this, let $\mathcal{O}$ be an ontology containing clauses (36) through (41):

$$C(x) \to R(x, f(x)) \quad (36) \qquad R(z, x) \to S(x, z) \quad (37)$$
$$C(x) \to B(f(x)) \quad (38) \qquad S(x, z_1) \wedge S(x, z_2) \to z_1 \approx z_2 \quad (39)$$
$$B(x) \to x \approx o \quad (40) \qquad A_1(x) \wedge A_2(x) \to \bot \quad (41)$$

Consider an arbitrary model $\mathcal{I}$ of $\mathcal{O}$ and an instance $a$ of $C$ in $\mathcal{I}$. Clauses (36) and (38) require that $a$ must have an $R$-filler $f(a)$ which is an instance of $B$. Clause (40) requires that $f(a)$ receives the same interpretation as $o$. Clause (37) defines role $S$ as the inverse of role $R$, so we have that the pair $\langle o^{\mathcal{I}}, a\rangle$ is in $S^{\mathcal{I}}$. Clause (39) makes $S^{\mathcal{I}}$ a functional relation, which implies that there can only be a single domain element $c$ such that $\langle o, c\rangle \in S^{\mathcal{I}}$. It follows that $a = c$, and hence all instances of $C$ must be identical in every model.

Concept $C$ plays a role similar to a nominal because, in any interpretation, it is either empty or it contains a single element. This feature weakens significantly the forest model property enjoyed by ExpTime DLs lacking either inverse roles, number restrictions or nominals. In such DLs, satisfiable ontologies always have a model consisting of a "cloud" of arbitrarily interconnected elements interpreting named individuals, a collection of tree-shaped fragments rooted in elements of this cloud, and role links between these two parts of the model. Tableau, resolution, and CB algorithms rely on the existence of such models to ensure correctness and completeness. In our example, however, the unique instance of $C$ can be connected to any other domain element, and not just to the cloud of domain elements representing

constants. Therefore, the interaction between inverse roles, number restrictions, and nominals immediately compromises the key properties of these calculi.

To see how this problem manifests itself in the CB framework from Section 3.2, assume that the contexts represented in Figure 4.4 are part of a context structure generated to classify an ontology $\mathcal{O}$. By clause (33), any element in $A_1^{\mathcal{I}} \cap D^{\mathcal{I}}$ is also in $C^{\mathcal{I}}$. If $\mathcal{O}$ is extended with clauses (36) through (41), then all instances of $A_1^{\mathcal{I}} \cap D^{\mathcal{I}}$ collapse into a single element $c$. If such element $c$ exists, then it is an instance of $A_1$. But then, clause (41), ensures that $c$ is not an instance of $A_2$. Thus, if $A_1^{\mathcal{I}} \cap D^{\mathcal{I}} \neq \emptyset$, we have $A_2^{\mathcal{I}} \cap C^{\mathcal{I}} = \emptyset$, and hence each instance of $A_2$ must be an instance of $B$ by clause (35). The dependency "*if the intersection of* $A_1$ *and* $D$ *is not empty, then every instance of* $A_2$ *is an instance of* $B$" is analogous to clause (31), but it cannot be split using the method from Section 4.2 since there is no constant in the context structure to represent $c$. Effectively, an anonymous domain element is acting as a nominal.

The tableau algorithm in [59] and the resolution calculus in [75] address these issues in a similar way: they introduce rules that name anonymous domain elements connected to a nominal $o$ via a role $R$ whenever $\mathsf{inv}(R)$ is constrained by an "at-most" number restriction. The tableau calculus directly introduces fresh elements acting as named individuals in a model, whereas the resolution calculus introduces new constants that may have different interpretations in different models. Furthermore, when applying an "at-most" number restriction with integer $n$, the tableau algorithm guesses the number of individuals (up to $n$) to be introduced; in contrast, the resolution calculus introduces $n$ constants, and it may later derive equalities between them. Fresh nominals may themselves trigger the introduction of new nominals; thus, both calculi must ensure that the generation of nominals is bounded in order to guarantee termination.

Our CB calculus will follow a similar approach to the resolution calculus in [75] by introducing a rule that can create new constants. Consider again the example from Figure 4.4. Suppose we introduce a new constant $c$ to represent the unique $S$-successor of $o$. In that case, if we derive a clause of the form $D(x) \rightarrow x \approx c$ in $v_{A_1}$ and a clause $\top \rightarrow B(x) \vee x \approx c$ in $v_{A_2}$, these clauses are analogous to clauses (29) and (30) from Section 4.2, so we could use the techniques for reasoning with nominals discussed in that section to compute all relevant consequences involving $c$. In order to ensure termination, our calculus will also impose a bound on the number of new nominals that can be introduced.

# Chapter 5

# The Consequence-Based Calculus for $\mathcal{ALCHOIQ}^+$

We next present our calculus for $\mathcal{ALCHOIQ}^+$. In Section 5.1, we give the definition of a context structure and its components, and in Section 5.2, we define the inference rules. In Section 5.3, we give the soundness and completeness claims for the calculus. In Section 5.4, we present a variant of the calculus for Horn ontologies. Finally, in Section 5.5, we illustrate the calculus in two examples. For the remainder of this section, we fix an arbitrary $\mathcal{ALCHOIQ}^+$ ontology $\mathcal{O}$ over a DL signature $\Sigma$.

## 5.1 Context Structure

We start this section by introducing the notation that we will later use for adding new constants to the context structure.

**Definition 1** (Nominal Labels)**.** *The set of* nominal labels *consists of the empty string $\epsilon$ and each string $\rho$ of the form $S_1^{i_1} \cdot S_2^{i_2} \cdot \cdots \cdot S_n^{i_n}$, with $\{n, i_1, \cdots, i_n\} \subseteq \mathbb{N}$, and $S_k \in \Sigma_S$ for each $1 \leq k \leq n$. We represent the length of a nominal label $\rho$ as $|\rho|$. The* extended set of individual names $\Sigma_u$ *contains a constant $o_\rho$ for $o$ a constant of $\mathcal{O}$ and $\rho$ a nominal label.*[1]

For convenience, we assume $\Sigma_u \subset \Sigma_c$, and distinguish within $\Sigma_u$ between *original constants*, which are the elements of the form $o_\epsilon$, and *auxiliary constants*, which are the elements of the form $o_\rho$ with $\rho \neq \epsilon$. We also assume that $\Sigma_c$ also contains a fresh constant $c$. This constant represents the "root" of the countermodel that we will build in the completeness proof. We also assume that $\Sigma_u$ contains no other constants. For any auxiliary constant $o_\rho$ we sometimes refer to $|\rho|$ as the *depth* of $o_\rho$.

[1]Observe that each constant $o$ in $\mathcal{O}$ is also in this set, since it corresponds to $o_\epsilon$.

Auxiliary constants will represent anonymous individuals that behave like nominals, that is, in the way discussed in Section 4.3. For instance, given a constant $o \in \Sigma_u$ and a role $S_1 \in \Sigma_S$, a constant of the form $o_{S_1^i}$ can be used to identify an element $t$ in a model satisfying $S_1(o,t)$. The superindices $i$ on $o_{S_1^i}$ can distinguish between different model elements $t$ satisfying $S_1(o,t)$.[2] For instance, if $S_1$ appears in an ontology clause of the form DL4:

$$B_1(x) \wedge \bigwedge_{1 \leq i \leq n+1} S_1(x, z_i) \rightarrow \bigvee_{1 \leq i < j \leq n+1} z_i \approx z_j$$

and $B_1(o)$ holds, our calculus can introduce $n$ constants $o_{S_1^1}, \cdots, o_{S_1^n}$ to represent the (at most $n$) successors of $o$ by $S_1$. The interpretation of these constants is not fixed: the calculus may later derive equalities between some of these constants, or prove that no element $t$ in the model satisfies $S_1(o,t)$, in which case the constants will be interpreted arbitrarily. Note that it is possible to introduce an auxiliary constant to identify a predecessor of another auxiliary constant. For instance, consider some $S_2 \in \Sigma_S$ which appears in an ontology clause of the form DL4 with $m+1$ equalities on the right-hand side. Suppose that $B_2(o_{S_1^i})$ holds for some $1 \leq i \leq n$; now our calculus can introduce $m$ constants of the form $o_{S_1^i \cdot S_2^j}$, with $1 \leq j \leq m$, to represent the (at most $m$) successors of $o_{S_1^i}$ by $S_2$.

The context structure used in our calculus builds on the corresponding definition for the $\mathcal{ALCHI}$ and $\mathcal{ALCHIQ}^+$ calculi discussed in Section 3.1.4. The main extension with respect to such prior work is the use of constants $u \in \Sigma_u$ in context clauses.

**Definition 2** (Context term)**.** *A* context $\mathsf{a}$-term *is a term of the form $x$, $y$, $f(x)$, $u$, or $f(u)$, where $f \in \Sigma_f$ and $u \in \Sigma_u$. A* context $\mathsf{p}$-term *is a term of the form $B(t)$, $S(x,t)$, or $S(t,x)$, where $B \in \Sigma_A$, $S \in \Sigma_S$, and $t$ is a context $\mathsf{a}$-term of the form $x$, $y$, $f(x)$, or $u$, with $f \in \Sigma_f$ and $u \in \Sigma_u$; or $S(u,t)$ or $S(t,u)$, where $S \in \Sigma_S$, $u \in \Sigma_u$, and $t$ is either in $\{y\} \cup \Sigma_u$ or of the form $f(u)$ for some $f \in \Sigma_f$.*

Notice that if we exclude context terms containing constants from $\Sigma_u$ we recover the definition of context terms in the $\mathcal{ALCHIQ}^+$ calculus from [20].

**Definition 3** (Context clause)**.** *A* context atom *is an equality of the form $A \approx \mathsf{true}$, where $A$ is a context $\mathsf{p}$-term. A* context literal *is either a context atom, or an expression of the form $l \bowtie r$, where $l$ and $r$ are $\mathsf{a}$-terms and $\bowtie \in \{\approx, \not\approx\}$. A* context clause *is a clause where the head contains only context literals, and the body contains only context atoms, or equalities of the form $u_1 \approx u_2$ with $u_1, u_2 \in \Sigma_u$.*

[2]Note that the subindex "1" in $S_1^i$ simply identifies the symbol $S_1$ in $\Sigma_S$.

Unlike previous CB calculi, our calculus allows $\mathsf{a}$-equalities in the body of context clauses; these are used to propagate information about two nominals collapsing into one. Similarly to [20], we define a *context order* to compare context literals in each context. These orders determine which inference rules are applicable in each context. Although context orders may compare literals differently in distinct contexts, there must be some level of agreement between them to avoid blocking relevant inferences. For instance, all context orders must agree on how constants are ordered, and constants in $\Sigma_u$ with longer nominal labels should be greater in any order than constants with shorter nominal labels to ensure well-foundedness. Variables $x$ and $y$ may represent model elements that can be equal, smaller, or greater than elements in $\Sigma_u$; thus, we require that the ordering must not compare variables $x$ and $y$ with any $u \in \Sigma_u$. Finally, we require, as in [20], that all literals in the head of context clauses in $v$ that need to be propagated to a predecessor context do not block literals that may be relevant for inferences in $v$. These compatibility conditions extend those presented in [20], and our definition of context order corresponds to that in [20] for ontologies without nominals.[3]

**Definition 4** (Admissible Order). *Let $\succcurlyeq$ be a total order defined on $\Sigma_f \cup \Sigma_c$. We say that $\succcurlyeq$ is* $\mathsf{a}$-admissible *if (i) $o_\rho \succcurlyeq a_{\rho'}$ whenever $|\rho| > |\rho'|$, or both $|\rho| = |\rho'|$ and $o \succcurlyeq a$; (ii) $c \succcurlyeq u$ for each $u \in \Sigma_u$, and (iii) $f \succcurlyeq u$ for each $f \in \Sigma_f$ and $u \in \Sigma_u$.*

Let $\succ$ be a strict order on context $\mathsf{a}$-terms, and $A$ an arbitrary context $\mathsf{p}$-term of the form $B(s_1)$ or $S(s_1, s_2)$, where $s_1$ and $s_2$ are context $\mathsf{a}$-terms. Let $t_1$ and $t_2$ be context $\mathsf{a}$-terms with $s_1 \succ t_1$ and $s_2 \succeq t_2$, or $s_1 \succeq t_1$ and $s_2 \succ t_2$. If $A\{s_1 \mapsto t_1, s_2 \mapsto t_2\}$ is a context $\mathsf{p}$-term, we say it is an $\mathsf{a}$-*reduction* of $A$ w.r.t. $\succ$. It is easy to see that $\{\langle A, A' \rangle \mid A' \text{ is an } \mathsf{a}\text{-reduction of } A\}$ is a transitive relation.

**Definition 5** (Context Order). *Let $\succcurlyeq$ be an $\mathsf{a}$-admissible order on $\Sigma_f \cup \Sigma_c$. A* context order *$\succ$ w.r.t. $\succcurlyeq$ is a strict order on the terms of the signature which satisfies the following:*

*1. for each context term $s \neq \mathsf{true}$, we have $s \succ \mathsf{true}$;*

*2. for each context $\mathsf{p}$-term $A$ with $A \neq \mathsf{true}$ we have $A \succ x$, and for each $u \in \Sigma_u$, we have $u \not\succ A$;*

[3] With an exception: condition (5) is more relaxed in our calculus than the analogous condition in [20]; this is merely to simplify the definition in our calculus.

*3. Order $\succ$ contains the comparisons in the lexicographic path order induced by $\succcurlyeq$ on context $\mathsf{a}$-terms; furthermore, it contains $x \succ y$, $f(x) \succ g(x)$ for each $f, g \in \Sigma_f$ with $f \succcurlyeq g$, and $f(x) \succ a$ for every $f \in \Sigma_f$ and $a \in \Sigma_c$. Finally, for every $s \in \{x, y\}$ and $u \in \Sigma_u$, we have $s \not\succ u$ and $u \not\succ s$;*

*4. Order $\succ$ is monotonic and it has the subterm property; and*

*5. Let $Z$ be the set of context atoms that have no occurrences of any $f \in \Sigma_f$, and at least one occurrence of $y$ or a constant in $\Sigma_u$. If $A \succ s$ for some context term $A$ with $A \approx \mathsf{true} \in Z$, then $s \in \{\mathsf{true}, x, y\} \cup \Sigma_u$ or $s$ is an $\mathsf{a}$-reduction of $A$ w.r.t. $\succcurlyeq$.*

In Section 6.1 we show that given any $\mathsf{a}$-admissible order $\succcurlyeq$ it is always possible to construct a context order $\succ$ w.r.t. $\succcurlyeq$. We are now ready to define the notion of a context structure.

**Definition 6.** *A* context structure *is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \theta \rangle$, where:*

- *$\mathcal{V}$ is a countable set of* contexts *which contains a* root context *$v_r$;*
- *$\mathcal{E}$ is a subset of $\mathcal{V} \times \mathcal{V} \times (\Sigma_f \cup \Sigma_u)$;*
- *$\mathsf{core}$ is a function mapping each context in $\mathcal{V}$ to a set of context atoms of the form $B(x), S(x, y), S(y, x)$, or $S(x, x)$, such that $\mathsf{core}(v_r) = \{\}$;*
- *$\mathcal{S}$ is a function that maps $v_r$ to a set $\mathcal{S}_{v_r}$ of context clauses that do not contain variable $x$, and each $v \neq v_r$ to a set of clauses that do not contain terms of the form $f(u)$ for $f \in \Sigma_f$ and $u \in \Sigma_u$;*
- *$\succcurlyeq$ is an $\mathsf{a}$-admissible order on $\Sigma_f \cup \Sigma_c$; and*
- *$\theta$ is a function that maps each context $v \in \mathcal{V}$ to a context order $\succ_v$.*

## 5.2 Inference Rules

This section presents the inference rules of our calculus. The rules extend the inference system from [20] in such a way that, in the absence of nominals, both calculi behave identically.

We first introduce the inference rules that can be independently applied to each context; we call these *single-context* rules. Let $v$ be an arbitrary, fixed context of $\mathcal{V}$. Table 5.1 shows all single-context inference rules of our calculus that can be applied to $v$. Rules Core, Hyper, Eq, Ineq, Factor, and Elim are analogous to the rules of the

same name in the $\mathcal{ALCHIQ}^+$ calculus in [20]. Rule Hyper has been modified so that, whenever it is applied to the root context $v_r$, variable $x$ in ontology clauses must unify with a constant in $\Sigma_u$. This stands in contrast with the behaviour of the rule for non-root contexts, where variable $x$ in ontology clauses must unify with variable $x$ in context clauses. Rules Eq, Ineq, and Factor implement paramodulation-based equality reasoning in the same way as in [20]; however, in our calculus, the Eq rule can be used to substitute $x$ with a constant in $\Sigma_u$ or vice-versa.

Rule Join implements a simple ground resolution step between two clauses in the same context. The purpose of this rule is to merge clauses involving ground equalities and satisfying the weaker locality condition described in Section 4.2. Finally, rule Nom introduces auxiliary constants when it detects a simultaneous interaction of an inverse role, an "at-most" number restriction, and a nominal. As discussed in Section 4.3, such constants are used to represent anonymous elements in a model behaving like named individuals. This rule is constrained by the *depth limit* $\Lambda$, a parameter of the calculus which is used to ensure that no auxiliary constants of depth greater than $\Lambda$ can be introduced in the context structure. The existence of this limit on the depth of auxiliary constants is crucial to ensure termination.

Next, we define *multi-context inference rules*, which exchange information between contexts. As in [20], we identify *trigger sets* of literals that may trigger the application of a multi-context inference rule. The set of *successor triggers* contains literals in a context $v$ which can trigger propagation of information to non-root successors of $v$ along edges labelled by symbols of $\Sigma_f$. Conversely, the set of *predecessor triggers* contains literals that need to be propagated from a non-root context $v$ to its predecessor $w$. Our definition of successor and predecessor trigger sets extends that in [20] to include ground atoms, and adds equalities of the form $x \approx u$ and $y \approx u$ for any $u \in \Sigma_u$ as predecessor triggers. Additionally, we define two new trigger sets to deal with exchanges of information involving the special root context. The set of *root successor triggers* contains literals that trigger propagation of information from a non-root context $v$ to $v_r$ along edges labelled with symbols of $\Sigma_u$. Conversely, the set of *root predecessor triggers* contains literals that trigger propagation of information from the root context to non-root predecessors of $v_r$.

**Definition 7** (Triggers)**.** *The set of* successor triggers Su *is the smallest set of atoms that contains:*

1. *atoms $B(x)$ and $\{B(u) \mid u \in \Sigma_u\}$ for each atom of the form $B(x)$ in the body of a clause in $\mathcal{O}$;*

| | | |
|---|---|---|
| Core | If | $A \in \mathsf{core}_v$, |
| | then | add $\top \to A$ to $\mathcal{S}_v$. |
| Hyper | If | $\bigwedge_{i=1}^{n} A_i \to \Delta \in \mathcal{O}$, |
| | | $\sigma$ is a substitution such that $\sigma(x) = x$ if $v \neq v_r$, and $\sigma(x) \in \Sigma_u$ otherwise, |
| | | $\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta_i \not\succ_v A_i\sigma$, for $1 \leq i \leq n$, |
| | then | add $\bigwedge_{i=1}^{n} \Gamma_i \to \bigvee_{i=1}^{n} \Delta_i \vee \Delta\sigma$ to $\mathcal{S}_v$. |
| Eq | If | $\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \in \mathcal{S}_v$ with $t_1 \not\succ_v s_1$ and $\Delta_1 \not\succ_v s_1 \approx t_1$, |
| | | $\Gamma_2 \to \Delta_2 \vee s_2 \bowtie t_2 \in \mathcal{S}_v$ with $t_2 \not\succ_v s_2$ and $\Delta_2 \not\succ_v s_2 \bowtie t_2$, where $\bowtie \in \{\approx, \not\approx\}$, |
| | | $p$ is a position such that $s_2|_p = s_1$, and $s_1 \approx t_1$ is not of the form $x \approx y$ or $y \approx u$, |
| | | if $s_2|_p = x$ then $s_2$ contains no $f \in \Sigma_f$ and no $y$, |
| | then | if $v = v_r$ and $s_1 \in \Sigma_u$, add $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[s_1/t_1] \bowtie t_2$ to $\mathcal{S}_v$, |
| | | otherwise add $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \bowtie t_2$ to $\mathcal{S}_v$. |
| Ineq | If | $\Gamma \to \Delta \vee t \not\approx t \in \mathcal{S}_v$, |
| | then | add $\Gamma \to \Delta$ to $\mathcal{S}_v$. |
| Factor | If | $\Gamma \to \Delta \vee s \approx t_1 \vee s \approx t_2 \in \mathcal{S}_v$ with $t_2 \not\succ_v s$ and $\Delta \vee s \approx t_1 \not\succ_v s \approx t_2$, |
| | then | add $\Gamma \to \Delta \vee t_1 \not\approx t_2 \vee s \approx t_2$ to $\mathcal{S}_v$. |
| Elim | If | $\Gamma \to \Delta \in \mathcal{S}_v$, |
| | | $\Gamma \to \Delta \;\hat{\in}\; \mathcal{S}_v \backslash \{\Gamma \to \Delta\}$, |
| | then | remove $\Gamma \to \Delta$ from $\mathcal{S}_v$. |
| Join | If | $\Gamma_1 \to \Delta_1 \vee A \in \mathcal{S}_v$ with $\Delta_1 \not\succ_v A$, |
| | | where $A$ is a ground atom or equality of the form $u_1 \approx u_2$ for distinct $u_1, u_2 \in \Sigma_u$ |
| | | $A \wedge \Gamma_2 \to \Delta_2 \in \mathcal{S}_v$, |
| | then | add $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2$ to $\mathcal{S}_v$. |
| Nom | If | $B_1(x) \wedge \bigwedge_{i=1}^{n+1} S_{B_2}(x, z_i) \to \bigvee_{1 \leq i < j \leq n} z_i \approx z_j \in \mathcal{O}$, |
| | | $\Gamma_1 \to \Delta_1 \vee B_1(o_\rho) \in \mathcal{S}_v$ with $\Delta_1 \not\succ_v B_1(o_\rho)$, for some $o_\rho \in \Sigma_u$ with $\lvert\rho\rvert < \Lambda$, |
| | | $\Gamma_2 \to \Delta_2 \vee S_{B_2}(o_\rho, x) \in \mathcal{S}_v$ with $\Delta_2 \not\succ_v S_{B_2}(o_\rho, x)$, |
| | then | add $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee \bigvee_{i=1}^{n} x \approx o_{\rho \cdot S^i_{B_2}}$ to $\mathcal{S}_v$. |

Table 5.1: Single-context inference rules from our calculus for an arbitrary context $v \in \mathcal{V}$. To avoid clutter, we have omitted the precondition $\Gamma \to \Delta \;\hat{\notin}\; \mathcal{S}_v$ from each inference rule except Elim, where $\Gamma \to \Delta$ is the conclusion in the inference rule.

2. *atom* $S(x, y)$ *for each atom of the form* $S(x, z_i)$ *with* $i \in \mathbb{N}$ *in the body of a clause in* $\mathcal{O}$*;*

3. *atom* $S(y, x)$ *for each atom of the form* $S(z_i, x)$ *with* $i \in \mathbb{N}$ *in the body of a clause in* $\mathcal{O}$*;*

4. *atoms* $S(u_1, u_2)$ *for each pair* $\{u_1, u_2\} \subseteq \Sigma_u$ *and predicate* $S$ *mentioned in the body of a clause in* $\mathcal{O}$*; and*

5. *each equality of the form* $u_1 \approx u_2$ *with* $\{u_1, u_2\} \subseteq \Sigma_u$*.*

*The set of* predecessor triggers $\mathsf{Pr}$ *is the smallest set that contains:*

1. *atoms* $B(y)$ *and* $\{B(u) \mid u \in \Sigma_u\}$ *for each atom of the form* $B(x)$ *in the body of a clause in* $\mathcal{O}$*;*

2. *atom* $S(x, y)$ *for each atom of the form* $S(z_i, x)$ *with* $i \in \mathbb{N}$ *in the body of a clause in* $\mathcal{O}$*;*

3. *atom* $S(y, x)$ *for each atom of the form* $S(x, z_i)$ *with* $i \in \mathbb{N}$ *in the body of a clause in* $\mathcal{O}$*;*

4. *atoms* $S(u_1, u_2)$ *for each pair* $\{u_1, u_2\} \subseteq \Sigma_u$*, whenever an atom of the form* $S(z_i, x)$ *or* $S(x, z_i)$ *appears in the body of a clause in* $\mathcal{O}$*; and*

5. *equality* $x \approx y$*, and each equality of the form* $x \approx u$*,* $y \approx u$*, or* $u_1 \approx u_2$ *with* $u, u_1, u_2 \in \Sigma_u$*.*

*The set of* root successor triggers $\mathsf{Su}^r$ *is the smallest consisting of each literal of the form* $B(u)$*,* $S(y, u)$*,* $S(u, y)$*,* $S(u_1, u_2)$*, and* $u_1 \approx u_2$*, with* $B \in \Sigma_A$*,* $S \in \Sigma_S$*, and* $\{u, u_1, u_2\} \subseteq \Sigma_u$*. The set of* root predecessor triggers $\mathsf{Pr}^r$ *extends* $\mathsf{Su}^r$ *with each atom of the form* $B(y)$ *with* $B \in \Sigma_A$*, and each equality of the form* $y \approx u$ *with* $u \in \Sigma_u$*.*

The policy for reusing a context or creating a new context when expanding the context structure is determined by an *expansion strategy*, which we define in the same way as in [125, 20].

**Definition 8.** *An expansion strategy* $\mathsf{strat}$ *is a polynomially computable function which takes as input a triple* $(f, K_1, \mathcal{D})$*, where* $f \in \Sigma_f$*,* $K_1 \subseteq \mathsf{Su}$*, and* $\mathcal{D}$ *is a context structure* $\langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succcurlyeq, \theta \rangle$*, and returns a triple* $(v, \mathsf{core}, \succ)$ *where* $\mathsf{core} \subseteq K_1$*,* $\succ$ *is a context order w.r.t* $\succcurlyeq$*, and either* $v \notin \mathcal{V}$ *(i.e. is a new context) or otherwise* $v \in \mathcal{V}$ *with* $v \neq v_r$*,* $\mathsf{core} = \mathsf{core}_v$ *and* $\succ = \succ_v$*.*

The *trivial strategy* [125] does not introduce new contexts, and always reuses the *trivial context* $v_\top$ with an empty core; it is defined as $\mathsf{triv}(f, K_1, \mathcal{D}) = \langle v_\top, \emptyset, \succ_\top \rangle$ for every $f \in \Sigma_f$, conjunction $K_1$, and context structure $\mathcal{D}$; where $\succ_\top$ is an arbitrary context order. The *eager strategy* [125] never reuses contexts, and introduces instead a fresh context for every relevant "type" discovered during saturation; it is defined as $\mathsf{eager}(f, K_1, \mathcal{D}) = \langle v_{K_1}, K_1, \succ_{K_1} \rangle$, where $v_{K_1}$ and $\succ_{K_1}$ are uniquely determined for each conjunction $K_1$. We refer the reader to [125] and [20] for additional examples of expansion strategies and a discussion of their comparative strengths and weaknesses.

Table 5.2 shows the multi-context inference rules of our calculus. Rules Succ and Pred are analogous to the corresponding rules in [20]. As in the case of Hyper, these rules have been adapted to ensure that the correct substitution is used whenever they propagate information along an edge from the root context to a non-root successor, or from a non-root successor to the root context (acting as predecessor). Furthermore, ground literals receive a special treatment in rule Pred: if they appear in the head of a clause to be propagated to a predecessor context, they are added to the head of the corresponding clause in the predecessor. Similarly, if ground atoms or equalities appear in the body of a clause to be propagated by Pred, they can be simply added to the body of the corresponding clause in the predecessor. This allows us to propagate ground atoms between contexts, and captures the fact that ground literals are global conditions which may be relevant in many contexts. To ensure soundness of the Pred rule when it propagates clauses from a non-root context $v$ to $v_r$, we require that the atoms of $\mathsf{core}_v$ should appear maximally in clauses of $v_r$, and in a suitably grounded form.

Finally, we introduce two new rules to address the case where the root context is acting as a successor of a non-root context. Rule $r$-Succ propagates relevant literals forward *to* the ground context, and works analogously to the Succ rule. Rule $r$-Pred propagates relevant literals *backward from* the root context to non-root contexts, and it works analogously to Pred.

## 5.3 Soundness and Completeness

In this section, we present the soundness and completeness statements for our calculus. Proofs of these results are given in Sections 6.2 and 6.3, respectively.

As in previous CB calculi, we define a notion of soundness for a context structure, which ensures that clauses derived in the context structure correspond to logical consequences of $\mathcal{O}$.

| | | |
|---|---|---|
| Succ | If | $\Gamma \to \Delta \vee A \in \mathcal{S}_v$ where $\Delta \not\succ_v A$, |
| | | if $v \neq v_r$ then $A$ contains $f(x)$ and otherwise $A$ contains $f(u)$ for $u \in \Sigma_u$, |
| | | there is no $\langle v, w, f\rangle \in \mathcal{E}$ such that $A' \to A' \hat{\in} \mathcal{S}_w$ for each $A' \in K_2 \backslash \mathsf{core}_w$, |
| | then | let $\langle w, \mathsf{core}', \succ'\rangle = \mathsf{strat}(f, K_1, \mathcal{D})$, |
| | | if $w \notin \mathcal{V}$ then add $w$ to $\mathcal{V}$, set $\mathsf{core}_w = \mathsf{core}'$, $\succ_w = \succ'$, and $\mathcal{S}_w = \emptyset$, |
| | | if $\langle v, w, f\rangle \notin \mathcal{E}$ then add $\langle v, w, f\rangle$ to $\mathcal{E}$, |
| | | if $v = v_r$, and $\langle v, w, u\rangle \notin \mathcal{E}$ then add $\langle v, w, u\rangle$ to $\mathcal{E}$, |
| | | add $A' \to A'$ to $\mathcal{S}_w$ for each $A' \in K_2 \backslash \mathsf{core}_w$, |
| | where | if $v \neq v_r$ then $\sigma = \{y \mapsto x, x \mapsto f(x)\}$ and otherwise $\sigma = \{y \mapsto u, x \mapsto f(u)\}$, |
| | | $K_1 = \{A' \in \mathsf{Su} \mid A'$ is an atom mentioning $x\sigma$, and $\top \to A'\sigma \in \mathcal{S}_v\}$, |
| | | $K_2 = \{A' \in \mathsf{Su} \mid \Gamma' \to \Delta' \vee A'\sigma \in \mathcal{S}_v$ with $\Delta' \not\succ_v A'\sigma\}$. |
| Pred | If | $v \neq v_r$, and there is $\langle w, v, f\rangle \in \mathcal{E}$, |
| | | $\bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C_i \to \bigvee_{i=1}^{k} L_i \in \mathcal{S}_v$, where if $w \neq v_r$ then $L_i \in \mathsf{Pr}$ for each |
| | | non-ground $L_i$, and if $w = v_r$ then for each non-ground $L_i$ |
| | | either $L_i \in \mathsf{Pr}$ or it is of the form $S(x, u)$ or $S(u, x)$, |
| | | $A_i$ is ground for $1 \leq i \leq m$, |
| | | $\Gamma_i \to \Delta_i \vee C_i\sigma \in \mathcal{S}_w$ with $\Delta_i \not\succ_w C_i\sigma$ for $1 \leq i \leq n'$, |
| | then | add $\bigwedge_{i=1}^{n'} \Gamma_i \wedge \bigwedge_{i=1}^{m} A_i \to \bigvee_{i=1}^{n'} \Delta_i \vee \bigvee_{i=1}^{k} L_i\sigma$ to $\mathcal{S}_w$, |
| | where | if $w \neq v_r$ then $\sigma = \{y \mapsto x,\ x \mapsto f(x)\}$, and $n' = n$, |
| | | and otherwise $\sigma = \{y \mapsto u, x \mapsto f(u)\}$ for some $u \in \Sigma_u$, and $\bigwedge_{i=n+1}^{n'} C_i = \mathsf{core}_v$. |
| $r$-Succ | If | $v \neq v_r$, |
| | | $\Gamma \to \Delta \vee A\sigma \in \mathcal{S}_v$ where $\Delta \not\succ_v A\sigma$ with $\sigma = \{y \mapsto x\}$, |
| | | $A \in \mathsf{Su}^r$ and contains $u \in \Sigma_u$, |
| | | there is no $\langle v, v_r, u\rangle \in \mathcal{E}$ such that $A \to A \ \hat{\in}\ \mathcal{S}_{v_r}$, |
| | then | add $\langle v, v_r, u\rangle$ to $\mathcal{E}$, |
| | | add $A \to A$ to $\mathcal{S}_{v_r}$. |
| $r$-Pred | If | $\bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C_i \to \bigvee_{i=1}^{k} L_i \in \mathcal{S}_{v_r}$, |
| | | $L_i \in \mathsf{Pr}^r$ for each nonground $L_i$, each $A_i$ is ground, |
| | | $C_i \in \mathsf{Su}^r$, and $u_i$ is the named individual in $C_i$; and |
| | | there is $\langle v, v_r, u_i\rangle \in \mathcal{E}$ for each $u_i$ such that |
| | | $\Gamma_i \to \Delta_i \vee C_i\sigma \in \mathcal{S}_v$ where $\Delta_i \not\succ_v C_i\sigma$ and $\sigma(y) = x$, |
| | then | add $\bigwedge_{i=1}^{n} \Gamma_i \wedge \bigwedge_{i=1}^{m} A_i \to \bigvee_{i=1}^{n} \Delta_i \vee \bigvee_{i=1}^{k} L_i\sigma$ to $\mathcal{S}_v$. |

Table 5.2: Multi-context inference rules from our calculus. To avoid clutter, we have omitted the precondition $\Gamma \to \Delta \hat{\notin} \mathcal{S}_v$ from Pred, and $r$-Pred, where $\Gamma \to \Delta$ is the conclusion in each rule.

**Definition 9** (Context Structure)**.** *Given a context structure* $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \theta \rangle$*, define* $\mathcal{C}_\mathcal{D}$ *as the smallest set of clauses which contains every clause of the form:*

$$B_1(o_\rho) \wedge S_{B_2}(o_\rho, x) \rightarrow \bigvee_{i=1}^{n} x \approx o_{\rho \cdot S^i_{B_2}} \tag{1}$$

*for each ontology clause of the form DL4 from Table 2.4 in* $\mathcal{O}$ *and each* $o_\rho \in \Sigma_u$ *occurring in* $\mathcal{D}$*. Context structure* $\mathcal{D}$ *is* sound *if we have:*

*S1.* $\mathcal{O} \cup \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma \rightarrow \Delta$ *for each* $v \in \mathcal{V}$ *and* $\Gamma \rightarrow \Delta$ *in* $\mathcal{S}_v$*;*

*S2.* $\mathcal{O} \cup \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \rightarrow \mathsf{core}_w\{x \mapsto f(x), y \mapsto x\}$ *for each* $\langle v, w, f \rangle \in \mathcal{E}$ *such that* $v \neq v_r$*.*

Clauses of the form (1) ensure that auxiliary constants have their intended meaning according to ontology clauses of the form DL4. Furthermore, whenever $\mathsf{core}_w$ is empty, the expression $\mathsf{core}_v \rightarrow \mathsf{core}_w\{x \mapsto f(x), y \mapsto x\}$ corresponds to the sentence $\mathsf{core}_v \rightarrow \top$, which is trivially satisfied by any interpretation. Observe that Condition S2 does not apply if $v = v_r$. This is due to the fact that outgoing edges from $v_r$ refer to specific named individuals in $v_r$, not necessarily to *every* individual represented by $v_r$.

**Theorem 1** (Soundness)**.** *Given a context structure* $\mathcal{D}$ *which is sound for* $\mathcal{O}$*, and an expansion strategy, the application of a rule from Table 5.1 or Table 5.2 to* $\mathcal{D}$ *yields a context structure which is sound for* $\mathcal{O}$*.*

In order to guarantee completeness, we need to ensure that the bound $\Lambda$ on the depth of auxiliary constants is sufficiently large. For this, we next define a lower limit to $\Lambda_0$ such that if $\Lambda \geq \Lambda_0$, the calculus in complete. The calculus need not introduce constants of depth greater than $\Lambda_0$, since we can show that in a saturated context structure there will already be constants of smaller depth and with the same interpretation.

**Definition 10** (Type)**.** *Let* $\mathsf{Su}_\tau$ *(resp.* $\mathsf{Pr}_\tau$*) be the set of elements in* $\mathsf{Su}$ *(resp.* $\mathsf{Pr}$*) of the form* $B(x)$*,* $S(y, x)$*, or* $S(x, y)$ *(resp.* $B(y)$*,* $S(y, x)$*, or* $S(x, y)$*), with* $B \in \Sigma_A$ *and* $S \in \Sigma_S$*. A* type *is a pair of the form* $\langle \Gamma, \Delta \rangle$ *where* $\Gamma \subseteq \mathsf{Su}_\tau$*, and* $\Delta \subseteq \mathsf{Pr}_\tau$*. We denote with* $\tau$ *the number of possible types, and with* $|\mathsf{Su}_\tau|$ *the number of atoms in* $\mathsf{Su}_\tau$*.*

Observe that $\tau$ is finite, since the unary and binary predicates in $\mathcal{O}$ and the number of variables are finite.

**Definition 11** (Derivation)**.** *A* derivation *is a (possibly infinite) countable sequence* $(\mathcal{D}_0, \mathcal{D}_1, \cdots)$ *where* $\mathcal{D}_0$ *is a sound context structure, and for every* $i \geq 0$ *such that* $\mathcal{D}_{i+1}$ *appears in the sequence,* $\mathcal{D}_{i+1}$ *is a sound context structure obtained by applying a rule from Tables 5.1 and 5.2 to* $\mathcal{D}_i$*.*

If there exists a finite derivation $(\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D})$, we say $\mathcal{D}$ *is derivable from* $\mathcal{D}_0$.

**Theorem 2** (Completeness)**.** *Let* $\mathcal{D}$ *be a sound context structure w.r.t.* $\mathcal{O}$ *derivable from a context structure with no auxiliary constants, and let* $\omega$ *be the number of contexts in* $\mathcal{D}$*. Suppose that the parameter* $\Lambda$ *used in the* Nom *rule is set to be greater or equal than* $\tau\omega$*, and that no rule in Table 5.1 or Table 5.2 can be applied to* $\mathcal{D}$*. Then, for each query clause* $\Gamma_Q \to \Delta_Q$ *such that* $\mathcal{O} \models \Gamma_Q \to \Delta_Q$*, and each context* $q \in \mathcal{V}$ *such that all of the following conditions hold:*

*C1. for each* $A \in \Gamma_Q$*, we have* $\Gamma_Q \to A \,\hat{\in}\, \mathcal{S}_q$*;*

*C2. for each context atom* $A \approx \mathsf{true} \in \Delta_Q$ *and each context term* $s$ *not mentioning* $y$ *and distinct from* $x$ *and* true *such that* $A \succ_q s$*, we have* $s \approx \mathsf{true} \in \Delta_Q$*;*

*we have that* $\Gamma_Q \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ *also holds.*

Please observe that Conditions C1 and C2 on the context ordering are only minimum requirements to ensure completeness. More restrictive orderings can be used to further reduce the number of clauses that must be derived to reach saturation. The requirement that $\Lambda$ must be greater or equal than $\tau\omega$ ensures that the Nom rule of the calculus can introduce a sufficiently large number of constants to represent elements of models of $\mathcal{O}$ that act like nominals, in the sense described in Section 4.3.

## 5.4 Variant of the Calculus for Horn Ontologies

We now define a variant of the calculus presented in Sections 5.1 to 5.3 which is applicable to Horn ontologies and enjoys better computational properties when checking a single subsumption with respect to Horn ontologies with nominals (as will be discussed in Chapter 7). The notion of soundness for a context structure in this variant of the calculus differs from Definition 9 in that it is relative to a conjunction $K$ of atoms of the form $B(x)$ for $B \in \Sigma_A$. In particular, the context clauses derived in $\mathcal{D}$ are only guaranteed to be satisfied by models of $\mathcal{O}$ for which the interpretation of $K$ is not empty.

| | | |
|---|---|---|
| Hyper | If | $\bigwedge_{i=1}^{n} A_i \rightarrow \Delta \in \mathcal{O}$,<br>$\sigma$ is a substitution such that $\sigma(x) = x$ or<br>$\sigma(x) \in \Sigma_u$ if either $v = v_r$ or $A_i\sigma$ contains $x$ for some $1 \leq i \leq n$,<br>$\Gamma_i \rightarrow \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta_i \not\succ_v A_i\sigma$, for $1 \leq i \leq n$, |
| | then | add $\bigwedge_{i=1}^{n} \Gamma_i \rightarrow \bigvee_{i=1}^{n} \Delta_i \vee \Delta\sigma$ to $\mathcal{S}_v$. |
| $r$-Succ | If | $v \neq v_r$,<br>$\Gamma \rightarrow \Delta \vee A\sigma \in \mathcal{S}_v$ where $\Delta \not\succ_v A\sigma$ with $\sigma = \{y \mapsto x\}$,<br>$A$ is ground, is contained in $\mathsf{Su}^r$, and contains $u \in \Sigma_u$,<br>there is no $\langle v, v_r, u\rangle \in \mathcal{E}$ such that $\top \rightarrow A \;\hat{\in}\; \mathcal{S}_{v_r}$, |
| | then | add $\langle v, v_r, u\rangle$ to $\mathcal{E}$ and $\top \rightarrow A$ to $\mathcal{S}_{v_r}$. |

Table 5.3: Modified versions of the Hyper and $r$-Succ rules.

**Definition 12.** *Let $K = B_1(x) \wedge \cdots \wedge B_n(x)$ be a conjunction of atoms where $B_i \in \Sigma_A$ for $1 \leq i \leq n$. Let $\mathcal{D}$ be a context structure $\langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \theta\rangle$ and let $\mathcal{C}_\mathcal{D}$ be the clause set defined in Definition 9. The context structure $\mathcal{D}$ is* sound for *$K$ if we have that:*

*Z1. there is a context $v \in \mathcal{V}$ with $\mathsf{core}_v = K$ such that every $w \in \mathcal{V}$ distinct from $v_r$ is reachable from $v$ or $v_r$ via edges in $\mathcal{E}$ labelled with symbols of $\Sigma_f$;*

*Z2. for every context $v \in \mathcal{V}$ and clause $\Gamma \rightarrow \Delta \in \mathcal{S}_v$, we have $\Gamma = \top$ and $\Delta$ contains at most one literal;*

*Z3. for every model $\mathcal{I} \models \mathcal{O} \cup \mathcal{C}_\mathcal{D}$ such that $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \rightarrow \bot$, we have:*

- *$\mathcal{I} \models \mathsf{core}_v \wedge \Gamma \rightarrow \Delta$ for each $v \in \mathcal{V}$ and $\Gamma \rightarrow \Delta$ in $\mathcal{S}_v$;*
- *$\mathcal{I} \models \mathsf{core}_v \rightarrow \mathsf{core}_w\{x \mapsto f(x), y \mapsto x\}$ for each $\langle v, w, f\rangle \in \mathcal{E}$ such that $v \neq v_r$; and*
- *$\mathcal{I} \not\models \mathsf{core}_v \rightarrow \bot$ for each $v \in \mathcal{V}$ with $v \neq v_r$ such that $\langle v_r, v, f\rangle \in \mathcal{E}$ for some $f \in \Sigma_f$.*

The modified calculus presented in this section replaces the Hyper rule from Table 5.1 and the $r$-Succ rule from Table 5.2 by the rules shown in Table 5.3. The version of the $r$-Succ rule in Table 5.3 differs from that in Table 5.2 as it only derives clauses with an empty body; this ensures that context clauses derived by this rule satisfy condition Z1 of Definition 12. Furthermore, the $r$-Succ rule in Table 5.3 is not triggered for atoms of the form $S(u, x)$ and $S(x, u)$, with $S \in \Sigma_S$ and $u \in \Sigma_u$, since doing so could lead to the derivation of unsound consequences. To retain completeness, the Hyper rule has been modified so that inferences which unify $x$ with some $u \in \Sigma_u$ are allowed in non-root contexts if an atom of the form $S(u, x)$ or $S(x, u)$ is selected in a context clause.

**Theorem 3** (Soundness)**.** *Suppose $\mathcal{O}$ is a Horn ontology. Given a context structure $\mathcal{D}$ which is sound for $K$, the application to $\mathcal{D}$ of a rule from Table 5.3, a rule from Table 5.1 other than* Hyper*, or a rule from Table 5.2 other than $r$-*Succ*, with the eager context strategy, yields a context structure which is sound for $K$.*

The proof of Theorem 3 is analogous to that of Theorem 1, and it is presented in Section 6.4. The completeness statement is similar to Theorem 2, but is restricted only to query clauses of the form $K \to \Delta$, where $K$ is such that the saturated context structure is sound for $K$, and $\Delta$ is an arbitrary disjunction. As a result, context structures created to decide a query clause of the form $K \to \Delta$ cannot be re-used for deciding query clauses of the form $K' \to \Delta'$, with $K' \neq K$. The completeness statement also uses a smaller lower bound for the depth limit $\Lambda$. The proof of Theorem 4 is given in Section 6.4.

**Theorem 4** (Completeness)**.** *Let $\mathcal{D}$ be a sound context structure for $K$ derivable from a context structure with no auxiliary constants, and let $\omega$ be the number of contexts in $\mathcal{D}$. Suppose that the parameter $\Lambda$ used in the* Nom *rule is set to be greater or equal than $|\mathsf{Su}_\tau|^2 \cdot \omega$, and that no rule from Table 5.3, or Table 5.1 except* Hyper*, or Table 5.2 except $r$-*Succ*, can be applied to $\mathcal{D}$. Then, for each query clause $K \to \Delta_Q$ such that $\mathcal{O} \models K \to \Delta_Q$, and each context $q \in \mathcal{V}$ satisfying the properties in condition Z1 of Definition 12, and such that all of the following conditions hold:*

*C1. for each $A \in K$, we have $\top \to A \,\hat{\in}\, \mathcal{S}_q$;*

*C2. for each context atom $A \approx \mathsf{true} \in \Delta_Q$ and each context term $s$ not mentioning $y$ and distinct from $x$ and* true *such that $A \succ_q s$, we have $s \approx \mathsf{true} \in \Delta_Q$;*

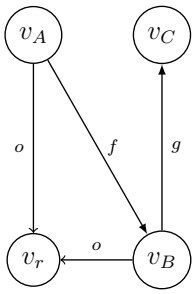
*we have that $K \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ also holds.*

## 5.5 Examples

The following examples illustrate the way in which our calculus deals with nominals. The first example focuses on non-local consequences, whereas the second illustrates how the calculus behaves when it encounters a simultaneous interaction between nominals, inverse roles, and "at-most" number restrictions.

$$\top \to A(x) \quad (12)$$
$$\top \to F(x) \vee D(x) \quad (13)$$
$$\top \to F(x) \vee x \approx o \quad (14)$$
$$\top \to F(x) \vee D(o) \quad (15)$$
$$\top \to B(f(x)) \quad (16)$$
$$\top \to R(x, f(x)) \quad (17)$$
$$\top \to F(x) \quad (18)$$

$$D(o) \to D(o) \quad (31)$$
$$C(o) \to C(o) \quad (32)$$
$$S(y, o) \to S(y, o) \quad (33)$$
$$C(o) \wedge D(o) \to E(o) \quad (34)$$
$$C(o) \wedge D(o) \wedge S(y, o) \to F(y) \quad (35)$$




$$\top \to C(x) \quad (19)$$
$$S(y, x) \to S(y, x) \quad (20)$$
$$\top \to x \approx o \quad (21)$$

$$\top \to B(x) \quad (22)$$
$$R(y, x) \to R(y, x) \quad (23)$$
$$\top \to C(g(x)) \quad (24)$$
$$\top \to S(x, g(x)) \quad (25)$$
$$\top \to g(x) \approx o \quad (26)$$
$$\top \to C(o) \quad (27)$$
$$\top \to S(x, o) \quad (28)$$
$$D(o) \to F(x) \quad (29)$$
$$D(o) \wedge R(y, x) \to F(y) \quad (30)$$

Figure 5.1: Calculus execution for Example 1. This figure summarises inferences that are relevant for deriving the query clause. Clauses (12) through (18) belong to context $v_A$ with core $A(x)$; clauses (22) through (30) belong to context $v_B$ with core $B(x)$, clauses (19) through (21) belong to context $v_C$ with core $C(x)$; all other clauses belong to $v_r$ with empty core.

**Example 1.** Consider the following ontology $\mathcal{O}_1$:

$$A(x) \to B(f(x)) \quad (2)$$
$$A(x) \to R(x, f(x)) \quad (4)$$
$$A(x) \to F(x) \vee D(x) \quad (6)$$
$$C(x) \wedge D(x) \to E(x) \quad (8)$$
$$C(x) \to x \approx o \quad (10)$$

$$B(x) \to C(g(x)) \quad (3)$$
$$B(x) \to S(x, g(x)) \quad (5)$$
$$S(z_1, x) \wedge E(x) \to F(z_1) \quad (7)$$
$$R(z_1, x) \wedge F(x) \to F(z_1) \quad (9)$$
$$D(x) \to x \approx o \quad (11)$$

We apply our calculus to prove $\mathcal{O}_1 \models A(x) \to F(x)$. In order to ensure that Condition C1 in Theorem 2 is satisfied, we start from a context structure $\mathcal{D}$ with root context $v_r$ and a context $v_A$ with core $A(x)$. To ensure the remaining completeness conditions, we assume that the context order does not make $F(x)$ bigger than any other atom, constant, or p-term not mentioning $y$. We choose an expansion strategy such that for every input $K_1$, it selects a core in the context structure containing only the unary atoms in $K_1$; if such core does not exist in $\mathcal{D}$, the strategy creates it.

Rule Core adds clause (12), which ensures that $v_A$ represents all instances of $A$ in an arbitrary model. An application of Hyper on clause (12) with ontology clause (6) yields clause (13). Since $F(x)$ cannot be bigger than $D(x)$ by our choice of $\succ_{v_A}$, we can apply Hyper to $D(x)$ in clause (13) and ontology clause (11) to obtain clause (14). Once again, we have that $F(x)$ cannot be bigger than $o$, so we can apply Eq to $x \approx o$ in clause (14) and $D(x)$ in clause (13) to generate clause (15). This clause requires that, in any model having an instance $t$ of $A$, either $t$ is also an instance of $F$, or ground atom $D(o)$ is satisfied. Ground atom $D(o)$ is then propagated to the root context

by the $r$-Succ rule, which adds clause (31) to $v_r$ and an edge $\langle v_A, v_r, o\rangle$. This clause illustrates that ground atom $D(o)$ can be relevant for the whole context structure, even though it may not hold in some models.

Clauses (16) and (17), and then clauses (19) through (25), are derived by applications of rules Core, Succ, and Hyper that are identical to those allowed by the calculus in Section 3.2, so we omit further details. Next, notice that clause (21) can be back-propagated to $v_B$ via Pred, and this leads to clause (26) in $v_B$. An application of Eq to such clause and clause (24) leads to clause (27), which is another ground clause. Please observe that this clause does not mean that $\mathcal{O} \models C(o)$; according to the soundness theorem, this clause must be interpreted as saying that $C(o)$ is satisfied only by those models where the interpretation of $B$ is non-empty. Another application of Eq with clause (26), this time to clause (25), yields clause (28). This clause, together with clause (27), can be propagated via $r$-Succ to the root context, where they are written as clause (33) and clause (32), respectively. Observe that the heads of clauses (31) and (32) unify with the body of clause (8), and since the ground context allows applications of Hyper which unify $x$ with a constant, we obtain clause (34). In turn, this clause and clause (33) can participate in a new Hyper inference with clause (7) to produce clause (35). The latter clause contains variable $y$, and hence it may be relevant in contexts representing elements connected to nominals, such as $v_B$. Rule $r$-Pred allows us to propagate clause (35) to $v_B$, using also clauses (27) and (28), to produce clause (29).

Deriving clause (29) would not have been possible without taking into account the "global" atom $D(o)$. Intuitively, we first derived in clause (31) that atom $D(o)$ may be relevant for proving the query, and now we have expanded upon that inference by proving that, if $D(o)$ is satisfied by a model, then any instance of $B$ in the model is also an instance of $F$. Clauses (23) and (29) can participate in a Hyper inference with clause (9) to produce clause (30). The head of this clause triggers Pred with clause (17) and clause (15), and propagates as clause (18) to $v_A$, which proves the query. □

Example 1 illustrates how ground literals in separate contexts can interact with each other via the root context. Although in this example the interaction involved contexts which are next to each other, it can involve in general contexts that are arbitrarily far apart from each other. Consider Figure 5.2 and suppose the ontology is such that clause (30) can be back-propagated via Pred to a clause of the form $D(o) \wedge \Gamma_n \rightarrow \Delta_n$ in a predecessor $v_{B_n}$ of $v_B$, with $\Gamma_n \rightarrow \Delta_n$ an arbitrary context clause. In turn, it could be the case that clause $D(o) \wedge \Gamma_n \rightarrow \Delta_n$ later participates in

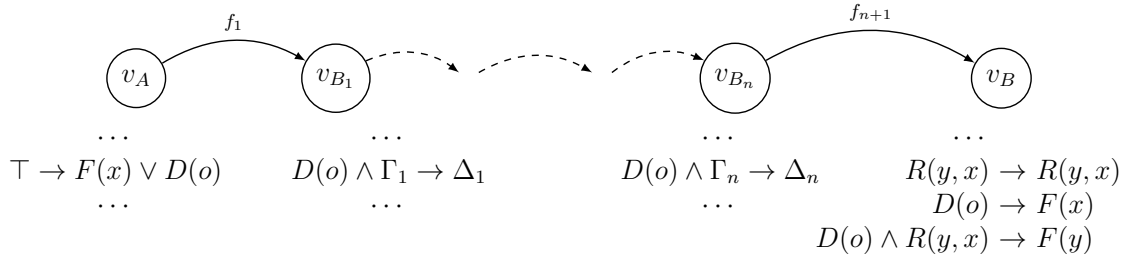



Figure 5.2: Sketch of a context structure representing an interaction analogous to Example 1, but where contexts $v_A$ and $v_B$ are arbitrarily distant

inferences which eventually result in the generation via Pred of a clause of the form $D(o) \wedge \Gamma_{n-1} \to \Delta_{n-1}$ in $v_{B_{n-1}}$, again with $\Gamma_{n-1} \to \Delta_{n-1}$ an arbitrary context clause. If this situation keeps recurring for all contexts between $v_A$ and $v_B$, eventually rule Pred may resolve the body atom $D(o)$ in clause $D(o) \wedge \Gamma_1 \to \Delta_1$ with the head atom $D(o)$ in clause (15), similarly to the last inference of Example 1.

The next example shows how our calculus deals with the interaction between nominals, inverse roles, and number restrictions.

**Example 2.** Consider again the query $A(x) \to F(x)$, and let $\mathcal{O}_2$ be the following ontology:

$$A(x) \to B_1(f(x)) \quad (36) \qquad A(x) \to B_2(g(x)) \quad (37)$$
$$A(x) \to R(x, f(x)) \quad (38) \qquad A(x) \to R(x, g(x)) \quad (39)$$
$$B_1(x) \to C(h(x)) \quad (40) \qquad B_2(x) \to C(h'(x)) \quad (41)$$
$$B_1(x) \to S_1(x, h(x)) \quad (42) \qquad B_2(x) \to S_1(x, h'(x)) \quad (43)$$
$$C(x) \to x \approx o \quad (44) \qquad S(x, z_1) \wedge S(x, z_2) \to z_1 \approx z_2 \quad (45)$$
$$B_1(x) \wedge B_2(x) \to B(x) \quad (46) \qquad R(z_1, x) \wedge B(x) \to F(z_1) \quad (47)$$
$$S_1(z_1, x) \to S_2(x, z_1) \quad (48)$$

Our initial context structure contains the root context $v_r$ and a context $v_A$ with core $A(x)$ such that the context ordering $\succ_{v_A}$ does not make $F(x)$ bigger than any other atom, constant, or p-term not mentioning $y$. In this case we choose the eager strategy, which will produce shorter clauses and simplify the example. We skip the discussion of the derivation of clauses (49) through (53), clauses (60) through (63), and clauses (81) and (82), since these clauses can be obtained through inferences that are already possible in the calculus from [20]. Such clauses ensure that every instance of $A$ in a model has an $R$-filler which is an instance of $B_1$, and every such element has an $S_1$-filler which is an instance of $C$. Clause (48) represents that $S_2$ is the inverse of $S_1$; an inference by Hyper with it and clause (82) leads to clause (83). Now, we proceed as in Example 1, and apply Hyper with clause (44) to derive clause (84). The

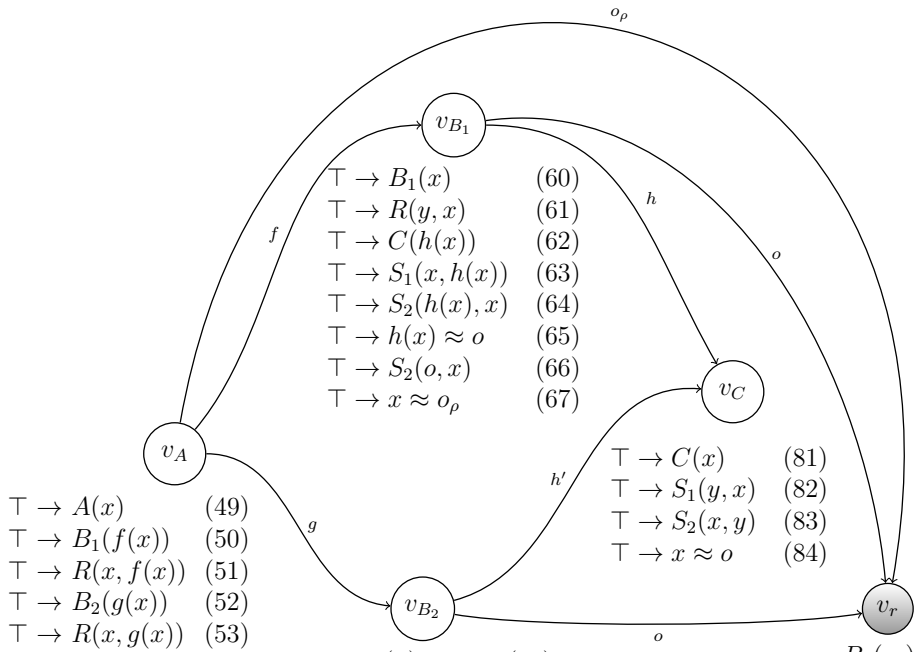



Figure 5.3: Relevant derived clauses for Example 2. Clauses (49)–(59) belong to $v_A$ with core $A(x)$; clauses (60)–(67) to $v_{B_1}$ with core $B_1(x) \wedge R(y, x)$; clauses (68)–(75) belong to $v_{B_2}$ with core $B_2(x) \wedge R(y, x)$; clauses (81)–(84) belong to $v_C$ with core $C(x) \wedge S_1(y, x)$; all other clauses belong to $v_r$ with empty core.

last two clauses in this context can be propagated via Pred to produce clauses (64) and (65). An inference with Eq which uses these two clauses produces clause (66).

The inverse role $S_2$ is now interacting with nominal $o$ and clause (45) in the way described in Section 4.3. Indeed, in a model, each instance of $B_1$ which is also an $R$-filler for some other domain element (that is, each domain element represented in context $v_{B_1}$) will be an $S_2$-filler for $o$. By clause (45), however, $o$ can only have one $S_2$-filler. Thus, there can only be a single domain element $t$ that is both an instance of $B_1$ and an $R$-filler for some other element; furthermore, any other $S_2$-filler of $o$ must be equal to $t$. Hence, the core of context $v_{B_1}$ acts like a nominal. To capture this behaviour explicitly, rule Nom derives clause (67) from clause (66) in $v_{B_1}$, with $o_\rho = o_{S_2}$ a fresh auxiliary constant. We can see clause (67) as capturing a consequence of clause (45) that is relevant for our derivation: every $S_2$-successor of $o$ in a model must be equal to the interpretation of individual $o_\rho$ in that model. Next, this clause can be back-propagated with Pred to yield clause (54) in $v_A$, which we use in two inferences by Eq on $v_A$ and clauses (50) and (51) to yield clauses (55) and (56).

The derivation illustrated so far can be straightforwardly "mirrored" using the axioms in clauses (52) and (53) and context $v_{B_2}$, and the result is the derivation of clauses (57) and (58) in $v_A$. Notice that the presence of $o_\rho$ in this context ensures that $r$-Succ is triggered and clauses (76) through (78) are generated in the root context. As in Example 1, the fact that these clauses appear in the root context allows us to apply the Hyper rule unifying $x$ with $o_\rho$, which leads to the derivation of clauses (78) and (79). The head of the later clause triggers $r$-Pred, and we can propagate it to $v_A$ with the help of clauses (55), (56) and (58). This leads to our target subsumption, represented by clause (59).

# Chapter 6

# Key Properties of the Calculus

This section contains proofs for the main properties of the calculus presented in the previous chapter. In Section 6.1, we show that given any $\mathsf{a}$-admissible order $\succcurlyeq$, there exists a context order w.r.t. $\succcurlyeq$. In Section 6.2 and Section 6.3, we prove soundess and completeness of our calculus, respectively. Finally, in Section 6.4, we prove soundness and completeness for the variant of our calculus for Horn ontologies with nominals.

## 6.1 Existence of Context Orders

This section shows how to construct a context order $\succ$ w.r.t. a given $\mathsf{a}$-admissible order $\succcurlyeq$ on $\Sigma_f \cup \Sigma_c$. For convenience, we recapitulate the definition of context order

**Definition 5** (Context Order)**.** *Let $\succcurlyeq$ be an $\mathsf{a}$-admissible order on $\Sigma_f \cup \Sigma_c$. A* context order $\succ$ *w.r.t. $\succcurlyeq$ is a strict order on the terms of the signature which satisfies the following:*

*1. for each context term $s \neq \mathsf{true}$, we have $s \succ \mathsf{true}$;*

*2. for each context $\mathsf{p}$-term $A$ with $A \neq \mathsf{true}$ we have $A \succ x$, and for each $u \in \Sigma_u$, we have $u \not\succ A$;*

*3. Order $\succ$ contains the comparisons in the lexicographic path order induced by $\succcurlyeq$ on context $\mathsf{a}$-terms; furthermore, it contains $x \succ y$, $f(x) \succ g(x)$ for each $f, g \in \Sigma_f$ with $f \succcurlyeq g$, and $f(x) \succ a$ for every $f \in \Sigma_f$ and $a \in \Sigma_c$. Finally, for every $s \in \{x, y\}$ and $u \in \Sigma_u$, we have $s \not\succ u$ and $u \not\succ s$;*

*4. Order $\succ$ is monotonic and it has the subterm property; and*

*5. Let $Z$ be the set of context atoms that have no occurrences of any $f \in \Sigma_f$, and at least one occurrence of $y$ or a constant in $\Sigma_u$. If $A \succ s$ for some context term $A$ with $A \approx \mathsf{true} \in Z$, then $s \in \{\mathsf{true}, x, y\} \cup \Sigma_u$ or $s$ is an $\mathsf{a}$-reduction of $A$ w.r.t. $\succcurlyeq$.*

To construct $\succ$, we consider the minimal term order which contains all comparisons of the following types:

Type 1 $s \succ \mathsf{true}$ for every context term $s \neq \mathsf{true}$.

Type 2 Let $f, g$ be arbitrary successor functions in $\Sigma_f$, and $a$ an arbitrary constant in $\Sigma_c$. This type includes the following comparisons: $f(x) \succ g(x)$ if $f \succcurlyeq g$, $f(x) \succ a$ for each $f \in \Sigma_f$, $a \in \Sigma_c$, and the lexicographic path order induced by $\succcurlyeq$ on context $\mathsf{a}$-terms. To simplify our notation, we define $\succ_\mathsf{a}$ as the order consisting of all comparisons of Type 2.

Type 3 $A \succ x$ and $A \succ y$ for every context $\mathsf{p}$-term $A$.

Type 4 $A_1 \succ A_2$ for each pair of distinct $\mathsf{p}$-terms $A_1$ and $A_2$ such that $A_2$ is an $\mathsf{a}$-reduction of $A_1$ w.r.t. $\succ_\mathsf{a}$.

Type 5 $A \succ s$ for each context $\mathsf{p}$-term $A$, $\mathsf{a}$-term $s$, and proper position $s$ such that $A|_p \succ_\mathsf{a} s$.

We first prove that $\succ$ is a strict order. First, note that it is straightforward to show that $\succ_\mathsf{a}$ is a strict order. To see that $\succ$ is irreflexive, notice that all comparisons listed above involve two terms of different sorts, except those of Type 1 or Type 4. However, if $A_1$ and $A_2$ are $\mathsf{p}$-terms with $A_1 \succ A_2$, then either $A_2 = \mathsf{true}$, so the comparison is of Type 1 and $A_1 \neq \mathsf{true}$, or $A_2 \neq \mathsf{true}$ so the comparison is of Type 4, and hence $A_2 \neq A_1$. To see that $\succ$ is transitive, suppose we have three terms $s_1, s_2, s_3$ such that $s_1 \succ s_2$ and $s_2 \succ s_3$. If one of them is $\mathsf{true}$, it can only be $s_3$; in that case, we have that $s_1 \succ \mathsf{true}$ is a comparison of type Type 1. If $\mathsf{true} \notin \{s_1, s_2, s_3\}$ (and hence neither $s_1 \succ s_2$ nor $s_2 \succ s_3$ is of Type 1), we consider the following cases according to the possible types of $s_1 \succ s_2$.

1. If $s_1 \succ s_2$ is of Type 2, then $s_2 \succ s_3$ can only be of Type 2, and the result follows because $\succ_\mathsf{a}$ is strict.

2. If $s_1 \succ s_2$ is of Type 3, it can only be equal to $A \succ x$ for some context $\mathsf{p}$-term $A$, with $s_2 \succ s_3$ equal to $x \succ y$, and the result follows since $A \succ y$ is a comparison of Type 3.

3. If $s_1 \succ s_2$ is of type Type 4 and $s_3$ is a $\mathsf{p}$-term, then $s_2 \succ s_3$ can only be of Type 4. But then, by transitivity of $\mathsf{a}$-reductions, $s_1 \succ s_3$ is a comparison of type Type 4. If $s_3$ is an $\mathsf{a}$-term, then $s_2 \succ s_3$ can only be of Type 3 or Type 5. In the former case, $s_1 \succ s_3$ is also a comparison of Type 3; in the latter case, let $p$ be a position such that $s_2|_p \succ_{\mathsf{a}} s_3$. Since $s_2$ is an $\mathsf{a}$-reduction of $s_1$, we have $s_1|_p \succeq_{\mathsf{a}} s_2|_p$, so $s_1|_p \succ_{\mathsf{a}} s_3$ and hence $s_1 \succ s_3$ is also a comparison of Type 5.

With this, we conclude that $\succ$ is a strict order. Now we show that is satisfies the properties of context orders. Conditions (1) and (2) are clearly satisfied due to the inclusion of comparisons of Type 1 and Type 3. Similarly, condition (3) is satisfied since each such comparison is of Type 2, and due to the fact that neither $x$ nor $y$ are compared with any constant in $\Sigma_u$. For condition (4), in order to see that $\succ$ is monotonic on context terms, suppose that $t$ is a context $\mathsf{p}$-term, and consider $\mathsf{a}$-terms such that $s_1 \succ s_2$, which could only have been added as comparisons of type Type 2. If $t$ is an arbitrary $\mathsf{p}$-term, then, since $t[s_2]_p$ is an $\mathsf{a}$-reduction of $t[s_1]_p$, we have $t[s_1]_p \succ t[s_2]_p$ is a comparison of Type 4. If $t$ is a context $\mathsf{a}$-term, in order for $t[s_1]_p$ and $t[s_2]_p$ to be a comparison of context terms, then $t[s_1]_p \succ t[s_2]_p$ is a comparison of ground terms with $s_1 \succ s_2$ also ground terms; therefore these are comparisons of Type 2 embedded in the LPO induced by $\succcurlyeq$, and since such LPO are simplification orders, and hence monotonic, the result follows. Next, observe that the order $\succ$ satisfies the subterm property due to the inclusion of comparisons of Type 5. Finally, to see that condition (5) is satisfied, suppose we have a comparison $A \succ s$ violating this property i.e. $s$ is neither a reduction of $A$ nor an element in $\{x, y, \mathsf{true}\} \cup \Sigma_c$. Since $A$ is a $\mathsf{p}$-term, this comparison can only be of Type 3, Type 4, or Type 5. However, $A \succ s$ cannot be a comparison of Type 3 since $s \notin \{x, y\} \cup \Sigma_u$. If $A \succ s$ is of Type 4, then $s$ is a reduction of $A$, so again we obtain a contradiction. Finally, if the comparison is of Type 5, then since $Z$ contains no function symbols but contains only variables or elements of $\Sigma_u$, any proper subterm of an $\mathsf{a}$-reduction of $A$ must be in $\{x, y\} \cup \Sigma_c$, which also leads to a contradiction. This concludes the proof. □

Notice that the context term defined above also satisfies condition C2, of Theorem 2. To show this, suppose there exist context terms $A$ and $s$ such that $A \approx \mathsf{true} \in \Delta_Q$, and $A \succ s$. By definition of $\succ$ and $\Delta_Q$, $A \succ s$ can only be $A \succ x$, $A \succ y$, or $A \succ \mathsf{true}$, but this contradicts the fact that $s$ does not mention $y$ and is distinct from $\mathsf{true}$ and $x$.

| | | | | | |
|---|---|---|---|---|---|
| 1 | $\mathcal{ALCHOIQ}^+$ atoms | $B(z_i)$<br>$S(x, z_i)$<br>$S(z_i, x)$ | $B(x)$<br>$S(x, x)$ | $B(u)$ | <br>$S(x, f(x))$<br>$S(f(x), x)$ |
| 2 | $\mathcal{ALCHOIQ}^+$ head a-literals | $z_i \approx z_j$ | | $x \approx u$ | $f(x) \not\approx g(x)$ |
| 3 | Context atoms (non-root) | $B(y)$<br>$S(x, y)$<br>$S(y, x)$ | $B(x)$<br>$S(x, x)$ | $B(u)$<br>$S(x, u)$<br>$S(u, x)$<br>$S(u_1, u_2)$ | $B(f(x))$<br>$S(x, f(x))$<br>$S(f(x), x)$ |
| 4 | Context a-literals (non-root) | $f(x) \bowtie y$<br>$u \bowtie y$<br>$x \bowtie y$<br>$y \bowtie y$ | $f(x) \bowtie x$<br>$u \bowtie x$<br>$x \bowtie x$ | $f(x) \bowtie u$<br>$u_1 \bowtie u_2$ | $f(x) \bowtie g(x)$ |
| 5 | Context atoms (root) | $B(y)$<br>$S(u, y)$<br>$S(y, u)$ | | $B(u)$<br>$S(u_1, u_2)$ | $B(f(u))$<br>$S(u, f(u))$<br>$S(f(u), u)$ |
| 6 | Context a-literals (root) | $f(u) \bowtie y$<br>$u \bowtie y$<br>$y \bowtie y$ | | $f(u_1) \bowtie u_2$<br>$u_1 \bowtie u_2$ | $f(u) \bowtie g(u)$ |

Table 6.1: Reference tables for literals, where $\bowtie \in \{\approx, \not\approx\}$.

## 6.2 Proof of Soundness

In this section we prove Theorem 1. For this section, we say that a structure $\mathcal{I}$ satisfies a set of clauses $\mathcal{S}$ if and only if it satisfies every clause in the set. In particular, if $\mathcal{S} = \emptyset$, then any $\mathcal{I}$ satisfies $\mathcal{S}$. If $\mathcal{S}$ is a set of clauses and $\tau$ is a substitution with range in the domain of $\mathcal{I}$, then $\mathcal{S}\tau$ is the set obtained by applying the substitution $\tau$ to every clause of $\mathcal{S}$. Given a clause $C$, we use $\mathsf{Body}(C)$ to represent the body of $C$.

**Theorem 1** (Soundness)**.** *Given a context structure $\mathcal{D}$ which is sound for $\mathcal{O}$, and an expansion strategy, the application of a rule from Table 5.1 or Table 5.2 to $\mathcal{D}$ yields a context structure which is sound for $\mathcal{O}$.*

*Proof.* Let $\mathcal{O}$ be an ontology, and let $\mathcal{D}$ be a context structure $\langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \theta \rangle$ that is sound for $\mathcal{O}$. Let $\mathcal{C}_\mathcal{D}$ be the set of clauses defined in Definition 9. We next show that applying an inference rule from Table 5.1 or Table 5.2 to $\mathcal{D}$ using an arbitrary expansion strategy produces a sound context structure. First, we show that each clause derived in an application of the inference rules is a context clause according to

Definition 2. Then, we show that any newly derived clause satisfies condition S1, and that any edge added to $\mathcal{E}$ satisfies condition S2. We discuss each inference rule as a separate case, and in each case except Succ and $r$-Succ, we denote by $C$ the (single) new clause added to the context structure, and we refer to the nomenclature used in Tables 5.1 and 5.2. The proof also uses the soundness property of hyperresolution [15]: let $\omega, \phi_i, \psi_i$, and $\gamma_i$ for each $i$ with $1 \leq i \leq n$ be arbitrary formulas with free variables in $\vec{x}$. Then, we have:

$$\left\{\forall \vec{x}. \left[\bigwedge_{i=1}^{n} \phi_i \to \omega\right]\right\} \cup \bigcup_{i=1}^{n} \{\forall \vec{x}. [\gamma_i \to \psi_i \vee \phi_i]\} \models \forall \vec{x} \left[\bigwedge_{i=1}^{n} \gamma_i \to \omega \vee \bigvee_{i=1}^{n} \psi_i\right] \quad (1)$$

*Rule* Core. In this case, $C$ is of the form $\top \to A(x)$ for some $A \in \mathsf{core}_v$. This is clearly a context clause, and sentence $\mathsf{core}_v \to A$ is a tautology so it is trivially entailed by $\mathcal{O} \cup \mathcal{C}_\mathcal{D}$.

*Rule* Hyper. To see that $C$ is a context clause, we need to prove that if $v \neq v_r$ then $\Delta\sigma$ is a disjunction of context terms in Block 3 or Block 4 of Table 6.1, and if $v = v_r$, then $\Delta\sigma$ is a disjunction of context terms of the forms in Block 5 or Block 6 of Table 6.1. Atoms from $\Delta_i$ for each $i$ with $1 \leq i \leq n$ are context literals by our assumption. Notice that literals in $\Delta$ must be of the forms in Block 1 or Block 2 of Table 6.1. Furthermore, the possible forms of body DL-atoms are: $B(x)$, $S(z_i, x)$, or $S(x, z_i)$. If $v \neq v_r$, then $\sigma(x) = x$; looking at Block 3 of Table 6.1, we can see that $z_i\sigma$ must be of the form $y$, $x$, $f(x)$, or $u$. Considering this, it is easy to check that the images by any $\sigma$ of literals in Block 1 or Block 2 of Table 6.1 correspond to literals of the form in Block 3 or Block 4 of Table 6.1. Similarly, if $v = v_r$, we have $\sigma(x) = u_1$ for some $u_1 \in \Sigma_u$, and looking at Block 5 of Table 6.1, we see that $z_i$ can only be mapped to $y, f(u_1)$, or $u_2$ for some $u_2 \in \Sigma_u$ (possibly equal to $u_1$). Once again, we can check that images by any such $\sigma$ of literals in Block 1 or Block 2 of Table 6.1 correspond to literals of the form in Block 3 or Block 4 of Table 6.1. Finally, condition S1 follows because, by assumption, we have: $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_i \to \Delta_i \vee A_i\sigma$ for each $1 \leq i \leq n$, and since $\bigwedge_{i=1}^{n} A_i \to \Delta$ is an axiom of $\mathcal{O}$, we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \bigwedge_{i=1}^{n} A_i\sigma \to \Delta\sigma$. We obtain the desired result by soundness of hyperresolution.

*Rule* Eq. Suppose that $v \neq v_r$. To see that $C$ is a context clause, first observe that the context literal $s_2 \bowtie t_2$ must be of one of the forms in Block 3 or Block 4 of Table 6.1. Next, we have that $s_1 \approx t_1$ must be of one of the forms in Block 4 with $\bowtie$ replaced by $\approx$, except $x \approx y$ and $y \approx u$. In addition, if $s_1 \approx t_1$ is of the form $u \approx x$, then $s_2 \bowtie t_2$ does not contain $f(x)$ or $y$. With this, and bearing in mind that $f(x) \succ_v \{u, x, y\}$, it is easy to check that $s_1[t_1]_p \bowtie t_2$ is another element of the form in Block 3 or

Block 4 of Table 6.1. An analogous argument proves the equivalent result in the case $v = v_r$, using Block 5 and Block 6 of Table 6.1. In particular, when the rule is applied to literals of the form $S(u_1, f(u_1))$ (resp. $S(f(u_1), u_1)$) with $s_1 \approx t_1$ of the form $u_1 \approx u_2$, the new literal is of the form $S(u_2, f(u_2))$ (resp. $S(u_2, f(u_2))$). Condition S1 follows because, by assumption, we have: $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_1 \rightarrow \Delta_1 \vee s_1 \approx t_1$, and $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_2 \rightarrow \Delta_2 \vee s_2 \approx t_2$. But then, since we have

$$s_1 \approx t_1 \wedge s_2[s_1]_p \approx t_2 \models s_2[t_1]_p \approx t_2 \wedge s_2[s_1/t_1] \approx t_2$$

due to the fact that $\approx$ is a congruence, we have both $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \rightarrow \Delta_1 \vee \Delta_2 \vee s_2[t_1] \approx t_2$ and $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \rightarrow \Delta_1 \vee \Delta_2 \vee s_2[s_1/t_1] \approx t_2$.

*Rule* Ineq. Clause $C$ is trivially a context clause. To see that condition S1 is satisfied, observe that since $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \Gamma \rightarrow \Delta \vee t \not\approx t$ by induction hypothesis, and no model satisfies $t \not\approx t$ for any context a-term $t$, we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \Gamma \rightarrow \Delta$.

*Rule* Factor. Suppose that $v \neq v_r$. To see that $C$ is a context clause, one can simply check that given any two context literals $s \approx t_1$ and $s \approx t_2$ from Block 4 of Table 6.1, the literal $t_1 \not\approx t_2$ is also in Block 4. The argument is analogous in the case $v = v_r$. To see that condition S1 is satisfied, observe that $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma \rightarrow \Delta \vee s \approx t_1 \vee s \approx t_2$ by induction hypothesis and

$$s \approx t_1 \vee s \approx t_2 \models t_1 \not\approx t_2 \vee s \approx t_2,$$

so we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma \rightarrow \Delta \vee t_1 \not\approx t_2 \vee s \approx t_2$.

*Rule* Elim. No clause is generated by this rule, so the theorem is trivially satisfied.

*Rule* Join. Clause $C$ is trivially a context clause. To see that condition S1 is satisfied, observe that since $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_1 \rightarrow \Delta_1 \vee A$ and $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge A \wedge \Gamma_2 \rightarrow \Delta_2$, by the soundness of hyperresolution, we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \rightarrow \Delta_1 \vee \Delta_2$.

*Rule* Nom. Clause $C$ is trivially a context clause. To see that condition S1 is satisfied, first observe that

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models B_1(x) \wedge \bigwedge_{i=1}^{n+1} S(x, z_i) \rightarrow \bigvee_{1 \leq i < j \leq n} z_i \approx z_j,$$

because this clause is an axiom of the ontology. To prove $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models C$, let $\mathcal{I}$ be an arbitrary model of $\mathcal{O}$, and let $\tau$ be a substitution such that $C\tau$ is ground. If $\mathcal{I} \not\models \Gamma_i\tau$ or $\mathcal{I} \models \Delta_i\tau$ for some $i \in \{1, 2\}$, then $\mathcal{I} \models C\tau$ trivially. Otherwise, we have $\mathcal{I} \not\models \Gamma_i\tau \rightarrow \Delta_i\tau$ for $i \in \{1, 2\}$. By assumption, we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \Gamma_1 \rightarrow \Delta_1 \vee B_1(o_\rho)$ and

$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \Gamma_2 \to \Delta_2 \vee S_{B_2}(o_\rho, x)$, and hence we have $\mathcal{I} \models B_1(o_\rho)\tau$ and $\mathcal{I} \models S_{B_2}(o_\rho, x)\tau$. Furthermore, by definition of $\mathcal{C}_\mathcal{D}$, we have:

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models B(o_\rho) \wedge S(o_\rho, z) \to \bigvee_{i=1}^{n} z \approx o_{\rho \cdot S^i}, \tag{2}$$

Thus, we conclude

$$\mathcal{I} \models \bigvee_{i=1}^{n} x\tau \approx o_{\rho \cdot S^i_{B_2}},$$

and therefore $\mathcal{I} \models C\tau$.

*Rule* Succ. The newly derived clauses are trivially context clauses. To see that condition S1 is satisfied, notice that each added clause is of the form $A' \to A'$, so we have that $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \wedge A' \to A'$ trivially. Finally, to prove condition S2, assume $v \neq v_r$, and notice that for every $A \in \mathsf{core}_w$, we have $\top \to A\sigma \in \mathcal{S}_v$. By soundness of $\mathcal{D}$, we have $\mathsf{core}_v \models A\sigma$; therefore, we conclude $\mathsf{core}_v \models \mathsf{core}_w\sigma$, with $\sigma = \{y \mapsto x, x \mapsto f(x)\}$, since $v \neq v_r$.

*Rule* Pred. To see that $C$ is a context clause, let $A$ be an arbitrary element in Pr and observe that $A\{y \mapsto x, x \mapsto f(x)\}$ is a literal of the forms in Block 3 or Block 4 of Table 6.1; similarly, $A\{y \mapsto u, x \mapsto f(u)\}$ is a context term of the forms in Block 5 or Block 6 of Table 6.1. To see that condition S1 is satisfied, consider first the case $v = v_r$. By assumption we have:

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v \wedge \bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C_i \to \bigvee_{i=1}^{k} L_i,$$

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \wedge \Gamma_i \to \Delta_i \vee C_i\sigma \qquad \text{for } 1 \leq i \leq n,$$

Due to the first of these claims we have:

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_v\sigma \wedge \bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C_i\sigma \to \bigvee_{i=1}^{k} L_i\sigma, \tag{3}$$

and, by soundness of hyperresolution,

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \wedge \mathsf{core}_v\sigma \wedge \bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} \Gamma_i \to \bigvee_{i=1}^{n} \Delta_i \vee \bigvee_{i=1}^{k} L_i\sigma.$$

Furthermore, by induction hypothesis we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \to \mathsf{core}_v\sigma$, so we conclude:

$$\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \wedge \bigwedge_{i=1}^{n} \Gamma_i \wedge \bigwedge_{i=1}^{m} A_i \to \bigvee_{i=1}^{n} \Delta_i \vee \bigvee_{i=1}^{k} L_i\sigma. \tag{4}$$

The argument for $w = v_r$ is analogous, albeit with a small difference: condition S2 of Definition 9 does not guarantee $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \rightarrow \mathsf{core}_v\sigma$. However, we can use the clauses of the form $\Gamma_i \rightarrow \Delta_i \vee C_i\sigma$ in $\mathcal{S}_{v_r}$ with $n+1 \leq i \leq n'$, which are such that $\bigwedge_{i=n+1}^{n'} C_i = \mathsf{core}_v$, to resolve the atoms $\mathsf{core}_v\sigma$ in the body of (3) and therefore obtain a clause analogous to (4), where $n$ is replaced by $n'$.

*Rule* $r$-Succ. The newly derived clauses are trivially context clauses that are allowed in the root contexts. To see that condition S1 is satisfied, notice that every added clause is of the form $A \rightarrow A$, so we have $\mathcal{O} \wedge \mathcal{C}_\mathcal{D} \models \mathsf{core}_w \wedge A \rightarrow A$ trivially. Furthermore, edges added in an inference by this rule are not labelled with elements of $\Sigma_f$, so S2 is already satisfied by assumption.

*Rule* $r$-Pred. To see that $C$ is a context clause, let $A$ be an arbitrary element in $\mathsf{Pr}^r$ and observe that $A\{y \mapsto x\}$ is a context term. The argument to show that condition S1 is satisfied is analogous to that used in the proof for the Pred, except that we cannot use condition S2 of Definition 9, but this is not an issue since $\mathsf{core}_v = \emptyset$.

□

## 6.3 Proof of Completeness

This section offers a proof of Theorem 2, which states that the calculus is complete for (multiple) subsumption. Our strategy consists in showing that there is a countermodel for each target subsumption that has not been derived in a saturated context structure. The section is structured as follows: in Section 6.3.1 we offer a high-level overview of the proof; in Section 6.3.2 we show how to build a fragment of the countermodel covering the "neighbourhood" of a single element $t$; in Section 6.3.3 we show that if a functional model element $t$ acts like a named individual, then it is reduced to a constant; finally in Section 6.3.4 we show how to combine the model fragments of Section 6.3.2 into a whole countermodel. For convenience, we restate Theorem 2:

**Theorem 2** (Completeness)**.** *Let $\mathcal{D}$ be a sound context structure w.r.t. $\mathcal{O}$ derivable from a context structure with no auxiliary constants, and let $\omega$ be the number of contexts in $\mathcal{D}$. Suppose that the parameter $\Lambda$ used in the* Nom *rule is set to be greater or equal than $\tau\omega$, and that no rule in Table 5.1 or Table 5.2 can be applied to $\mathcal{D}$. Then, for each query clause $\Gamma_Q \rightarrow \Delta_Q$ such that $\mathcal{O} \models \Gamma_Q \rightarrow \Delta_Q$, and each context $q \in \mathcal{V}$ such that all of the following conditions hold:*

*C1. for each $A \in \Gamma_Q$, we have $\Gamma_Q \rightarrow A \,\hat{\in}\, \mathcal{S}_q$;*

*C2. for each context atom $A \approx \mathsf{true} \in \Delta_Q$ and each context term $s$ not mentioning $y$ and distinct from $x$ and $\mathsf{true}$ such that $A \succ_q s$, we have $s \approx \mathsf{true} \in \Delta_Q$;*

*we have that $\Gamma_Q \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ also holds.*

### 6.3.1 Proof Overview

Let $\mathcal{O}$ be a fixed ontology, $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \succcurlyeq, \theta \rangle$ a saturated context structure, $\Gamma_Q \to \Delta_Q$ a query clause, and $q$ a context in $\mathcal{V}$, such that conditions C1 and C2 are satisfied w.r.t. $\Gamma_Q \to \Delta_Q$. We show the contrapositive of the theorem's statement. Assume that $\Gamma_Q \to \Delta_Q \,\hat{\notin}\, \mathcal{S}_q$; then, we show $\mathcal{O} \not\models \Gamma_Q \to \Delta_Q$ by constructing a Herbrand equality model $R^*$ of $\mathcal{O}$ containing an element $a \in \Sigma_c$ which satisfies $\Gamma_Q\{x \mapsto a\}$ but not $\Delta_Q\{x \mapsto a\}$. The element $a$ will either be $c$, the fresh constant introduced at the beginning of Section 5.1, or a constant in $\Sigma_u$.

We construct the model inductively, starting with an empty rewrite system, and extending it with a new fragment $R_t$ at each step. We use structural induction on the elements of HU. The induction order is dictated by the order $\succcurlyeq$, with one exception: $c$ is chosen before any other constant in $\Sigma_c$. This simplifies considerably the proof. At each step, given a term $t \in \mathrm{HU}$ irreducible w.r.t. any of the fragments defined so far (and mentioned in them), we define a new *model fragment* $R_t$ by mapping $t$ to a context $X_t \in C$, and ensuring that $R_t^*$ satisfies the grounding by $\{x \mapsto t, y \mapsto t'\}$ of all relevant clauses in $\mathcal{S}_{X_t}$. To ensure completeness, each fragment $R_t^*$ must satisfy the following properties:

- If $t \notin \Sigma_u$, then $R_t^*$ contains the set $\mathsf{H}_t$ of *hyperresolution triggers* of $t$, namely, each irreducible atom in $R^*$ which unifies with the body of a clause in $\mathcal{O}$ by a substitution $\sigma$ such that $x\sigma = t$.
- If $t \in \Sigma_u$, then $R_t^*$ contains all atoms in $\mathsf{H}_t \backslash \mathsf{V}_t$, where $\mathsf{V}_t \subseteq \mathsf{H}_t$ is the set of *vines* of $R^*$, containing all atoms in $\mathsf{H}_t$ of the form $S(t, s)$ or $S(s, t)$ for some element $s \notin \Sigma_c$

Intuitively, we can think of this property as requiring that each $R_t^*$ must be "complete" for $t$. In the case where $t \in \Sigma_u$, the atoms in $\mathsf{V}_t$, which represent connections between an element of $\Sigma_c$ and some arbitrary element of a tree-like part of the model, may be scattered across other fragments of $R^*$. However, this will not be problematic, because if we let $\mathsf{V}_{t,s}$ be the subset of $\mathsf{V}_t$ containing atoms of the form $S(s, t)$ or $S(t, s)$, we have that all atoms in $\mathsf{V}_{t,s}$, as well as all atoms in $\mathsf{H}_s$ that interact with them, appear in the fragment $R_s^*$.

Ensuring that $R_t^*$ is complete for $t$ in an ontology with nominals is particularly challenging when the ontology contains DL-clauses with multiple neighbour variables $z_i$ in the body. To see this, let $C$ be a clause with atoms $S(x, z_1)$ and $S(x, z_2)$, and let $\sigma$ be a substitution such that $x\sigma = u$ for some $u \in \Sigma_u$, and both $z_1\sigma$ and $z_2\sigma$ are ground and irreducible. We would like to ensure that there exists a model fragment which contains atoms $S(u, z_1\sigma)$, $S(u, z_2\sigma)$. However, $z_1\sigma$ and $z_2\sigma$ can be arbitrary terms in HU, which makes it difficult to construct the fragments $R_t^*$. Rule $\mathsf{Nom}$ addresses this challenge by ensuring that atoms like $S(u, z_1\sigma)$ and $S(u, z_2\sigma)$ can be found in the fragment for $u$, that is, $R_u^*$. In particular, the $\mathsf{Nom}$ rule ensures that $z_1\sigma$ and $z_2\sigma$ are in $\Sigma_u$ by introducing new constants to identify $S$-fillers for $u$ (and hence $S$-fillers of $u$ are always reduced to constants in $R$), since ground atoms where all terms are in $\Sigma_u$ are part of the neighbourhood of any term. We also show that when $\mathsf{Nom}$ is not applicable, there exist other clauses in the context which ensure that any $S$-fillers for $u$ are reduced to a constant.

To ensure that $R_t^*$ is complete for $t$, we require that the fragment $R_t^*$ must contain *every* element in $\mathsf{H}_t$ that appears in previous fragments, and it does not *introduce any* element in $\mathsf{H}_s$ for some term $s$ whose fragment $R_s^*$ has already been defined, and that it does not reduce some term $s$ which is currently irreducible w.r.t. fragments already defined. To accomplish this, for each $t \in \mathrm{HU}$ we define a set $\Gamma_t$ of *required literals*, which contains all equalities that *must* be satisfied by $R_t^*$ according to the current global model, and a set $\Delta_t$ of *forbidden literals* which contains all literals that *must not* be generated in $R_t^*$. We then have that $R_t^*$ must disprove the "query" $\Gamma_t \to \Delta_t$.

The conjunction $\Gamma_t$ guides how to choose a context from $\mathcal{V}$ for each term $t$. In particular, to construct model fragment $R_t^*$, we choose a context $v$ where the non-ground form of $\Gamma_t$ has been expanded; for instance, if $B(t) \in \Gamma_t$ for some $B \in \Sigma_A$, we need a context that expands (at least) atom $B(x)$. The fragment $R_t^*$ is constructed using a variant of a standard deterministic procedure for proving correctness of resolution-based calculi [15].

Our proof technique is similar to that used in [20] to prove completeness of the $\mathcal{ALCHIQ}^+$ consequence-based algorithm. However, our case presents several challenges which invalidate many of the intermediate results in their proof. In particular, the model generated in the proof by Bate et al. has a single, anonymous "root" element, which spawns a (possibly infinite) tree. By contrast, our model consists of a cluster of "root" elements, corresponding to elements of $\Sigma_c$, which can be arbitrarily interconnected. Furthermore, each of these root elements can spawn a (possibly infinite) tree. Finally, anonymous elements in the tree can be directly linked to root

elements through a binary predicate. Thus, while the overall structure of the proof remains the same, most intermediate results have been extended and/or fundamentally modified.

For instance, the construction of fragment $R_c^*$ decides which function-free ground equalities constructed with constants in $\Sigma_u$ are satisfied in $R^*$. To ensure that no new equalities of this form are introduced by other fragments $R_t^*$, we define a set $\Delta_{\mathsf{Rt}}$ which contains all function-free, ground literals (literals about"root" elements of the model) mentioning only elements of $\Sigma_u$ which do not appear in $R_c^*$, and we require $\Delta_{\mathsf{Rt}} \subseteq \Delta_t$ for every $t \neq c$. Furthermore, we need to ensure that all function-free, ground equalities in $R_c^*$ mentioning only elements in $\Sigma_u$ (equalities about "root" elements of the model) are satisfied in any $R_t^*$, so that: (i) function-free, ground atoms mentioning only elements in $\Sigma_u$ which are selected in $R_c^*$ can participate in inferences in every context, and (ii) any constants of $\Sigma_u$ appearing in $R_t$ can be reduced to their normal form w.r.t. $R_c^*$. To accomplish this, we define $\Gamma_{\mathsf{Rt}}$ as the fragment with all function-free, ground equalities in $R_c^*$ mentioning only elements in $\Sigma_u$ and ensure $\Gamma_{\mathsf{Rt}} \subseteq \Gamma_t$ for each $t \neq c$.

Another difficulty is the fact that, in contrast to the proof by Bate et al., no bijection exists now between clauses in $\mathcal{S}_{X_t}$ and their grounding by $x\sigma = t$ (and $y\sigma = t'$, if $t = f(t')$ for some $f \in \Sigma_f$). This forces us to consider multiple non-ground clauses that could have generated a particular ground clause. Finally, the proof by Bate et al. relies on the fact that atoms in $\Delta_t$ can always be made smaller in the grounding of the context order $\succ_{X_t}$ to build each fragment $R_t^*$ using the method in [15]. However, this is no longer true in our case; to work around this issue, we have devised a new variant of the fragment construction method in [15]. □

We conclude this introduction by stating and proving a preliminary lemma for rewrite systems which we will later need in the proof.

**Lemma 4.** *Let $R$ be an arbitrary Church-Rosser rewrite system $R$, and let $\succ$ be a simplification order embedding the rules of $R$. Let $t_1, t_2$ be a pair of terms with $R^* \models t_1 \not\approx t_2$. For any pair of terms $s_1 \succ s_2$ such that $s_1 \succ \{t_1, t_2\}$, if the rewrite system $\hat{R} = R \cup \{s_1 \Rightarrow s_2\}$ is Church-Rosser, then $\hat{R} \models t_1 \not\approx t_2$.*

*Proof.* Since $R$ is Church-Rosser, let $t_1'$ and $t_2'$ be the unique normal forms of $t_1$ and $t_2$ with respect to $R$; observe that we have $R^* \models t_1' \not\approx t_2'$ since $R^*$ is a congruence. Since $\succ$ is a simplification order, we have $t_1 \succeq t_1'$ and $t_2 \succeq t_2'$; hence, $s_1 \succ t_1'$ and $s_1 \succ t_2'$. Furthermore, since $\succ$ is a simplification order, $s_1$ is neither a subterm of $t_1'$ nor $t_2'$. Thus, there is no position $p$ such that rewrite rule $s_1 \Rightarrow s_2$ can be applied to

either $t'_1|_p$ or $t'_2|_p$. This, together with the fact that $t'_1$ and $t'_2$ are irreducible by $R$, implies that $t'_1$ and $t'_2$ are irreducible by $\hat{R}$. Since $\hat{R}$ is Church-Rosser, every term has a unique normal form, so $t'_1$ and $t'_2$ are still the normal forms of $t_1$ and $t_2$, respectively. Furthermore, $\hat{R}^* \not\models t'_1 \approx t'_2$, and thus $\hat{R} \models t_1 \not\approx t_2$ since $\hat{R}$ is a congruence. □

### 6.3.2 The Model Fragment $R_t^*$

This section defines, for each $\mathsf{a}$-term $t$ in the countermodel, the model fragment $R_t^*$ which covers the "neighbourhood" of $t$ and disproves the query $\Gamma_t \to \Delta_t$, which is either the target query, or expresses a compatibility condition with the fragments defined previously.

The section is organised as follows: in Section 6.3.2.1 we state a set of conditions that must hold as a prerequisite for building the fragment for $t$; in Section 6.3.2.2 we formally define the neighbourhood of $t$, in Section 6.3.2.3 we define a total order on the terms in the neighbourhood, which is necessary for the fragment-building method; in Section 6.3.2.4 we define the fragment of clauses $N_t$ that we use in the fragment-building method; in Section 6.3.2.5 we describe this method, which is based on standard canonical model construction methods for resolution calculi; in Section 6.3.2.6 we show that the fragment satisfies the clauses in $N_t$; finally, in Section 6.3.2.7 we show that the model fragment refutes the query $\Gamma_t \to \Delta_t$.

#### 6.3.2.1 Prerequisites

In order to build the model fragment, we assume that the following elements are given:

- A term $t$.
- A context $v$ in $\mathcal{D}$, which is $q$ if $t = c$ and $v_r$ if $t \in \Sigma_u$.
- A conjunction of equalities $\Gamma_t$.
- A disjunction of literals $\Delta_t$ which is disjoint with $\Gamma_t$.

Define *root terms* as those which are either in $\Sigma_u$ or function-free, ground $\mathsf{p}$-term mentioning only constants in $\Sigma_u$. Let $\mathsf{Rt}$ be the set of all literals built using root terms and $\mathsf{true}$. We also assume that there exists a fixed Church-Rosser rewrite system $R_{\mathsf{Rt}}$ embedded in $\succcurlyeq$ which contains only rules of the form $B(u) \Rightarrow \mathsf{true}$, $S(u_1, u_2) \Rightarrow \mathsf{true}$, or $u_1 \Rightarrow u_2$. We assume there exists a set $\Gamma_{\mathsf{Rt}}$ consisting of all equalities $s \approx l$ for each $s \Rightarrow l \in R_{\mathsf{Rt}}$, and we let $R^*_{\mathsf{Rt}}$ be the Herbrand equality interpretation induced by this

system. Similarly, we assume that there exists a fixed set of literals $\Delta_{\mathsf{Rt}}$ of the form $B(u)$, $S(u_1, u_2)$, or $u_1 \bowtie u_2$, with $\bowtie \in \{\approx, \not\approx\}$. Intuitively, $R^*_{\mathsf{Rt}}$ and $\Delta_{\mathsf{Rt}}$ represent a partition on the set of function-free ground equalities built using only constants in $\Sigma_u$, where $R^*_{\mathsf{Rt}}$ contains those satisfied by the model, and $\Delta_{\mathsf{Rt}}$ the rest; $\Delta_{\mathsf{Rt}}$ also contains all inequalities corresponding to $\mathsf{a}$-equalities not in $\Delta_{\mathsf{Rt}}$. To enforce this, we assume that the following properties are satisfied:

L1. For any $u_1, u_2$ in $\Sigma_u$, we have $u_1 \approx u_2 \in \Delta_{\mathsf{Rt}}$ and $u_1 \approx u_2 \notin R^*_{\mathsf{Rt}}$ if and only if it is not the case that $u_1 \not\approx u_2 \in \Delta_{\mathsf{Rt}}$ and $u_1 \approx u_2 \in R^*_{\mathsf{Rt}}$.

L2. For any atom $A$ of the form $B(u)$ or $S(u_1, u_2)$, $A \in \Delta_{\mathsf{Rt}}$ if and only if $A \notin R^*_{\mathsf{Rt}}$.

Notice that condition L1 implies $u \not\approx u \in \Delta_{\mathsf{Rt}}$ for every $u \in \Sigma_u$ since $u \approx u \in R^*_{\mathsf{Rt}}$ always. For $\Gamma_t$ and $\Delta_t$, the following conditions are assumed:

L3. If $t = c$, then $\Delta_c = \Delta_Q \sigma_t$; if $t \in \Sigma_u$, then $\Gamma_t = \Gamma_{\mathsf{Rt}}$ and $\Delta_t = \Delta_{\mathsf{Rt}}$; otherwise, $\Gamma_t \subseteq \mathsf{Su}_t$ and $\Delta_t \subseteq \mathsf{Pr}_t \cup \Delta_{\mathsf{Rt}}$.

L4. If $t \neq c$, we have $\Gamma_{\mathsf{Rt}} \subseteq \Gamma_t$ and $\Delta_{\mathsf{Rt}} \subseteq \Delta_t$; also, $\Delta_t \cap \mathsf{Rt} \subseteq \Delta_{\mathsf{Rt}}$.

L5. If $t \in \Sigma_u$, then we have $t \approx u \in \Delta_t$ for every $u \in \Sigma_u$ with $t \succcurlyeq u$. If $t = f(t')$ we have $t \approx u \in \Delta_t$ for each $u \in \Sigma_u$, $t' \approx u \in \Delta_t$ for each $u \in \Sigma_u$ with $t' \succcurlyeq u$, and $t \approx t' \in \Delta_t$.

Let $\pi_t$ be a function which removes all elements in $\Delta_t$ from a disjunction. Define $N_t$ as the set:

$$\{\Gamma\sigma_t \to \pi(\Delta\sigma_t) \mid \Gamma \to \Delta \in \mathcal{S}_v, \text{ both } \Gamma\sigma_t \text{ and } \Delta\sigma_t \text{ are ground, and } \Gamma\sigma_t \subseteq \Gamma_t.\}$$

Finally, we assume that the following conditions are satisfied:

L6. $\Gamma_t \to \bot \hat{\notin} N_t$;

L7. For each $A \in \Gamma_t$, we have $\Gamma_t \to A \,\hat{\in}\, N_t$; in particular, for $t \neq c$, if $A \in \Gamma_{\mathsf{Rt}}$, then $A \to A \,\hat{\in}\, \mathcal{S}_v$.

#### 6.3.2.2 The Neighbourhood of $t$

**Definition 13.** *We define the* a-neighbourhood *of each term* $t \in \mathrm{HU}$ *as the set which contains term* $t$*, the term* $f(t)$ *for each* $f \in \Sigma_f$*, the predecessor* $t'$ *of* $t$ *if it exists, and every constant in* $\Sigma_u$*.*

Elements in the a-neighbourhood of $t$ will be treated as constants for the remainder of this section. Thus, we assume that terms like $t$, $f(t)$, or $t'$ do not have subterms at proper positions.

**Definition 14.** *We define the* p-neighbourhood *of each term* $t \in \mathrm{HU}$ *as the smallest set containing the* p*-terms* $B(t)$*,* $S(t,t)$*,* $S(t,f(t))$*,* $S(f(t),t)$*,* $S(t,u_1)$*,* $S(u_1,t)$*,* $B(f(t))$*,* $B(u_1)$*,* $S(u_1,u_2)$*, for each* $f \in \Sigma_f, B \in \Sigma_A, S \in \Sigma_S, \{u_1,u_2\} \subseteq \Sigma_u$*; and also, if* $t'$ *exists, the* p*-terms* $S(t,t')$*,* $S(t',t)$*,* $B(t')$ *for each* $B \in \Sigma_A$*,* $S \in \Sigma_S$*.*

The *neighbourhood* of a term $t \in \mathrm{HU}$ is the union of its a-neighbourhood and its p-neighbourhood.

**Definition 15.** *We define the* grounding substitution $\sigma_t$ *of a term* $t \in \mathrm{HU}$ *as the substitution* $\{x \mapsto t, y \mapsto t'\}$ *if* $t'$ *exists, and* $\{x \mapsto t\}$ *otherwise.*

We define also:

$$\mathsf{Su}_t = \{A\sigma_t \mid A \in \mathsf{Su} \text{ and } A\sigma_t \text{ is ground } \}$$

$$\mathsf{Pr}_t = \{A\sigma_t \mid A \in \mathsf{Pr} \text{ and } A\sigma_t \text{ is ground } \}$$

$$\mathsf{Ref}_t = \{S(t,t) \mid S \text{ is a binary atom } \}$$

$$\mathsf{Nom}_t = \{A\sigma_t \mid A \in \{S(x,u), S(u,x)\} \text{ for some } u \in \Sigma_u \text{ and } S \in \Sigma_S \ \}$$

$$Z_t = \{A\sigma_t \mid A \in Z\},$$

where $Z$ is the set of function-free context atoms which have at least one occurrence of $y$ or a constant from $\Sigma_u$.

#### 6.3.2.3 Simplification Order on the Neighbourhood of $t$

For the construction of the model fragment, we define a *total simplification order* $>_t$ on the set of ground terms in the neighbourhood of $t$ in a way which satisfies the following conditions for any pair of context terms $s_1, s_2$:

O1. If $s_1 \succ_v s_2$, then $s_1\sigma_t >_t s_2\sigma_t$;

O2. If $t = c$, having $s_1\sigma_c \approx \mathsf{true} \in \Delta_c$ and $s_1\sigma_c >_c s_2\sigma_c$ implies $s_2\sigma_c \in \{c, \mathsf{true}\} \cup \Sigma_u$, $s_2\sigma_c \approx \mathsf{true} \in \Delta_c$, or $s_2\sigma_c$ is an $\mathsf{a}$-reduction of a term $s_3$ w.r.t. $\gtrdot$ such that $s_3 \approx \mathsf{true} \in \Delta_c$; and

O3. If $t \neq c$, having $s_1\sigma_t \approx \mathsf{true} \in \Delta_t$ and $s_1\sigma_t >_t s_2\sigma_t$ implies that $s_2\sigma_t \in \{t, t', \mathsf{true}\} \cup \Sigma_u$, or $s_2\sigma \approx \mathsf{true} \in \Delta_t$, or $s_2\sigma_t$ is an $\mathsf{a}$-reduction of a term $s_3$ w.r.t. $\gtrdot$ such that $s_3 \approx \mathsf{true} \in \Delta_t$.

To define $>_t$, we use the following strategy: first, we define the partial order $\succ_t$ as the order consisting of comparison $s_1\sigma_t \succ_t s_2\sigma_t$ for each pair of context terms $s_1, s_2$ such that $s_1 \succ_v s_2$. Then, we totalise $\succ_t$ to obtain a total order $>_t$. Using this strategy ensures that $>_t$ satisfies condition O1. We have that $\succ_t$ is a strict order because $\sigma_t$ is surjective except for $t'$ (an $\mathsf{a}$-term which, if $t' \in \Sigma_u$, can be obtained via $y\sigma_t$ or $t'\sigma_t$), but conditions (3) and (5) ensure that no two comparisons of terms in the neighbourhood mentioning $t'$ conflict with each other.

To show that $>_t$ will be monotonic, observe that due to condition (3) and the fact that $\succ_v$ is closed under canonic substitutions, $\succ_v$ already defines a total order on the $\mathsf{a}$-neighbourhood of $t$. Hence, for any totalisation of $\succ_t$ and any $\mathsf{a}$-terms in the neighbourhood $s_1, s_2$ with $s_1 >_t s_2$, we have $s_1 \succ_v s_2$. Thus, for any term $r$ in the $\mathsf{p}$-neighbourhood of $t$ and any position $p$, we have $r[s_1]_p \succ_v r[s_2]_p$ due to condition (4); by definition of $>_t$, this implies $t[s_1]_p >_t t[s_2]_p$. To show that $>_t$ has the subterm property, notice that condition (4) ensures that we have $s \succ_v s|_p$ for any term $s$ in the $\mathsf{p}$-neighbourhood, and hence by definition of $>_t$, $s >_t s|_p$. This shows that any totalisation $>_t$ of $\succ_t$ will be a simplification order. To complete the proof, we only need to show how to extend $\succ_t$ to a total order $>_t$ in a way that satisfies conditions O2 and O3. Thus, consider the possible forms of $t$:

- **(Case $t = c$)** In this case, for any term $s$ in the neighbourhood of $c$ there is a unique context term $s'$ such that $s'\sigma_c = s$. We refer to such $s'$ as the *lifted form* of $s$. Notice that lifted forms of elements in the neighbourhood of $c$ never contain variable $y$. To extend $\succ_c$ to a total order $>_c$, we start by noticing that the smallest element of $>_c$ must be $\mathsf{true}$ by condition (1). We then place the elements of $\Sigma_u$ in order, and then $c$, due to condition (2). Let $Q$ be the set of terms of the form $A\sigma_c$ in the neighbourhood of $t$ such that $\{A \approx \mathsf{true} \in \Delta_Q\}$. Next in $>_c$ we add the terms of $Q$, ensuring that before each term is added, all its $\mathsf{a}$-reductions by $\gtrdot$ are added to the total order too. With this, ordering $>_c$ clearly satisfies condition O2. Adding the terms of $Q$ to $>_c$ is possible because,

according to condition C2, the only order restrictions for terms in $Q$ that can be imposed by $\succ_v$, other than those already accounted for, are with other atoms of $Q$ (no lifted form of a term in the neighbourhood of $t$ mentions variable $y$); furthermore, according to condition (5), p-terms that are root terms can only be greater than their respective a-reductions w.r.t. $\gtrdot$. Thus, as long as $>_c$ respects the ordering between terms in $Q$ imposed by $\succ_v$, as well as the orderings between p-terms and their a-reductions w.r.t. $\gtrdot$, the terms in $Q$ and p-terms that are root terms can be ordered in any way. Finally, the remaining terms are totalised arbitrarily.

- **(Case $t = u$ for $u \in \Sigma_u$)** To define $>_u$, we totalise $\succ_u$ as follows. It is easy to see that the first element in $>_u$ is $\mathsf{true}$, followed by all elements of $\Sigma_u$ in the order defined by $\gtrdot$. Now let $Q$ be the set containing each term $A$ such that $A \approx \mathsf{true} \in \Delta_{\mathsf{Rt}}$. We next add to the ordering $>_u$ all elements of $Q$, as well as the a-reductions w.r.t. $\gtrdot$ of terms in $\Delta_{\mathsf{Rt}}$, respecting the ordering between terms in $Q$ imposed by $\succ_u$, as well as the orderings between p-terms and their a-reductions w.r.t. $\gtrdot$. This is possible because by condition (5), p-terms that are root terms can only be greater than their respective a-reductions w.r.t. $\gtrdot$. Thus, we ensure condition O3. To conclude, totalise the remainder of $\succ_t$ arbitrarily.

- **(Case $t = f(t')$)** We add $\mathsf{true}$ as the smallest element in $>_t$, followed by all elements of $\Sigma_u$, followed by $t'$ (unless $t'$ already in $\Sigma_u$), and then $t$. Now, let $Q$ be the set consisting of every term $s$ in the neighbourhood such that $s \approx \mathsf{true} \in \Delta_t$. We next add to $>_t$ all elements from $\Delta_t$, ensuring that for each such term added to the order, we have already added all its a-reductions by $\gtrdot$. This is possible because, by condition L3, for each term $s$ such that $s\sigma_t$ is in $Q$ we have that $s \in Z$, and condition (5) requires that p-terms for atoms in $Z$ can only be greater than their corresponding a-reductions. With this, we ensure condition O3. To conclude, totalise the remaining elements arbitrarily.

For the remainder of this section, we use the multiset extension of $>_t$ to define an order on ground literals and ground disjunctions as discussed in Section 2.2; to keep our notation simple, we use the same symbol $>_t$ for these orders.

#### 6.3.2.4 Grounding of clauses

Let us write the clauses in $N_t$ as $\{C^1, \ldots, C^n\}$. Observe that none of these clauses can have the empty disjunction as their head, according to condition L6. We represent

each clause $C^i$ as $\Gamma^i \to \Delta^i \vee L^i$, where $L^i >_t \Delta^i$; notice that this is possible because clause heads have no duplicate literals due to the fact that they are defined as sets. We also assume that the sequence is ordered in such a way that if $j > i$ then $L^j \vee \Delta^j >_t L^i \vee \Delta^i$. For these clauses, we have the following result:

**Lemma 5** (Lifting of clauses in $N_t$)**.** *Let $C^i = \Gamma^i \to \Delta^i \vee L^i$ be a clause of $N_t$ such that $L^i$ is not in* Rt *or an* a*-(in)equality between terms in $\{t, t'\} \cup \Sigma_u$. Then, there exits a clause $\Gamma \to \Delta_1 \vee \Delta_2$ in $\mathcal{S}_v$ such that*

$$\Gamma\sigma_t = \Gamma^i, \qquad \Delta_1\sigma_t \subseteq \Delta^i \vee \Delta_t, \qquad \Delta_2\sigma_t = L^i \quad \text{and} \quad \Delta_1 \not\succ_v \Delta_2.$$

*Proof.* By definition of $N_t$, there is a clause $\Gamma \to \Delta_1 \vee \Delta_2 \in \mathcal{S}_v$ such that $\Gamma\sigma_t = \Gamma^i$, $\Delta_1\sigma_t \subseteq \Delta^i \vee \Delta_t$, and $\Delta_2\sigma_t = L^i$. Furthermore, if $L \in \Delta_1$ is such that $L\sigma_t \in \Delta^i$, then $L \not\succ_v \Delta_2$, for otherwise condition O1 implies $\Delta^i >_t L^i$, which contradicts the definition of $L^i$. Thus, suppose $L \in \Delta_1$ with $L\sigma_t \in \Delta_t$, and suppose $L \succ_v L'$ for some $L' \in \Delta_2$. Due to the form of $\Delta_t$, we only have these possibilities:

- $L$ is of the form $S(u_1, u_2)$ or $B(u)$. But then, condition (5) of Definition 5 requires $L'$ must be a reduction of $L$, or an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u$. But this contradicts our assumption on the form of $L^i$.
- $L$ is an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u$. But then condition (5) of Definition 5 requires $L'$ must also be an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u$, which again contradicts the assumptions on $L^i$.
- $L$ is of the form $S(y, x)$, $S(x, y)$, or $B(y)$, in which case condition (5) of Definition 5 requires $L'$ can only be an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u$, which yields the same contradiction.

Thus, we conclude $L \not\succ_v \Delta_2$, and this completes the proof. □

We are interested in clarifying under which circumstances $\Delta_2$ contains more than one literal. The following lemma shows that this can only happen if $t = f(u)$ for some $f \in \Sigma_f$ and $u \in \Sigma_u$; furthermore, literals in $\Delta_2$ can only differ in replacements of term $y$ by $u$ and vice versa.

**Lemma 6.** *Consider two distinct* a*-terms $s_1, s_2$ appearing in $v$ such that $s_1 \neq s_2$ and $s_1\sigma_t = s_2\sigma_t$, where $s_1\sigma_t$ is ground. Then $t = f(u)$ for some $u \in \Sigma_u$ and $f \in \Sigma_f$; furthermore, $\{s_1, s_2\} = \{y, u\}$.*

*Proof.* We prove this by considering all possible forms of $s_1$ and $s_2$:

- If $s_1 = x$, then $x\sigma_t = t$; but then $s_2$ cannot be $y$ or $f(x)$, since their ground forms by $\sigma_t$ are $t'$ and $f(t)$. If $s_2 = u$ for $u \in \Sigma_u$, by hypothesis we have that $u = t$. However, since $x$ is a context term, $v \neq v_r$, so $t \notin \Sigma_u$, and we reach a contradiction. The fact that $v \neq v_r$ also implies that $s_2$ cannot be of the form $f(u)$ for some $u \in \Sigma_u$, $f \in \Sigma_u$. There are no options left for $s_2$, so this case is not possible.

- If $s_1 = f(x)$, then $f(x)\sigma_t = f(t)$, so clearly $s_2$ cannot be $y$, $x$ or in $\Sigma_u$. Furthermore, since $x$ is a context term, $v \neq v_r$, so $s_2$ cannot be of the form $f(u)$, and hence this case is not possible.

- If $s_1 = y$, then $y\sigma_t = t'$, so $s_2$ cannot be $x$ or $f(x)$. It also cannot be of the form $f(u)$, for this would imply $v = v_r$, but $v_r$ is only selected if $t \in \Sigma_u$, and here $t \notin \Sigma_u$ because it is of the form $f(t')$ for some $f \in \Sigma_f$. Thus, it can only be the case that $s_2 \in \Sigma_u$; however, the fact that $s_1\sigma_t = s_2\sigma_t$ ensures $t' \in \Sigma_u$ and $s_2 = t'$.

- If $s_1 = u$, then $u\sigma_t = u$; using an argument analogous to the previous case, we conclude that $s_2$ cannot be $x$, $f(x)$, $f(u)$, or in $\Sigma_u$, so it can only be the case that $s_2 = y$ and $t' = u$.

□

In the remainder of the proof, we often argue as follows: given a clause $C^i \in N_t$, identify a clause $C \in \mathcal{S}_v$ such that $\pi_t(C\sigma_t) = C^i$. Then, use that $\mathcal{S}_v$ is closed by application of inference rules to conclude that there is some clause $C' \,\hat{\in}\, \mathcal{S}_v$ such that $C'' = \pi_t(C'\sigma_t)$ is a "reduction" (in the standard sense of resolution-based theorem proving) of $C^i$ and is also satisfied by $R_t^*$. Variants of this reductive argument are commonly used in resolution-based reasoning [15]. Typically, one first proves that $R_t^*$ satisfies clauses in $N_t$ and then extends the result to clauses contained up to redundancy in $N_t$ using some version of Lemma 8. This suffices for our purposes since, as Lemma 7 proves, if $C' \,\hat{\in}\, \mathcal{S}_v$, then $C'' \,\hat{\in}\, N_t$.

**Lemma 7.** *For any clause $\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v$ such that $\Gamma\sigma_t \subseteq \Gamma_t$ and $\Gamma\sigma_t \to \Delta\sigma_t$ is ground, we have that $\Gamma\sigma_t \to \Delta\sigma_t \,\hat{\in}\, N_t$.*

*Proof.* If clause $\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v$ then we have exactly three possibilities:

- There is an equality $l \approx l \in \Delta$, and hence there is a literal $l\sigma_t \approx l\sigma_t$ in $\Delta\sigma_t$, which implies $\Gamma\sigma_t \to \Delta\sigma_t \,\hat{\in}\, N_t$.

- The literals $l \approx r$ and $l \not\approx r$ are in $\Delta$, so we have literals $l\sigma_t \approx r\sigma_t$ and $l\sigma_t \not\approx r\sigma_t$ in $\Delta\sigma_t$, and therefore $\Gamma\sigma_t \to \Delta\sigma_t \,\hat{\in}\, N_t$.
- There exist $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that $\Gamma' \to \Delta' \in \mathcal{S}_v$. Since $\Gamma'\sigma_t \to \Delta'\sigma_t$ is ground and $\Gamma'\sigma_t \subseteq \Gamma_t$, by definition of $N_t$ we have that $\Gamma'\sigma_t \to \pi_t(\Delta'\sigma_t) \in N_t$, which implies $\Gamma\sigma_t \to \Delta\sigma_t \,\hat{\in}\, N_t$.

□

**Lemma 8.** *Let $\mathcal{I}$ be an arbitrary interpretation of $\mathcal{O}$. Let $C = \Gamma \to \Delta$ be a clause contained up to redundancy in $N_t$. Suppose that either: (i) $\mathcal{I} \models \Delta^j \vee L^j$ for each $j$ with $1 \leq j \leq n$, or (ii) there is some $i$ with $1 \leq i < n$ such that $\mathcal{I} \models \Delta^j \vee L^j$ for each $1 \leq j \leq i$, and $L^{i+1} \vee \Delta^{i+1} >_t \Delta$. Then, $\mathcal{I} \models \Delta$.*

*Proof.* Suppose $\mathcal{I} \not\models \Delta$ for the sake of contradiction. Since $\mathcal{I} \not\models \Delta$, we have that $\Delta$ cannot contain a tautology, so $C \,\hat{\in}\, N_t$ implies that there is some $k$ with $1 \leq k \leq n$ such that $\Gamma^k \subseteq \Gamma$ and $\Delta^k \vee L^k \subseteq \Delta$. If $\mathcal{I} \models \Delta^j \vee L^j$ for each $1 \leq j \leq n$, we immediately obtain $\mathcal{I} \models \Delta$, which contradicts our assumption. Thus, let $i$ with $1 \leq i < n$ be such that $\mathcal{I} \models \Delta^j \vee L^j$ for each $1 \leq j \leq i$, and $L^{i+1} \vee \Delta^{i+1} >_t \Delta$. If $k \leq i$, we obtain the same contradiction as above, so assume $k > i$. The fact that $\Delta^k \vee L^k \subseteq \Delta$ implies $\Delta \geq_t L^k \vee \Delta^k$; however, since $k \geq i + 1$ we have $L^k \vee \Delta^k \geq_t L^{i+1} \vee \Delta^{i+1} >_t \Delta$, which contradicts the previous claim. □

We conclude this section by proving an auxiliary lemma for the next section. The proof is an example of the type of reductive argument discussed above.

**Lemma 9.** *If there is a clause $\Gamma \to \Delta \vee l \not\approx l \in N_t$, then $\Gamma \to \Delta \,\hat{\in}\, N_t$.*

*Proof.* Suppose there is a clause $\Gamma \to \Delta \vee l \not\approx l \in N_t$. By definition of $N_t$, there is a clause $\Gamma' \to \Delta'_1 \vee \Delta'_2 \in \mathcal{S}_v$ with $\Gamma'\sigma_t = \Gamma$, $\Delta'_1\sigma_t \subseteq \Delta \cup \Delta_t$ and $\Delta'_2\sigma_t = l \not\approx l$. If $t \neq u$ for every $u \in \Sigma_u$, Lemma 6 ensures that $\Delta'_2$ is of the form $l' \not\approx l'$ with $l'\sigma_t = l$. If $t = u$ for some $u \in \Sigma_u$, then $l \neq u$, for $u \not\approx u \in \Delta_{\mathsf{Rt}}$ by condition L1 and $\Delta_{\mathsf{Rt}} \subseteq \Delta_t$ by condition L4, but Lemma 6 then ensures that $\Delta'_2$ is of the form $l' \not\approx l'$ with $l'\sigma_t = l$. Since rule $\mathsf{Ineq}$ is not applicable to $\Gamma' \to \Delta'_1 \vee l' \not\approx l'$, we have that $\Gamma' \to \Delta'_1 \,\hat{\in}\, \mathcal{S}_v$; by Lemma 7, this implies $\Gamma \to \Delta \,\hat{\in}\, N_t$. □

#### 6.3.2.5 Construction of the rewrite system $R_t$

Let $n$ be the number of clauses in $N_t$. Consider the following sequence of monotonically growing rewrite systems $\{R_t^0, \ldots, R_t^n\}$ defined inductively as follows:

- $R_t^0 := \emptyset$
- $R_t^i = R_t^{i-1} \cup \{l^i \Rightarrow r^i\}$ if $L^i$ is of the form $l^i \approx r^i$ such that:

  R1. $(R_t^{i-1})^* \not\models \Delta^i \vee L^i$,

  R2. $l^i >_t r^i$,

  R3. $l^i$ is irreducible by $R^{i-1}$, and

  R4. $(R^{i-1})^* \not\models s \approx r^i$ for each $l^i \approx s \in \Delta^i$.

- $R^i = R_t^{i-1}$ in all other cases.

Let $R_t = R_t^n$. If a clause verifies conditions R1 through R4, we call it a *generative* clause, and $\{l^i \Rightarrow r^i\}$ is the *generated* rewrite rule in $R_t$. The model induced by $R_t$ will be represented as $R_t^*$. Before showing that $R_t^*$ satisfies all clauses in $N_t$ and that it is a counterexample for the relevant query $\Gamma_t \to \Delta_t$, we prove some properties of the series of the rewrite systems $R_t^i$. Lemmas 10 to 12 follow simply from the definition of $R_t^*$:

**Lemma 10** (Church-Rosser property)**.** *Each rewrite system $R_t^i$ for $1 \leq i \leq n$ is Church-Rosser. In particular, $R_t$ is Church-Rosser.*

*Proof.* Let $i$ be an arbitrary number in $1 \leq i \leq n$. To show that the rewrite system $R_t^i$ is terminating, notice that each rule of $R_t^i$ is of the form $l \Rightarrow r$ with $l >_t r$ by condition R2. Hence, all rules are embedded in $>_t$, and since $>_t$ is a simplification order, $R_t^i$ is terminating by Lemma 1. To show that the rewrite system is left-reduced, let $l \Rightarrow r$ be an arbitrary rewrite rule in the system, and let us show that no rewrite rule applies to $l$. Let $C^i$ be the clause that generates $l \Rightarrow r$, and suppose that there is some $1 \leq j < i$ such that $C^j$ is generative and $l$ is reduced by $l^j \Rightarrow r^j$ at position $p$. However, since $l^j \Rightarrow r^j \in R_t^{i-1}$, this contradicts condition R3. Now suppose that there is some $k > i$ such that $l$ is reduced by the rule $l^k \Rightarrow r^k$ generated by $C^k$. However, since $k > i$, we have $l^k \approx r^k \geq_t l \approx r$, which implies $l^k \geq_t l$. But if $l^k = l$, then the fact that $C^k$ is generative violates condition R3, and if $l^k >_t l$, then $l^k$ cannot reduce $l$, since we have shown that rewrite rules are embedded in $>_t$, and due to the subterm property of $>_t$, no subterm $l|_p$ of $l$ can be such that $l|_p >_t l^k$. □

**Lemma 11** (Inequality Monotonicity)**.** *For each* $1 \leq i \leq n$ *and every* $l \not\approx r \in \Delta^i \vee L^i$ *we have that* $(R_t^i)^* \models l \not\approx r$ *if and only if* $R_t^* \models l \not\approx r$*.*

*Proof.* If $(R_t^i)^* \not\models l \not\approx r$, then $l \approx r \in (R_t^i)^*$, and since $(R_t^i)^* \subseteq R_t^*$, then $l \approx r \in R_t^*$, so $R_t^* \not\models l \not\approx r$.

Suppose $(R_t^i)^* \models l \not\approx r$, which is equivalent to $(R_t^i)^* \not\models l \approx r$. Let $j$ be the smallest element $i < j \leq n$ such that $(R_t^{j-1})^* \models l \approx r$. We assume that $C^j$ is generative, since otherwise $(R_t^{j-1})^*$ also satisfies $l \approx r$, and thus we contradict the assumption that $j$ is minimal. Let $L^j = l^j \approx r^j$ and $L^i = l^i \bowtie r^i$, where $L^j$ and $L^i$ are the maximum literals of $C^j$ and $C^i$, respectively. Since $j > i$, we have $l^j \approx r^j \geq_t l^i \bowtie r^i \geq_t l \not\approx r$. If $\bowtie$ is an inequality, then $l^j >_t l^i$ and hence $l^j >_t l$; if $\bowtie$ is an equality, then $l^i >_t l$, and thus $l^j >_t l$. A symmetric argument applies to $r$. Since both $R_t^{j-1}$ and $R_t^j$ are Church-Rosser by Lemma 10, Lemma 4 implies $(R_t^j)^* \not\models l \approx r$, which contradicts our supposition that $(R_t^{j-1})^* \models l \approx r$ since $(R_t^{j-1})^* \subseteq (R_t^j)^*$. Therefore, no such $j$ exists, so $(R_t^n)^* \not\models l \approx r$. □

**Lemma 12** (Canonicity)**.** *For any* $1 \leq i \leq n$*, if* $C^i$ *is generative, we have* $R_t^* \not\models \Delta^i$*.*

*Proof.* Let $L$ be a literal in $\Delta^i$. If $L$ is an inequality, then $(R_t^{i-1})^* \not\models L$ implies $R_t^*$ by Lemma 11. If $L$ is an equality, let $L$ be written as $l \approx r$ with $l >_t r$ (if $l = r$, then $C^i$ is trivially satisfied by $(R_t^{i-1})^*$, so it cannot be generative). Let $j$ be the smallest integer $i \leq j \leq n$ such that $(R_t^j)^* \models l \approx r$, which implies $(R_t^{j-1})^* \not\models l \approx r$ and also that $C^j$ is generative. Notice that $l^j \approx r^j \geq_t l^i \approx r^i >_t l \approx r$. Thus, $l^j \geq_t l$ and $l^j >_t r$. If $l^j > l$, then since $R_t^{j-1}$ and $R_t^j$ are Church-Rosser by Lemma 10, Lemma 4 implies $(R_t^j)^* \models l \approx r$, which contradicts our assumption. If $l^j = l$, then $l^i = l$, and $r^i \neq r$, for otherwise $R_t^*$ satisfies $l \approx r$ trivially since $(R_t^i)^* \models l \approx r$ by construction. Since $C^i$ is generative, by condition R4 we have $(R_t^{j-1})^* \not\models r \approx r^i$. Since $l >_t r^i$ and $l >_t r$, Lemma 4 entails $(R_t^j)^* \not\models r \approx r^i$, which contradicts the assumption $(R_t^j)^* \models l \approx r$ and the fact that $(R_t^i)^* \subseteq (R_t^j)^* \models l \approx r^i$, because $\approx$ is a congruence. Hence, no such $j$ exists, so $R_t^* \not\models l \approx r$. □

#### 6.3.2.6 Fragment Adequacy

The next part of the proof consists in showing that $R_t^*$ is a model of the clauses $N_t$. We start by proving preliminary results in Lemmas 13 to 15.

**Lemma 13.** *For any generative clause* $C^i$*, if* $L \in \Delta^i$ *then* $L$ *is neither of the form* $s \approx s$ *nor of the form* $s \not\approx s$*.*

*Proof.* If $L$ is of the form $s \approx s$, then the literal is trivially satisfied and the clause cannot be generative because condition R1 is not satisfied. Thus, let $i$ be the minimal integer such that $C^i$ contains a literal of the form $s \not\approx s$. Let us write $C^i$ as $\Gamma^i \to \Delta^i \vee s \not\approx s$. By an argument analogous to the proof of Lemma 9, there exists a clause of the form $\Gamma' \to \Delta' \vee l' \not\approx l'$ in $\mathcal{S}_v$. Since $\mathsf{Ineq}$ is not applicable, we have $\Gamma' \to \Delta' \,\hat{\in}\, \mathcal{S}_v$. Furthermore, the fact that $\mathsf{Elim}$ is not applicable to $\Gamma' \to \Delta' \vee l' \not\approx l' \in \mathcal{S}_v$ implies that $\Delta'$ contains no tautology of the form $\{s_1 \approx s_1\}$ or $\{s_1 \approx s_2 \vee s_1 \not\approx s_2\}$, and therefore there exist $\Gamma'' \subseteq \Gamma'$ and $\Delta'' \subseteq \Delta'$ such that $\Gamma'' \to \Delta'' \in \mathcal{S}_v$. But this contradicts the fact that $\mathsf{Elim}$ is not applicable to $\Gamma' \to \Delta' \vee l' \not\approx l'$.

□

**Lemma 14** (Root Equality Invariance)**.** *If $t \neq c$ and $u_1, u_2$ are elements of $\Sigma_u$ such that $u_1 \approx u_2 \in (R_t^i)^*$ for some $1 \leq i \leq n$, then $u_1 \approx u_2 \in R^*_{\mathsf{Rt}}$.*

*Proof.* We use proof by contradiction. Let $i$ be the smallest integer for which the lemma is false, so there exist $u_1$ and $u_2$ with $u_1, u_2 \in \Sigma_u$ such that $(R_t^i)^* \models u_1 \approx u_2$, but $u_1 \approx u_2 \notin R^*_{\mathsf{Rt}}$. Notice that if $u_1 = u_2$, we immediately obtain a contradiction, so we assume $u_1 >_t u_2$. Since $i$ is the smallest integer that makes the lemma false, we have that $i$ cannot be 0 since $(R_t^0)^* \not\models u_1 \approx u_2$. We also have that $(R_t^{i-1})^* \not\models u_1 \approx u_2$ and $C^i$ is generative. Let $l^i \approx r^i$ be $L^i$. If $l^i \approx r^i \in R^*_{\mathsf{Rt}}$ then by induction hypothesis and the fact that by definition of $>_t$, $(R_t^i)^*$ can only contain equalities between elements of $\Sigma_u$, then $(R_t^i)^* \subseteq R^*_{\mathsf{Rt}}$; but then $(R_t^i)^* \models u_1 \approx u_2$ implies $R^*_{\mathsf{Rt}} \models u_1 \approx u_2$, which contradicts our hypothesis. Therefore, $l^i \approx r^i \notin R^*_{\mathsf{Rt}}$; by condition L1, we have $l^i \approx r^i \in \Delta_{\mathsf{Rt}}$, and since $t \neq c$, by condition L4, we have $l^i \approx r^i \in \Delta_t$, which contradicts the fact that $l^i \approx r^i$ appears in $\Delta^i \vee L^i$.

□

**Lemma 15** (Irreducibility of $t$ and $t'$)**.** *If $t \neq c$, terms $t$ and $t'$ (if it exists) are irreducible by $R_t$.*

*Proof.* By definition of $>_t$, we have that if $t'$ is reducible, there exists a generative clause $C^i$ of the form $\Gamma^i \to \Delta^i \vee t' \approx u$ for some $u \in \Sigma_u$ and $t' >_t u$. If $t' \in \Sigma_u$, then $t' >_t u$ implies $t' \succcurlyeq u$, and by condition L5 we have $t' \approx u \in \Delta_t$, so we reach a contradiction. If $t' \notin \Sigma_u$, then condition L5 implies $t' \approx u \in \Delta_t$ and we again obtain a contradiction. Thus, $t'$ is irreducible.

Suppose $t$ can be reduced. Then, there exists a generative clause $C^i$ of the form $\Gamma^i \to \Delta^i \vee t \approx u$ or $\Gamma^i \to \Delta^i \vee t \approx t'$. However, condition L5 ensures $t \approx u \in \Delta_t$ and $t \approx t' \in \Delta_t$, so we again reach a contradiction.

□

We now show that the interpretation defined above satisfies all clauses of $N_t$. We do this by showing that $R_t^*$ satisfies the *head* of each clause in $N_t$. For the remainder of this sub-section, we assume that, unless otherwise noted, if $t = c$, then $c$ is irreducible w.r.t. $R_c$. This is due to the fact that during construction of the model, if $c$ is not irreducible w.r.t. $R_c$, we will not use this fragment.

**Lemma 16** (Fragment Adequacy)**.** *For each* $1 \leq i \leq n$*, we have* $R_t^* \models \Delta^i \vee L^i$*.*

*Proof.* We proceed using proof by contradiction. Let $i$ be the smallest number between 1 and $n$ such that $R_t^* \not\models \Delta^i \vee L^i$. We assume that $C^i$ is not generative, because all generative clauses are trivially satisfied by $R_t^*$. To complete the proof, we consider all possible forms of $L^i$:

- Case $L^i = l^i \approx l^i$. But then we have that $R_t^* \models l^i \approx l^i$, which contradicts the main hypothesis.
- Case $L^i = l^i \approx r^i$, with $l^i >_t r^i$. We reach a contradiction by showing that conditions R1 through R4 are verified, and therefore $C^i$ should be generative.
  - Condition R1. Suppose this condition is not verified. Then, there is some literal $L \in \Delta^i \vee L^i$ such that $(R_t^{i-1})^* \models L$. But if $L$ is an equality, then $R_t^* \models L$ trivially, and if $L$ is an inequality, then since $C^i$ is not generative we have $(R_t^i)^* \models L$, and then by Lemma 11 we have $R_t^* \models L$, which contradicts the main hypothesis.
  - Condition R2 is trivially satisfied in this case.
  - Condition R3. Suppose $l^i$ can be reduced by $R_t^{i-1}$. Let $l^j \Rightarrow r^j$ be the rule in $R_t^{i-1}$ which reduces $l^i$ at (one of) the deepest position(s), which we name $p$. Let $C^j$ be the generative clause for rule $l^j \Rightarrow r^j$.

    **Lifting** $C^j$ If $t \neq c$, notice that $l^j$ is neither $t$ nor $t'$, since these terms are irreducible If $l^j \in \Sigma_u$, then by Lemma 14, $R_{\mathsf{Rt}}^* \models l^j \approx r^j$. Hence, there exists some $r_j \in \Sigma_u$ such that $l^j \approx r_j \in \Gamma_\Omega$. Hence, by condition L7, we have $\top \rightarrow l^j \approx r_j \in \mathcal{S}_v$ (without loss of generality; the body could be $l^j \approx r_j$ and the treatment would be identical), which follows from the fact that any clause subsuming it will be in $N_t$ because $\Gamma_\Omega \subseteq \Gamma_t$ due to condition L4, and hence the head of this clause is not empty due to condition L6. In this case, we define $\Gamma' = l^j \approx r_j$ and $\Delta' = \bot$; notice that this is a clause of the form (5) below, with the same properties. Otherwise,

since $l^j$ and $r^j$ are $\mathsf{a}$-terms, $l^j$ must be of the form $f(t)$ for some $f \in \Sigma_f$. If $r^j \neq t' \in \Sigma_u$, then Lemmas 5 and 6 ensure that there exists a clause

$$\Gamma' \to \Delta' \vee l' \approx r' \in \mathcal{S}_v, \tag{5}$$

with $\Gamma'\sigma_t = \Gamma^j$, $\Delta'\sigma_t \subseteq \Delta^j \cup \Delta_t$, $l'\sigma_t = l^j$ and $r'\sigma_t = r^j$ with $\Delta' \not\succ_v l' \approx r'$. We now prove that such a clause can also be found when $r^j = t' \in \Sigma_u$. Indeed, suppose there is no clause of the form (5). Then, Lemmas 5 and 6 imply that there is a clause $\Gamma' \to \Delta' \vee f(x) \approx y \vee f(x) \approx u \in \mathcal{S}_v$ with $\Gamma'\sigma_t = \Gamma^j$, $\Delta'\sigma_t \subseteq \Delta^j \cup \Delta_t$, $f(x)\sigma_t = l^j$; furthermore, $\Delta' \not\succ_v f(x) \approx y \vee f(x) \approx u$. But since $\mathsf{Factor}$ is not applicable, then there exists a clause $\Gamma' \to \Delta' \vee y \not\approx u \vee f(x) \approx u \,\hat{\in}\, \mathcal{S}_v$ (without loss of generality; we could replace $u$ by $y$ in the last literal). Notice that this clause cannot contain a tautology, since if $y \approx u \in \Delta'$, then $u \approx u \in \Delta^j$, so $C^j$ is trivially satisfied by $(R_t^{j-1})^*$ and hence is not generative. Hence, there is a clause $C'$ in $\mathcal{S}_v$ that subsumes $\Gamma' \to \Delta' \vee y \not\approx u \vee f(x) \approx u$; but such clause must contain $f(x) \approx u$, for otherwise the grounding by $\sigma_t$ of $C'$ is in $N_t$ and subsumes $C^j$, and hence it is a smaller clause in $N_t$, so by induction this clause is satisfied by $(R_t^{j-1})^*$, and hence $C^j$ is not generative. But then, the existence of $C'$ contradicts the assumption that there is no clause of the form (5). Hence, the existence of a clause of the form (5) is guaranteed.

If $t = c$ and $c$ is irreducible, then $l^j$ cannot be $t'$ since this is not defined. If $l^j = u$, then $r^j = u'$ for some $u' \in \Sigma_u$ with $u \succcurlyeq u'$, and there exists a clause $\Gamma' \to \Delta' \vee u \approx u'$ in $\mathcal{S}_v$ with $\Gamma'\sigma_c = \Gamma^j$, $\Delta'\sigma_c \subseteq \Delta^j \vee \Delta_c$. Notice that if $L \in \Delta'$ is such that $L\sigma_c \in \Delta^j$ then $L \not\succ_q u \approx u'$ due to $u \approx u' >_t \Delta_t$; otherwise $L \in \Delta_Q$, and then by condition C2 we have $L \not\succ_q u$ and hence $L \not\succ_q u \approx u'$. This is our clause of the form (5). Finally, if $l^j$ is of the form $f(t)$, we proceed as in the previous paragraph.

**Lifting $C^i$** If $t \neq c$, then $l^i$ cannot be $t$ or $t'$ either, due to Lemma 15. If $l^i \in \Sigma_u$, then by definition of $>_t$, we have $r^i \in \Sigma_u$ too. But then, since $l^i \approx r^i$ appears in $C^i$, we have $l^i \approx r^i \notin \Delta_{\mathsf{Rt}}$, and hence by condition L1 we have $l^i \approx r^i \in R^*_{\mathsf{Rt}}$. As a result, there exist equalities $l^i \approx r_1, \ldots, l_K \approx r_K$ in $\Gamma_\Omega$ which imply $l^i \approx r^i$. By an argument analogous to the case above, we have that for each $k$ with $1 \leq k \leq K$, we have (without loss of generality) a clause $C^k = \top \to l_k \approx r_k$ in $N_t$. Furthermore, due to the fact that $l^i$ is in $\Sigma_u$, we have $l^j = l^i$ and since $i > j$ then $r^i >_t r^j$, $r^i >_t r_1$, $r_1 \geq_t l_k$,

and $r^j \geq_t l_k$, so all clauses $C^k$ are smaller than $C^i$. Thus, by induction hypothesis, $(R_t^{i-1})^*$ satisfies such equalities, and hence it satisfies $l^i \approx r^i$. If $l^i \in \Omega$, the argument is analogous. If $l^i$ is of the form $f(t)$, we proceed as in the previous case and obtain a clause

$$\Gamma \to \Delta \vee l \approx r \in \mathcal{S}_v \tag{6}$$

with $\Gamma\sigma_t = \Gamma^i$, $\Delta'\sigma_t \subseteq \Delta^i \cup \Delta_t$, $l\sigma_t = l^i$ and $r\sigma_t = r^i$. If $l^i$ is a $\mathsf{p}$-term not in $\Omega$, notice that due to the form of $l^j$, $l^i$ cannot contain $t'$, so Lemma 6 guarantees the existence of a clause of the form (6). In both cases, we are in the conditions of Lemma 5, so $\Delta \not\succ_v l \approx r$.

If $t = c$ and $c$ is irreducible, then if $l^i \in \Sigma_u$, then $r^i \in \Sigma_u$ and we argue as in the case of $C^j$ to the same conclusion. Finally, if $l^i$ is of a different form, the argument is identical to that for the case $t \neq c$.

**Saturation by $\mathsf{Eq}$** Observe that due to the form and properties of (5) and (6), we are in the conditions of the $\mathsf{Eq}$ rule, which ensures that there is a clause $\Gamma \wedge \Gamma' \to \Delta \vee \Delta' \vee l[r']_p \approx r \,\hat{\in}\, \mathcal{S}_v$; Lemma 7 then ensures that $\Gamma\sigma_t \wedge \Gamma'\sigma_t \to \Delta^i \vee \Delta^j \vee l^i[r^j]_p \approx r^i \,\hat{\in}\, N_t$; notice that the head of this clause is smaller than $\Delta^i \vee l^i \approx r^i$ due to the monotonicity property of the ground order. Thus, by Lemma 8, we conclude $R_t^* \models \Delta^i \vee \Delta^j \vee l^i[r^j]_p \approx r^i$. However, the fact that $C^j$ is generative, together with Lemma 12 implies $R_t^* \not\models \Delta^j$; similarly, by hypothesis we have $R_t^* \not\models \Delta^i$. Finally, since $R_t^* \models l^j \approx r^j$ but $R_t^* \not\models l^i \approx r^i$, and $R_t^*$ is a congruence, we have that $R_t^* \not\models l^i[r^j]_p \approx r^i$. We have thus obtained a contradiction, and hence condition R3 must be satisfied.

– Condition R4. Suppose that there exists $l^i \approx s^i \in \Delta^i$, with $r^i >_t s^i$ and $(R_t^{i-1})^* \models r^i \approx s^i$. Notice that in this case, $l^i$, $r^i$, and $s^i$ are $\mathsf{a}$-terms, since if they were $\mathsf{p}$-terms, we would have $r^i = s^i = \mathsf{true}$.

**Lifting $C^i$** If $t \neq c$, notice that $l^i$ cannot be $t$ or $t'$ by Lemma 15, and it cannot be in $\Sigma_u$ for otherwise we could argue as in the previous case to show $R_t^* \models l^i \approx r^i$. Thus, $l^i$ is of the form $f(t)$, and we argue as above to show that there is a clause of the form:

$$\Gamma \to \Delta \vee \Delta' \vee l \approx r \in \mathcal{S}_v \tag{7}$$

with $\Gamma\sigma_t = \Gamma^i$, $\Delta\sigma_t \subseteq (\Delta^i \backslash \{l^i \approx s^i\}) \cup \Delta_t$, $\Delta'\sigma_t = l^i \approx s^i$, $l\sigma_t = l^i$ and $r\sigma_t = r^i$, and since we are in the conditions of Lemma 5, $\Delta \vee \Delta' \not\succ_v l \approx r$.

Furthermore, $\Delta'$ can contain at most two literals $l \approx s_1$, $l \approx s_2$ such that $l\sigma_t = l^i$, $s_1\sigma_t = s^i$, and $s_2\sigma_t = s^i$, since due to the form of $l^i$, either $t' \notin \Sigma_u$ or $s^i \neq t' \in \Sigma_u$; in both cases Lemma 6 ensures that there is only one such literal, name it $l \approx s_1$, or $t' = u \in \Sigma_u$ and (w.l.o.g.) $s_1 = y$ and $s_2 = u$.

If $t = c$ and $c$ is irreducible, then we already argued that $l^i$ cannot be $t'$; hence $l^i = u \in \Sigma_u$ and $r^i = u' \in \Sigma_u$ or $l^i$ is of the form $f(t)$. In all these cases, we have shown that there exists a clause of the form (7) with $\Delta \vee \Delta' \not\succ_v l \approx r$.

**Saturation by** Factor Since the Factor rule is not applicable, we have $\Gamma \to \Delta'' \vee s_1 \not\approx r \vee l \approx r \,\hat{\in}\, \mathcal{S}_v$, where $\Delta'' = \Delta \vee \{l \approx s_1\}$ if $l \approx s_2 \in \Delta'$, and otherwise $\Delta'' = \Delta$. Notice that this clause does not contain a tautology, since that would imply $s^i \approx r^i \in \Delta^i$ and hence $C^i$ would be satisfied, contrarily to our main assumption. Thus, there exists a clause $C'$ in $\mathcal{S}_v$ which subsumes this one. Notice that the head of this clause must contain $l \approx r$, for otherwise the grounding of $C'$ by $\sigma_t$ is a smaller clause in $N_t$, which must be satisfied by $R_t^*$ by induction hypothesis. But then, since we have $R_t^* \models s^i \approx r^i$ and $R_t^* \not\models l^i \approx r^i$, we cannot have $R_t^* \models l^i \approx s^i$, so it must be the case that $R_t^*$ satisfies some literal of $\pi_t(\Delta\sigma_t)$, and this contradicts $R_t^* \not\models C^i$. If the head of $C'$ does not contain $l \approx s_2$, then we have that the grounding of $C'$ is in $N_t$ and since $l^i \approx s^i >_t r^i \not\approx s^i$, the head of such clause is smaller than the head of $C^i$, so by induction hypothesis we have $R_t^* \models \pi_t(\Delta\sigma_t)$; however, $\pi_t(\Delta\sigma_t) \subseteq \Delta^i \vee s^i \not\approx r^i$, and we have that $R_t^* \not\models \Delta^i$ (by hypothesis) and $R_t^* \not\models s^i \not\approx r^i$ since $R_t^* \models s^i \approx r^i$ by hypothesis; thus, this leads to a contradiction. Finally, if the head of $C'$ contains $l \approx s_2$, we have that $C'$ is a clause of the form (7) where $\Delta'$ contains only one literal. We apply the same argument via saturation of the Factor rule to this clause, and we obtain a contradiction as in the case above.

- Case $L^i = l^i \not\approx l^i$. In this case, we have that since $u \not\approx u \in \Delta_{\mathsf{Rt}}$ for every $u \in \Sigma_u$, we have $l^i \notin \Sigma_u$. We are in the conditions of Lemma 6, and hence there exists a clause $\Gamma \to \Delta \vee l \not\approx l$ where $\Gamma\sigma_t = \Gamma^i$, $\Delta\sigma_t \subseteq \Delta^i \vee \Delta_t$, and $l\sigma_t = l^i$. Furthermore, we are in the conditions of Lemma 5, so $\Delta \not\succ_v l \not\approx l$. By Lemma 9, we have $\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v$, and since $L_i \vee \Delta_i >_t \Delta_i$, by Lemma 7 and Lemma 8, we have $R_t^* \models \Delta_i$, which contradicts our hypothesis.

- Case $L^i = l^i \not\approx r^i$ with $l^i >_t r^i$. By Lemma 11, we have that $(R_t^{i-1})^* \not\models l^i \not\approx r^i$, which means that $(R_t^{i-1})^* \models l^i \approx r^i$, and hence $l^i$ is reducible by $R_t^{i-1}$. The contradiction is then obtained as in the proof for condition R3 in the previous case; we merely replace $l^i \approx r^i$ by $l^i \not\approx r^i$. Furthermore, we disprove the case where $l^i \in \Sigma_u$ since by definition of $>_t$, we have $r^i \in \Sigma_u$ too, and then since $l^i \not\approx r^i$ appears in $C^i$, we have $l^i \approx r^i \in \Delta_{\mathsf{Rt}}$, and hence $l^i \approx r^i \notin R^*_{\mathsf{Rt}}$ by condition L1. But then the fact that $R_t^* \models l^i \approx r^i$ contradicts Lemma 14.

□

**Corollary 17.** *For any clause* $\Gamma \to \Delta \,\hat{\in}\, N_t$*, we have* $R_t^* \models \Delta$.

*Proof.* The result follows trivially from Lemma 8 given that choosing $i = n$ satisfies the conditions of the theorem due to Lemma 16.

□

#### 6.3.2.7 Query Refutation

We now know $R_t^*$ is a model of $N_t$ (and all clauses subsumed by $N_t$). To conclude the proof, we only need to show that $R_t^*$ is a valid counterexample for the relevant query, that is, $R_t^* \not\models \Gamma_t \to \Delta_t$. We start by proving this for the relevant atoms in $\mathsf{Rt}$.

**Lemma 18.** $R_t^* \models R^*_{\mathsf{Rt}}$ *and* $R_t^* \not\models \Delta_{\mathsf{Rt}}$.

*Proof.* The lemma is trivially true in the case $t = c$, so we assume $t \neq c$ and then, by condition L4, we have $\Gamma_{\mathsf{Rt}} \subseteq \Gamma_t$; hence $R_t^* \models R^*_{\mathsf{Rt}}$ by condition L7 and Corollary 17. Consider $u_1 \approx u_2 \in \Delta_{\mathsf{Rt}}$, and suppose $R_t^* \models u_1 \approx u_2$. By Lemma 14, $R_t^* \models u_1 \approx u_2$ implies $R^*_{\mathsf{Rt}} \models u_1 \approx u_2$. However, by condition L1, $u_1 \approx u_2 \in \Delta_{\mathsf{Rt}}$ implies $R^*_{\mathsf{Rt}} \not\models u_1 \approx u_2$; hence we obtain a contradiction. Now, consider $u_1 \not\approx u_2 \in \Delta_{\mathsf{Rt}}$ such that $R_t^* \models u_1 \not\approx u_2$, that is, $R_t^* \not\models u_1 \approx u_2$. By condition L1, $u_1 \not\approx u_2 \in \Delta_{\mathsf{Rt}}$ implies $R^*_{\mathsf{Rt}} \models u_1 \approx u_2$, and hence $\Gamma_t \models u_1 \approx u_2$, which contradicts our previous claim. Consider a function-free, ground atom $A \in \Delta_{\mathsf{Rt}}$ and suppose $R_t^* \models A$. By condition L2, we have $R^*_{\mathsf{Rt}} \not\models \Delta_{\mathsf{Rt}}$, which implies $R_t^* \not\models A$, as seen above. This contradiction concludes the proof, as we have exhausted all cases for elements of $\Delta_{\mathsf{Rt}}$. □

For $t \neq c$ or $t = c$ with $c$ irreducible w.r.t. $R_c$, we can use the final results in the previous section to prove that $R_t^*$ refutes the query $\Gamma_t \to \Delta_t$ (see Lemma 21). However, we also need to prove refutation of the query also for the case when $t = c$ and $c$ is not irreducible w.r.t. $R_c$. To that effect, we prove first Lemmas 19 and 20.

**Lemma 19.** *In case that $t = c$ and $c$ is not irreducible w.r.t. $R_c$, we have $R_t^* \models \Gamma_t$.*

*Proof.* Suppose there is $A \in \Gamma_c$ such that $R_c^* \not\models A$. By definition of $\Gamma_c$, $A$ is of the form $A(x)$ for $A \in \Sigma_A$, and since $c$ is not irreducible w.r.t. $R_c$, there exists a generative clause $\Gamma^i \to \Delta^i \vee c \approx u$ for some $u \in \Sigma_u$. Since $C^i$ is generative, we have that $x$ is irreducible by $R_{i-1}^* t$, so we can apply an argument analogous to the proof of Lemma 16 in the case $t = c$ with $c$ irreducible to show that $(R_c^{i-1})^* \models C^k$ for each $1 \leq k \leq i-1$.

Now, if we consider $C^i$, we are in the conditions of Lemma 6, so there is a clause $\Gamma \to \Delta \vee x \approx u$ with $\Gamma\sigma_c = \Gamma^i$ and $\Delta\sigma_c \subseteq \Delta_i \cup \Delta_c$; observe now that if some $B \in \Delta$ is such that $A \succ_q x \approx u$, we have that $A\sigma_c \notin \Delta^i$ since $c \approx u >_c \Delta^i$, but then $B$ is of the form $B(x) \in \Delta_Q$, by definition of $\Delta_c$, and this contradicts condition (2) of Definition 5. Thus, $\Delta \not\succ_q x \approx u$.

Furthermore, by condition C1 of Theorem 2, we have a clause $\Gamma' \to A(x) \in \mathcal{S}_q$, with $\Gamma' \subseteq \Gamma_Q$, and hence its $\mathsf{a}$-terms only $x$, so the preconditions of the $\mathsf{Eq}$ rule are satisfied, and since this clause is not applicable, we have $\Gamma \wedge \Gamma' \to \Delta \vee A(u) \,\hat{\in}\, \mathcal{S}_v$. If $\Delta$ contains a tautology, then we contradict the fact that $C^i$ is generative due to condition R1, so there is a clause $\Gamma^j \to \Delta^j \vee L^j \in N_c$ with $\Gamma^j \subseteq \Gamma\sigma_c \wedge \Gamma'_c$, $\Delta^j \subseteq \Delta\sigma_c \vee A(u)$, and in particular we have $A(u) \in \Delta^j$, for otherwise the fact that this clause is smaller than $C^i$ implies, by Lemma 8, $(R_c^{i-1})^* \models \Delta^j$, contradicting the fact that $C^i$ is generative.

However, the fact that this clause is smaller than $C^i$, together with $R_c^* \not\models \Delta^i$ by Lemma 12 implies $(R_c^{i-1})^* \models A(u)$ and hence $R_c^* \models A(u)$. This, together with the fact that $R_c^* \models u \approx c$, implies $R_c^* \models A(c)$. □

**Lemma 20.** *In case that $t = c$ and $c$ is not irreducible w.r.t. $R_c$, we have $R_t^* \not\models \Delta_t$.*

*Proof.* Since $c$ is not irreducible w.r.t. $R_c$, there is a generative clause $C^i = \Gamma_i \to \Delta_i \vee c \approx u$ in $N_c$ for some $u \in \Sigma_u$. Therefore, by Lemma 6, there is a clause of the form $\Gamma \to \Delta_1 \vee x \approx u \vee \Delta_2 \in \mathcal{S}_q$ with $\Gamma \subseteq \Gamma_Q$, $\Delta_1 \not\succ_q x \approx u$, and $\Delta_2 \subseteq \Delta_Q$. Every literal in $\Delta'_2$ is in $\Delta_Q$, so it is of the form $B(x)$ for some $B \in \Sigma_A$, and by Condition C2, it cannot be the case that $B(x) \succ_q u$, and hence we have $\Delta_2 \not\succ_q x \approx u$.

Suppose that $u$ is not the normal form of $c$ w.r.t. $R_c$. Then, there exists a generative clause $C^j = \Gamma_j \to \Delta_j \vee u \approx u'$ for some $u'$ with $u \succcurlyeq u'$. Notice that since $c >_c u$, we have $j < i$. Thus, by Lemma 6, there exists a clause $\Gamma' \to \Delta'_1 \vee \Delta'_2 \vee u \approx u' \in \mathcal{S}_q$ with $\Gamma'\sigma_c = \Gamma_i$, $\Delta'_1\sigma_c = \Delta_i$, and $\Delta'_2 \subseteq \Delta_Q$. Notice that we have $\Delta'_1 \not\succ_q u \approx u'$ due to $u \approx u' >_c \Delta_i$; furthermore, every literal in $\Delta'_2$ is in $\Delta_Q$, so it is of the form $B(x)$ for some $B \in \Sigma_A$, and by C2, it cannot be the case that $B(x) \succ_q u$ or

$B(x) \succ_q u'$. Thus, literal $u \approx u'$ is maximal in this clause. We are therefore in the conditions of the $\mathsf{Eq}$ rule, and since this rule is not applicable, there is a clause $\Gamma \wedge \Gamma' \rightarrow \Delta \vee \Delta_1 \vee \Delta_2 \vee \Delta'_1 \vee \Delta'_2 \vee x \approx u' \,\hat{\in}\, \mathcal{S}_q$. Notice that $(\Gamma \wedge \Gamma')\,\sigma_c \subseteq \Gamma_c$, and the mapping by $\sigma_c$ of the head of this clause is ground. Thus, by Lemma 7 we have $\Gamma_i \wedge \Gamma_j \rightarrow \Delta_i \vee \Delta_j \vee c \approx u' \,\hat{\in}\, N_c$. Furthermore, we have that either $x \approx u >_c \Delta_i \vee \Delta_j \vee c \approx u'$, so by Lemma 8, we have $(R_c^{i-1})^* \models \Delta_i \vee \Delta_j \vee c \approx u'$; however, since $C^i$ is generative, condition R1 ensures $(R_c^{i-1})^* \not\models \Delta_i$. In addition, Lemma 11 and Lemma 12 ensure $(R_c^{i-1})^* \not\models \Delta_j$. Thus, we conclude $(R_c^{i-1})^* \models c \approx u'$, but this contradicts condition R3 for $C^i$. Hence, we reach a contradiction and conclude that $u$ must be the normal form of $c$ w.r.t. $R_c$.

Now, by definition of $\Delta_c$, we have that $L$ is of the form $B(c)$ for some $B \in \Sigma_A$. Since $R_c$ is Church-Rosser, there must be a generative clause $C^k$ of the form $\Gamma_k \rightarrow \Delta_k \vee B(u)$ (recall that $\mathsf{p}$-terms in rewrite rules are irreducible w.r.t. $R_c$ due to condition R3). Thus, by Lemma 6 there exists a clause $\Gamma'' \rightarrow \Delta''_1 \vee \Delta''_2 \vee B(u)$ with $\Gamma'' \subseteq \Gamma_Q$, $\Delta''_1 \not\succ_q B(u)$ and $\Delta''_2 \subseteq \Delta_Q$. Notice that due to condition C2 we have $\Delta_Q \not\succ_q B(u)$. Thus, we are in the conditions of the $\mathsf{Eq}$ rule, and hence there is a clause $\Gamma \wedge \Gamma'' \rightarrow \Delta \vee \Delta'' \vee B(x) \,\hat{\in}\, \mathcal{S}_q$. Clearly, the image of the body by $\sigma_c$ is in $\Gamma_c$, and the image of the head by $\sigma_c$ is ground, so by Lemma 7, we have $\Gamma_i \wedge \Gamma_k \rightarrow \Delta_i \vee \Delta_k \,\hat{\in}\, N_c$; observe that since $B(c) \in \Delta_c$, it is not present in the head of the ground clause. Observe that $k > i$ due to the choice of ordering $>_c$, and since $B(u) >_c \Delta_i \vee \Delta_k$, by Lemma 8, we have $(R_c^{k-1})^* \models \Delta_i \vee \Delta_k$. However, condition R1 ensures $(R_c^{k-1})^* \not\models \Delta_k$. In addition, Lemma 11 and Lemma 12 ensure $(R_c^{k-1})^* \not\models \Delta_i$. Thus, we reach the desired contradiction. □

**Lemma 21** (Query Refutation)**.** *The model fragment $R_t^*$ is such that $R_t^* \models \Gamma_t$ and $R_t^* \not\models \Delta_t$.*

*Proof.* In order to prove that $R_t^* \models \Gamma_t$, observe that condition L7 implies that for each $A \in \Gamma_t$ we have $\Gamma_t \rightarrow A \,\hat{\in}\, N_t$, so if $t \neq c$ or $c$ is irreducible w.r.t. $R_t$, the result follows from Corollary 17. Otherwise, the result follows from Lemma 19.

Now, we prove $R_t^* \not\models \Delta_t$, using proof by contradiction. Let $L$ be the smallest literal $L \in \Delta_t$ such that $R_t^* \models L$. If $L$ is an inequality $s_1 \not\approx s_2$, by condition L3 it must be the case that $t \neq c$ and $s_1 \not\approx s_2 \in \Delta_{\mathsf{Rt}}$, which contradicts Lemma 18. Hence, let $L$ be an equality, written as $s_1 \approx s_2$, with $s_1 >_t s_2$.

In the case where $t = c$ and $c$ can be reduced w.r.t. $R_c$, the results follows from Lemma 20. Thus, we assume that if $t = c$, then $c$ is irreducible w.r.t. $R_c$. Since $R_t^* \models L$, there exists a position $p$ such that $s_1|_p = l^i$ for some rule $l^i \Rightarrow r^i \in R_t$. Since

$L \geq_t l^i \approx r^i$, we have that $L \geq_t \Delta^i \vee l^i \approx r^i$. Let $l \bowtie r$ be an arbitrary literal of $\Delta^i \vee l^i \approx r^i$, where $l \geq_t r$. We consider the possible forms of $l \bowtie r$; please note that we exclude all forms that cannot appear in $\Delta_t$ by its definition.

- $l$ and $r$ are $\mathsf{a}$-terms. In that case, condition L3 and the definition of $>_t$ ensure that $t \geq_t l$ and $t \geq_t r$. Notice that by condition L5, if $t \neq c$, we cannot have that $l \bowtie r$ is $t \approx u$, or $t \approx t'$, or $t' \approx u$ for $t' >_t u$. If $t = c$ and $c$ is irreducible w.r.t. $R_c$, we can also discard the case $t \approx u$, plus all cases with $t'$, since $t'$ does not exist in this case. It can also not be of the form $l \approx l$ or $l \not\approx l$, since then we contradict Lemma 13. If it is of the form $t \not\approx t'$, by Lemma 15 we have $R_t^* \models t \not\approx t'$, and hence by Lemma 11 $(R_t^{i-1})^* \models t \not\approx t'$, which makes $C^i$ no longer generative. For $t \not\approx u$ (resp. $t' \not\approx u$) with $t \notin \Sigma_u$ (resp. $t' \notin \Sigma_u$), the argument is analogous. For the remaining case, $l \in \Sigma_u, r \in \Sigma_u$, and since $l \bowtie r \notin \Delta_t$, we have $l \bowtie r \notin \Delta_{\mathsf{Rt}}$, due to condition L4, and hence $R^*_{\mathsf{Rt}} \models l \bowtie r$, so $R_t^* \models l \bowtie r$ by Lemma 18. Since $C^i$ is generative, Lemma 12 ensures $l \bowtie r$ must be $l^i \approx r^i$. But this implies that $L$ must be a function-free ground atom (since $t$ is irreducible, and so is $t'$, if it exists), and hence in $\mathsf{Rt}$. But then, by condition L4, $L \in \Delta_{\mathsf{Rt}}$. However, Lemma 14 implies we have $R^*_{\mathsf{Rt}} \models L$; but due to condition L2, we have $L \notin \Delta_{\mathsf{Rt}}$; hence, we obtain a contradiction.

- $l$ and $r$ are $\mathsf{p}$-terms with $r = \mathsf{true}$. Since $L \geq_t l \approx \mathsf{true}$ and $L \in \Delta_t$, we have that either $l \approx \mathsf{true} \in \Delta_t$, in which case we have a contradiction, or $l$ is an $\mathsf{a}$-reduction of an atom in $\Delta_t$, and hence $l \approx \mathsf{true}$ is an atom in $\mathsf{Rt}$. By condition L2, either $l \approx \mathsf{true}$ is in $\Delta_{\mathsf{Rt}}$, which results in a contradiction due to condition L4 or $R^*_{\mathsf{Rt}} \models l \approx \mathsf{true}$. In the latter case, we have that by Lemma 18, $R_t^* \models l \approx \mathsf{true}$; hence by Lemma 12 and the fact that $C^i$ is generative, $l \approx \mathsf{true} = l^i \approx r^i$. However, since $l^i$ reduces $s_1$, and $l^i$ is a $\mathsf{p}$-term, we have $L = l \approx \mathsf{true}$, and hence $l \approx \mathsf{true} \in \Delta_t$; since $l \approx \mathsf{true} \in \Omega$, we obtain by condition L4 that $L \in \Delta_{\mathsf{Rt}}$, which is a contradiction with $R^*_{\mathsf{Rt}} \models l \approx \mathsf{true}$ according to Lemma 14.

□

Before we move on to the next section, we prove a key property of $R_t^*$, namely, that none of the ontology clauses expressing an "at-most" number restriction is triggered by elements of $R_t^*$ mentioning functional terms. As we discussed in Section 6.3.1, this is crucial to ensure that all atoms satisfying the body of an "at-most" number restriction clause can be found in the same fragment.

### 6.3.3 Naming of Nominal-like Elements

In this section we prove that if $t$ acts "like a nominal" in our model, that is, if $t$ is an $S$-filler for a constant, where $S$ is an "at-most" numerically restricted role, then $R_t^* \models t \approx u$ for some $u \in \Sigma_u$. This is expressed in the following lemma:

**Lemma 22.** *Suppose $t$ is irreducible w.r.t. $R_t$. For any DL-clause of the form DL4 and $u \in \Sigma_u$ irreducible w.r.t. $R_t$, if $t = c$ or $t = f(t')$ for some $t' \neq u$ with $f \in \Sigma_f$, then $R_t^* \not\models B_1(u) \wedge S_{B_2}(u, t)$.*

Since the proof is technically involved, we provide a brief overview of it in Section 6.3.3.1. Next, in Section 6.3.3.2, we give preliminary definitions which we will use in the remainder of the section. In Section 6.3.3.3 we prove several intermediate results. Finally, in Section 6.3.3.4, we give the proof of Lemma 22.

#### 6.3.3.1 Proof overview

Lemma 22 will be easy to prove when the depth of $u$ is smaller than $\Lambda$. Indeed, if we assume $R_t^* \models S_{B_2}(u, t)$, it is not hard to show that there exists a clause $\Gamma' \rightarrow \Delta' \vee S(u, x) \in \mathcal{S}_v$ with $\Delta' \not\succ_v S(u, x)$ such that the grounding of this clause is satisfied by $R_t^*$, with $R_t^* \not\models \Delta'\sigma_t$. If the depth of $u$ is smaller than $\Lambda$, we will see that saturation by the $\mathsf{Nom}$ rule ensures that $t$ collapses into auxiliary constants. The crux of the proof consists therefore in showing that in cases where the $\mathsf{Nom}$ rule is not applicable because $u$ has the maximum depth, there are clauses in $\mathcal{S}_v$ which ensure that $t$ collapses into a constant.

Let $o_i$ be the auxiliary constant $o_{\rho_i}$, where $\rho_i$ is the prefix of $\rho$ length $i$. We prove the main result by showing that there exists a "chain"

$$S_1(o_0, o_1), S_2(o_1, o_2), \cdots, S_n(o_{n-1}, o_n)$$

with $o_n = u$, with $S_i \in \Sigma_S$ for $1 \leq i \leq n$, such that for every $i$ with $1 \leq i \leq n$, $o_i$ is irreducible w.r.t. $R_t^*$. In order to do this, we prove in Lemma 25 that for each $o_i$, some context will contain u.t.r. a clause of the form:

$$\Gamma_{\mathsf{Rt}} \wedge \Gamma \rightarrow \Delta_{\mathsf{Rt}} \vee \Delta \vee \Delta_{\mathsf{Eq}}, \tag{8}$$

where:

- $\Gamma_{\mathsf{Rt}} \subseteq \mathsf{Rt}$, $\Delta_{\mathsf{Rt}} \subseteq \mathsf{Rt}$,

- $\Gamma$ and $\Delta$ contain only non-ground literals in $\mathsf{Su}$ and $\mathsf{Pr}$, respectively, and in particular $\Delta$ does not contain equalities of the form $x \approx u$ or $y \approx u$ for any $u \in \Sigma_u$,
- $\Delta_{\mathsf{Eq}}$ is of the form

$$x \approx u_1 \vee \cdots \vee x \approx u_n \vee y \approx u'_1 \vee \cdots \vee y \approx u'_m,$$

  with $u_i, u'_j \in \Sigma_u$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

These clauses are carefully defined to ensure that $R^*_t \models \Gamma_{\mathsf{Rt}}$ and $R^*_t \not\models \Delta_{\mathsf{Rt}}$. Furthermore, we show in Lemma 27 that if the clauses of the form in (8) for two different constants $o_i$, $o_j$ with $1 \leq i < j \leq n$ appear in the same context and correspond to the same $\Gamma$ and $\Delta$, then it will no longer be the case that all elements $o_0, \cdots o_n$ are irreducible w.r.t. $R^*_t$. As we show in Lemma 28, this leads to a collapse of the "chain" and implies that $u$ is no longer irreducible w.r.t. $R^*_t$, contradicting our hypothesis.

We therefore conclude that clauses of the form in (8) for two different constants $o_i$, $o_j$ with $1 \leq i < j \leq n$ appearing in the same context cannot have the same $\Gamma$ and $\Delta$. Observe that there are only a limited number of pairs $\langle \Gamma, \Delta \rangle$ where $\Gamma$ and $\Delta$ are of the form defined in (8). Furthermore, $\Lambda$ is lower-bounded by the number of pairs $\langle \Gamma, \Delta \rangle$ and the total number of contexts. Using this observation and the pigeonhole principle, we conclude that there exists a clause of the form in (8) contained u.t.r. in $\mathcal{S}_v$ with $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$. Finally, we show in Lemma 29 that this clause is satisfied by $R^*_t$ in a way that ensures that $\Delta_{\mathsf{Eq}}\sigma_t$ is satisfied, and therefore $t$ collapses into a constant.

#### 6.3.3.2 Basic Definitions

For this section, we use the following notation convention: if a context structure is represented as $\mathcal{D}_i$, with $i \in \mathbb{N}$, we represent its components as $\mathcal{V}^i, \mathcal{E}^i, \mathsf{core}^i, \mathcal{S}^i, \gg^i$, and $\theta^i$; furthermore, we represent $\mathcal{S}^i(v)$ as $\mathcal{S}^i_v$ for any $v \in \mathcal{V}^i$. Similarly, if a context structure is represented as $\mathcal{D}$, we represent its components as $\mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \gg$, and $\theta$. We will also refer to a context structure with no occurrences of synthetic constants as a *pure* context structure. Also, if $\Gamma_1 \wedge \Gamma_2 \rightarrow \Delta_1 \vee \Delta_2$ is a clause with $\Gamma_1 \subseteq \mathsf{Su}_\tau$, $\Gamma_2 \cap \mathsf{Su}_\tau = \emptyset$, $\Delta_1 \subseteq \mathsf{Pr}_\tau$, $\Delta_2 \cap \mathsf{Pr}_\tau = \emptyset$, we say that the *type of this clause* is $\langle \Gamma_1, \Delta_1 \rangle$.

**Definition 16** (Occurrences of literals)**.** *Let $\mathcal{D}$ be a context structure. An* occurrence of a literal $L$ *in $\mathcal{D}$ is a 4-tuple $\langle \mathcal{D}, v, C, L \rangle$, where $C \in \mathcal{S}_v$ and $L$ appears in $C$.*

For every inference rule in Tables 5.1 and 5.2, we refer to all clauses to be added to the context structure as *conclusions* of that inference, except in the case of Succ or $r$-Succ, and to all clauses mentioned in the preconditions, except conclusions, as *premises* of that inference. In the case of Pred (resp. $r$-Pred), we refer to the clause in $v$ (resp. $v_r$) as the *main premise*, and to the other premises as the *side premises*.

**Definition 17** (Selected literal)**.** *Let $C$ be a premise in an inference by a rule from Tables 5.1 and 5.2. The* selected literals *of $C$ are defined as: $A_i\sigma$ in* Hyper*, $s_1 \approx t_1$ or $s_2 \bowtie t_2$ (correspondingly) in* Eq*, $t \not\approx t$ in* Ineq*, $s \approx t_1$ in* Factor*, $A$ in* Join*; $B_1(o_\rho)$ or $S(o_\rho, x)$ (correspondingly) in* Nom*, $A$ in* Succ*, and $A\sigma$ in $r$-*Succ*, $C_i$ in the main premise or $C_i\sigma$ in the side premises of* Pred *and $r$-*Pred*.*

**Definition 18** (Generated literal)**.** *Let $C$ be a conclusion in an inference by a rule from Tables 5.1 and 5.2, the* generated literals *of $C$ are defined as follows: $A$ in* Core*, $\Delta\sigma$ in* Hyper*, $s_2[s_1/t_1]$ or $s_2[t_1]_p \bowtie t_2$ (correspondingly) in* Eq*, $t_1 \not\approx t_2$ in* Factor*, the set of all $x \approx o_{\rho \cdot S^i_{B_2}}$ in* Nom*, $A'$ in* Succ*, each $L_i\sigma$ in* Pred*, $A$ in $r$-*Succ*, and $L_i\sigma$ in $r$-*Pred*.*

**Definition 19** (Descendance)**.** *Let $(\mathcal{D}_0, \cdots)$ be a derivation. Consider an occurrence of a literal $\langle \mathcal{D}_n, v, C, L\rangle$ with $n \in \mathbb{N}$ and $\mathcal{D}_n$ occurring in the derivation. We define the set of* descendants *of this occurrence inductively:*

- *Occurrence $\langle \mathcal{D}_n, v, C, L\rangle$ descends from itself.*
- *For every $m$ with $m \geq n$ and every occurrence of a literal $\langle \mathcal{D}_m, v', C', L'\rangle$ which descends from $\langle \mathcal{D}_n, v, C, L\rangle$, we have that:*
  - *If $C' \in \mathcal{S}^{m+1}_{v'}$, then $\langle \mathcal{D}_{m+1}, v', C', L'\rangle$ descends from $\langle \mathcal{D}_n, v, C, L\rangle$.*
  - *If the $m$-th inference step is an application of* Elim *to $C'$ on $\mathcal{S}^m_{v'}$, and there exists a clause $C'' \in \mathcal{S}_{v'}$ subsuming $C'$ which contains $L'$, then $\langle \mathcal{D}_{m+1}, v', C'', L'\rangle$ descends from $\langle \mathcal{D}_n, v_n, C_n, L\rangle$.*
  - *If $C'$ participates in the inference from $\mathcal{D}_m$ to $\mathcal{D}_{m+1}$ for any rule other than* Elim*, then for every inference conclusion $C''$ to be added to a context $w$, and for each generated literal $L''$, we have that $\langle \mathcal{D}_{m+1}, w, C'', L''\rangle$* descends *from $\langle \mathcal{D}_n, v_n, C_n, L\rangle$.*

Finally, our proof uses a special property of the calculus: inferences involving ground atoms can generally be "postponed." For instance, if there is an inference that selects ground atoms in clauses $C_1$ and $C_2$ to produce conclusion $C$, we can

simply skip this step and then, for each inference that takes $C$ as a premise, apply the inference directly to either $C_1$ or $C_2$, appropriately. This occurs because the properties of Definition 5 and the preconditions of inference rules in Tables 5.1 and 5.2 together ensure that ground atoms in the head of a clause $C$ cannot "block" inferences on other literals in the head of $C$, with the exception of other ground literals; furthermore, the existence of clauses of the form $A \to A$ ensure that we can perform any relevant inferences on ground literals. Afterwards, the resulting collection of clauses can be recombined using the $\mathsf{Join}$ rule to recover (u.t.r.) the conclusion of the original inference on $C$. We next describe formally the collections of clauses obtained when "postponing" inferences selecting ground atoms, and give a precise meaning to the notion of postponing such inferences.

**Definition 20** (Ground decomposition)**.** *Let $\Gamma \to \Delta$ be a clause. We define a* ground decomposition *of $\Gamma \to \Delta$ as a triple $\langle T, \lambda, \mu \rangle$ where $T$ is a non-empty tree, $\lambda$ maps nodes of $T$ to clauses, and $\mu$ maps nodes of $T$ to literals, which satisfies the following properties:*

- *For each node $v \in T$, $\lambda(v)$ is of the form $\Gamma_v^{\mathsf{Rt}} \wedge \Gamma_v \to \Delta_v \vee \Delta_v^{\mathsf{Rt}}$, where $\Gamma_v \subseteq \Gamma$, $\Delta_v \subseteq \Delta$, $\Gamma_v^{\mathsf{Rt}} \subseteq \mathsf{Rt}$ and $\Delta_v^{\mathsf{Rt}} \subseteq \mathsf{Rt}$, and $\mu(v)$ is a literal appearing in $\lambda(v)$.*
- *For each node $v \in T$ and $A \in \Delta_v^{\mathsf{Rt}}$ such that $A$ is a ground equality, there exists a successor $w$ of $v$ such that $\mu(w) = A$ and $A \in \Gamma_w^{\mathsf{Rt}}$.*
- *For each node $v \in T$ and $A \in \Delta_v^{\mathsf{Rt}}$ such that $A$ is a ground inequality $u \not\approx a$, there exists successors $w_1, \cdots, w_k$ of $v$ such that $\Delta_{w_i}^{\mathsf{Rt}}$ contains $\mu(w_i)$, which is of the form $u_{i+1} \approx u_i$, with $u_i \in \Sigma_u$ for $1 \leq i \leq k+1$, $u_{k+1} = u$ and $u_1 = a$.*
- *For each node $v \in T$ and $A \in \Gamma_v^{\mathsf{Rt}}$ such that $A$ is a ground equality, there exists a successor $w$ of $v$ such that $\mu(w) = A$ and $A \in \Delta_w^{\mathsf{Rt}}$.*
- *For each leaf $v$, we have that either $\Gamma_v^{\mathsf{Rt}} = \top$ and $\Delta_v^{\mathsf{Rt}} = \mu(v)$, or $\Delta_v^{\mathsf{Rt}} = \bot$ and $\Gamma_v^{\mathsf{Rt}} = \mu(v)$.*

Consider a pair $(C, \langle T, \lambda, \mu \rangle)$, where $C = \Gamma \to \Delta$ is a clause, and $\langle T, \lambda, \mu \rangle$ is a ground decomposition of $C$. Let $\xi$ be an inference distinct from $\mathsf{Elim}$ which uses $C$ as a premise.

**Definition 21.** *The* natural successor *of $(C, \langle T, \lambda, \mu \rangle)$ by $\xi$ is the pair $(C', \langle T', \lambda', \mu' \rangle)$ defined as follows:*

- *if $\xi$ uses $\mathsf{Succ}$ or $r$-$\mathsf{Succ}$, then $C' = C$ and $\langle T', \lambda', \mu' \rangle = \langle T, \lambda, \mu \rangle$.*

- *if the selected literal in $C$ is in $\Delta$, but does not appear in $\Delta_w$ for any $w \in T$, we define $C'$ as the conclusion of inference $\xi$, and $\langle T', \lambda', \mu' \rangle$ as $\langle T, \lambda, \mu \rangle$. Observe that if $C' = \Gamma' \to \Delta'$, then $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$, so $\langle T, \lambda, \mu \rangle$ is a ground decomposition of $C'$.*

- *if the selected literal in $C$ is in $\Delta$ and appears in $\Delta_w$ for some $w \in T$, and $L$ is non-ground, observe that by definition of $\succ_v$ it cannot be the case that $A \succ_v L$ for some ground equality $A$, and since $L$ is maximal among the literals of $\Delta$, then is maximal among the literals of $\Delta_w$. In this case, we define $C'$ as the conclusion of $\xi$, and we define $\langle T', \lambda', \mu' \rangle$ identically to $\langle T, \lambda, \mu \rangle$, except that we replace $C_w$ by the conclusion of an inference $\xi'$ identical to $\xi$ except in that $C$ is replaced by $C_w$, and unnecessary additional premises do not participate. It is easy to check that $\langle T', \lambda', \mu' \rangle$ is a ground decomposition of $C'$.*

- *if the selected literal(s) in $C$ are in $\Gamma$, then we define $C'$ as the conclusion of $\xi$, and for each $w \in T$ we replace each $C_w$ by the conclusion of an inference $\xi'$ identical to $\xi$ except in that $C$ is replaced by $C_w$, and unnecessary additional premises do not participate. It is easy to check that $\langle T', \lambda', \mu' \rangle$ is a ground decomposition of $C'$.*

In other cases, the natural successor is undefined. The natural successor of a clause and its ground decomposition by an inference represent the result of applying such inference to the clause and the ground decomposition simultaneously.

**Definition 22.** *The* alternative successor *of $(C, \langle T, \lambda, \mu \rangle)$ by $\xi$ is defined as the following pair $(C', \langle T', \lambda', \mu' \rangle)$:*

- *if the selected literal in $C$ is in $\Delta$ and is of the form $u \not\approx a$ and there exists some $w$ in $T$ where $\Delta_w^{\mathsf{Rt}}$ contains $u' \not\approx a$, with $u' = u$, or there are successors $w'_1, \cdots, w'_l$ such that $\Delta_{w'_i}^{\mathsf{Rt}}$ contains a literal $u_{i+1} \approx u_i$ in the head, with $u_{l'+1} = u'$, and $u_1 = u$, we define $C'$ as the conclusion of $\xi$, and define $\langle T', \lambda', \mu' \rangle$ as the following extension of $\langle T, \lambda, \mu \rangle$: since in this case $C$ participates in $\xi$ together with a premise $C''$ where the selected literal is an equality of the form $u \approx o$, we add a successor $w'$ (or $w'_{l+1}$) of $w$ and define $\lambda(w') = C''$ and $\mu(w') = u \approx o$.*

- *if the selected literal $L$ in $C$ is in $\Delta$, appears in $\Delta_w^{\mathsf{Rt}}$ for some $w \in T$, and is a ground equality, we define $C'$ as the conclusion of $\xi$ and define $\langle T', \lambda', \mu' \rangle$ as the extension of $\langle T, \lambda, \mu \rangle$ which adds a successor $w'$ to $w$ and where $\lambda(w')$ is the conclusion of an inference $\xi'$ identical to $\xi$ except that $C$ is replaced by $L \to L$; also, $\mu(w') = L$.*

- *if the selected literal $L$ in $C$ is in $\Gamma$, and appears in $\Gamma_w^{\mathsf{Rt}}$ for some $w \in T$, and is a ground equality, we have that one of the other premises in $C$ must be of the form $\Gamma'' \to \Delta'' \vee L$, so we define $C'$ as the conclusion of $C$ and $\langle T', \lambda', \mu' \rangle$ as the extension of $\langle T, \lambda, \mu \rangle$ which adds a successor $w'$ to $w$ with $\lambda(w') = \Gamma'' \to \Delta'' \vee L$ and $\mu(w') = L$.*

In other cases, the alternative successor is undefined. The alternative successor represents the result of "postponing" (in the sense discussed above) an inference which selects ground atoms. Notice that for any clause $C$, ground decomposition of $C$, and inference $\xi$, if the natural successor is undefined, then the alternative successor is well defined.

#### 6.3.3.3 Preliminary Results

We start this section by introducing two preliminary lemmas related to clauses contained up to redundancy in context structures.

**Lemma 23.** *Let $(\mathcal{D}_0, \cdots)$ be a derivation. For every $i \geq 0$, if $C \,\hat{\in}\, \mathcal{S}_v^i$, then $C \,\hat{\in}\, \mathcal{S}_v^j$ for each $j \geq i$. Furthermore, if $C = \Gamma \to \Delta$ and $\Delta$ does not contain a tautology, then for each $j \geq i$ there exist $\Gamma'$ and $\Delta'$ such that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$, with $\Gamma' \to \Delta' \in \mathcal{S}_v^j$.*

*Proof.* We prove the lemma by induction. The base case is trivially true by hypothesis. Now, suppose $C \,\hat{\in}\, \mathcal{S}_v^j$ for some $j \geq i$. If the head of $C$ contains a tautology, then $C \,\hat{\in}\, \mathcal{S}_v^{j+1}$ is trivially true, by definition of our redundancy notion. Otherwise, let us write $C$ as $\Gamma \to \Delta$; we then have that there exists a clause $\Gamma' \to \Delta' \in \mathcal{S}_v^j$, with $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$. If the inference from $\mathcal{D}_j$ to $\mathcal{D}_{j+1}$ is anything other than an application of $\mathsf{Elim}$ on $\Gamma' \to \Delta'$ in $\mathcal{S}_v^j$, then we have $\Gamma' \to \Delta'$ in $\mathcal{S}_v^{j+1}$, which implies the desired result. Otherwise, we have that $\Gamma' \to \Delta' \,\hat{\in}\, \mathcal{S}_v^j \backslash \{\Gamma' \to \Delta'\}$. Notice that $\Delta'$ cannot contain a tautology, for otherwise $\Delta$ contains a tautology, contrarily to our assumption. Thus, there exists $\Gamma'' \to \Delta'' \in \mathcal{S}_v^j$ with $\Gamma'' \subseteq \Gamma'$, and $\Delta'' \subseteq \Delta'$, and since $\mathsf{Elim}$ is not applied on this clause, we have $\Gamma'' \to \Delta'' \in \mathcal{S}_v^{j+1}$, which implies the result of the lemma since $\Gamma'' \subseteq \Gamma$, and $\Delta'' \subseteq \Delta$.

For the second part of the lemma, suppose $C = \Gamma \to \Delta$ and $\Delta$ does not contain a tautology. Since $C \,\hat{\in}\, \mathcal{S}_v^i$, there exist $\Gamma'$ and $\Delta'$ with $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, and $\Gamma' \to \Delta' \in \mathcal{S}_v^i$. By the first part of the lemma, we have $\Gamma' \to \Delta' \in \mathcal{S}_v^j$ for every $j \geq i$, and since $\Delta'$ does not contain a tautology, we have $\Gamma'' \to \Delta'' \in \mathcal{S}_v^j$ with $\Gamma'' \subseteq \Gamma' \subseteq \Gamma$, and $\Delta'' \subseteq \Delta' \subseteq \Delta$. □

If $\mathcal{S}$ is a set of clauses and $\Gamma \to \Delta \vee L$ is a clause, we write $\Gamma \to \Delta \vee [L] \,\hat{\in}\, \mathcal{S}$ to represent the claim that there exists a clause $\Gamma' \to \Delta' \vee L \in \mathcal{S}$ with $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$. Notice that $\Gamma \to \Delta \vee [L] \,\hat{\in}\, \mathcal{S}$ implies $\Gamma \to \Delta \vee L \,\hat{\in}\, \mathcal{S}$.

**Lemma 24.** *For all inference rules in Tables 5.1 and 5.2, if the premises are contained u.t.r. in $\mathcal{S}_v$ for a context $v$ in a saturated context structure $\mathcal{D}$, then the conclusion is contained u.t.r. in $\mathcal{S}_w$ for every relevant context $w$ in $\mathcal{D}$, except in the case of* Eq*, when the selected literal is of the form $s_2 \not\approx t_2$ with $s_2 \approx t_2 \in \Delta_2$, and no clause in $\mathcal{S}_v$ subsumes $\Gamma_2 \to \Delta_2 \vee s_2 \approx t_2$.*

*Proof.* We consider each inference rule separately.

- For the Hyper rule, suppose $\Gamma_i \to \Delta_i \vee A_i\sigma \,\hat{\in}\, \mathcal{S}_v$ for some $1 \leq i \leq n$. Due to the form of $A_i\sigma$, it cannot be part of any tautology. If $\Delta_i$ contains a tautology, then the result is satisfied trivially. Otherwise, there exists a clause $\Gamma'_i \to \Delta'_i \in \mathcal{S}_v$ with $\Gamma'_i \subseteq \Gamma_i$, $\Delta'_i \subseteq \Delta_i \vee A_i\sigma$. If $A_i\sigma \notin \Delta'_i$, then the result follows trivially. Otherwise, we have that for each $1 \leq i \leq n$, there exists a clause $\Gamma'_i \to \Delta''_i \vee A_i\sigma \in \mathcal{S}_v$ with $\Delta''_i \subseteq \Delta_i$. Since $\mathcal{D}$ is closed by application of Hyper, we have that $\bigwedge_{i=1}^{n} \Gamma'_i \to \bigvee_{i=1}^{n} \Delta''_i \vee \Delta\sigma \,\hat{\in}\, \mathcal{S}_v$, which entails the desired claim.

- For the Eq rule, suppose $\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \,\hat{\in}\, \mathcal{S}_v$. If $\Delta_1$ contains a tautology, we reason as in the previous case. If $s_1 \approx t_1$ is part of a tautology, it can only be the case that $t_1 = s_1$, for $s_1 \not\approx t_1 \succ_v s_1 \approx t_1$. But then, the conclusion of the inference is subsumed by the second premise, since $s_2[t_1]_p = s_2$. Similarly, if $\Delta_2$ contains a tautology, the conclusion trivially follows, and if $s_2 \bowtie t_2$ is part of a tautology, we have two possibilities: the literal is of the form $s_2 \not\approx t_2$, which is covered by the exception in the lemma, or the literal is of the form $s_2 \approx s_2$, in which case $s_2$ is an a-term, so $s_1 = s_2$ and hence the result for the first premise implies the result for the conclusion. Finally, if none of the premises contain a tautology, there exist clauses $\Gamma'_1 \to \Delta'_1$ and $\Gamma'_2 \to \Delta'_2$ with $\Gamma'_1 \subseteq \Gamma_1$ and $\Gamma'_2 \subseteq \Gamma_2$. If $s_1 \approx t_1 \notin \Delta'_1$, this clause subsumes the conclusion; we reason analogously for $s_2 \bowtie t_2$. In the remaining case, we have clauses $\Gamma'_1 \to \Delta''_1 \vee s_1 \approx t_1 \in \mathcal{S}_v$, and $\Gamma'_2 \to \Delta''_2 \vee s_2 \bowtie t_2 \in \mathcal{S}_v$, and since Eq is not applicable, the result follows.

- For the Ineq rule, if $\Delta$ contains a tautology we use the same argument as in the previous cases. It $t \not\approx t$ is part of a tautology, the other part is $t \approx t \in \Delta$, and hence the result follows directly. Finally, if there exists a clause $\Gamma' \to \Delta' \in \mathcal{S}_v$ with $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, and $t \not\approx t \notin \Delta'$, the conclusion is already true. Otherwise,

there is a clause $\Gamma' \to \Delta'' \vee t \not\approx t \in \mathcal{S}_v$, with $\Delta'' \subseteq \Delta$, and since Ineq is not applicable, we conclude $\Gamma' \to \Delta'' \,\hat{\in}\, \mathcal{S}_v$, which implies the desired result.

- For Factor, if $\Delta$ contains a tautology we reason as in the previous cases. If either $s \approx t_1$ or $s \approx t_2$ is part of a tautology, the only possibility is $s = t_1$ or $s = t_2$, as in the case of the proof for the Eq rule. If $s = t_1$, the conclusion contains literals of the form $s \not\approx t_2$ and $s \approx t_2$, which form a tautology, so the result follows. If $s = t_2$, the conclusion contains a literal of the form $s \approx s$, which also forms a tautology, and the result follows.

- The Elim rule does not have a conclusion, so the result does not apply.

- For the Join rule, if $\Delta_1$ or $\Delta_2$ contain a tautology, the result follows trivially. Otherwise, the form of $A$ implies that it cannot be part of a tautology, so there exist clauses $\Gamma_1' \to \Delta_1'$ with $\Gamma_1' \subseteq \Gamma_1$, and $\Delta_1' \subseteq \Delta_1 \vee A$; if $A \notin \Delta_1'$, then the results follows trivially. We reason analogously for the second premise, and conclude that there exist clauses $\Gamma_1' \to \Delta_1'' \vee A \in \mathcal{S}_v$, with $\Delta_1'' \subseteq \Delta_1$, and $A \wedge \Gamma_2' \to \Delta_2' \in \mathcal{S}_v$, with $\Gamma_2' \subseteq \Gamma_2$, $\Delta_2' \subseteq \Delta_2$, and since Join is not applicable to the saturated context structure, we have $\Gamma_1' \wedge \Gamma_2' \to \Delta_1'' \vee \Delta_2' \,\hat{\in}\, \mathcal{S}_v$, which implies the result.

- For the Nom rule the argument is entirely analogous to the case for the Hyper rule.

- For the Succ and $r$-Succ rules, there is no conclusion, so the lemma follows trivially.

- For the Pred rule, for each $1 \leq i \leq n$, if $\Delta_i$ contains some tautology, the result follows trivially. Furthermore, due to its form, it cannot be the case that $L_i$ is part of a tautology, so there exists a clause $\Gamma_i' \to \Delta_i' \in \mathcal{S}_w$ with $\Gamma_i' \subseteq \Gamma_i$, $\Delta_i' \subseteq \Delta_i \vee C_i\sigma$. If $C_i\sigma \notin \Delta_i$, we have that the result follows trivially; otherwise, we assume that there exists a clause $\Gamma_i' \to \Delta_i'' \vee C_i\sigma \in \mathcal{S}_w$ with $\Delta_i'' \subseteq \Delta_i$. Similarly, we have that if $\bigvee_{i=1}^{k} L_i$ contains a tautology, the result follows trivially, so we assume that this is not the case, and therefore there exists a clause $\bigwedge_{i=1}^{m'} A_i' \wedge \bigwedge_{i=1}^{n'} C_i' \to \bigvee_{i=1}^{k'} L_i' \in \mathcal{S}_v$ with $\bigwedge_{i=1}^{m'} A_i' \subseteq \bigwedge_{i=1}^{m} A_i$, $\bigwedge_{i=1}^{n'} C_i' \subseteq \bigwedge_{i=1}^{n} C_i$, and $\bigvee_{i=1}^{k'} L_i' \subseteq \bigvee_{i=1}^{k} L_i$. With all these clauses, we are in the conditions of the Pred rule, since selected atoms in the premises are still maximal. Therefore, since the Pred is not applicable, there exists a clause

contained u.t.r. in $\mathcal{S}_v$ which subsumes the original conclusion, and therefore the result follows.

- For the $r$-$\mathsf{Pred}$ rule, the argument is analogous to the $\mathsf{Pred}$ rule.

□

Next, we present a lemma which shows that at least $n$ clauses of the form in (8) always exist in a context structure whenever an auxiliary constant of depth $n$ is mentioned in the context structure.

**Lemma 25.** *Let $o_\rho$ be an auxiliary constant with $|\rho| = n$. For every $i$ with $0 \leq i \leq n$, define $o_i$ inductively as follows: if $i = n$ then $o_n = o_\rho$, and if $0 \leq i < n$, let $o_{i+1} = o_{\rho' \cdot S}$ for some $S \in \Sigma_S$, and define $o_i$ as $o_{\rho'}$. Furthermore, assume that $\rho$ is of the form $S_1^{i_1} \cdot \cdots \cdot S_n^{i_n}$.*

*Let $\mathcal{D}$ be a derivation from a pure context structure. Suppose that there exists an occurrence of a literal $\langle \mathcal{D}, v, C, L \rangle$ for some context $v \neq v_r$ such that $L$ mentions $o_n$. Then, there exists a sequence of $m$ occurrences of literals, from $\langle \mathcal{D}_1, v_1, C_1, L_1 \rangle$ to $\langle \mathcal{D}_m, v_m, C_m, L_m \rangle$ with the following properties:*

*P1.* $C_m = C$.

*P2. for $1 \leq i < m$, the occurrence of $L_{i+1}$ in $\mathcal{D}_{i+1}$ descends from the occurrence of $L_i$ in $\mathcal{D}_i$;*

*P3. for each $1 \leq i \leq m$, we have that if $C_i = \Gamma_i \to \Delta_i \vee L_i$, then $\Gamma_i \to \Delta_i \vee [L_i] \,\hat{\in}\, \mathcal{D}$;*

*P4. for every $1 \leq i \leq n$, there exists $1 \leq a_i < m$, with $a_i < a_{i'}$ if $1 \leq i < i' \leq m$, such that $C_{a_i}$ is a clause of the form $\Gamma_i \to \Delta_i \vee S_i(o_i, x)$, with $L_{a_i} = S_i(o_i, x)$, and the inference from $\mathcal{D}_{a_i}$ to $\mathcal{D}_{a_i+1}$ uses $C_{a_i}$ in $v_{a_i}$ as a premise in for the $\mathsf{Nom}$ rule, with $S_i(o_i, x)$ as the selected literal.*

*P5. for each $i$ with $1 \leq i \leq m$, if $a_j \leq i < a_{j+1}$ for some $1 \leq j \leq n$ or $a_j \leq i$ with $j = n$, then $L_i$ mentions $o_j$.*

*P6. for $1 \leq i \leq n$, there exists $1 \leq b_i \leq m$, with $a_i < b_i < a_{i+1}$, such that $C_{b_i}$ is a clause of the form $\Gamma_i \to \Delta_i \vee \Delta_i^{\mathsf{Eq}} \vee x \approx o_i$, with $L_{b_i} = x \approx o_i$, $\Delta_i^{\mathsf{Eq}}$ is of the form:*

$$\bigvee_{j=1}^{k} x \approx u_j \quad \vee \bigvee_{j=k+1}^{k'} y \approx u_j \qquad\qquad u_j \in \Sigma_u \text{ for } 1 \leq j \leq k' \tag{9}$$

*$\Delta_i \subseteq \mathsf{Pr}_\tau \cup \mathsf{Rt}$, and the inference from $\mathcal{D}_{b_i}$ to $\mathcal{D}_{b_{i+1}}$ uses $C_{b_i}$ in $v_{b_i}$ as the main premise for the $\mathsf{Pred}$ rule.*

*Proof.* We define the sequence and show that the properties hold using reverse induction. The following will be our induction conditions, for $1 \leq i \leq m$:

1. Property P2 holds for $i$.
2. Either $L_i$ mentions $o_n$, or there exist $a_n > a_{n-1} > \cdots > a_{n'} > i$ satisfying the conditions of property P4, and $L_i$ mentions $o_{n'-1}$.
3. Property P3 holds for $i$.

For the base case, let $m$ be the smallest element in $\mathbb{N}$ such that $C \in \mathcal{S}_v^m$, and define $L_m$ as the literal containing $o_n$, $C_m = C$, and $v_m = v$. This ensures the sequence satisfies property P1. Furthermore, notice that all the induction conditions are satisfied in the base case; in particular, property P2 holds vacuously since it does not affect the case $i = m$; also, $L_m$ mentions $o_n$.

Now, suppose that $\langle \mathcal{D}_{i+1}, v_{i+1}, C_{i+1}, L_{i+1} \rangle$ have been defined for some $1 \leq i < m$. We assume that the induction conditions hold for $i + 1$. Since $o_{n'-1}$ is an auxiliary constant, we have that no clause of $\mathcal{O}$ mentions $o_{n'-1}$, and hence we have two options: (i) there exists at least one occurrence of a literal $\langle DD_i, v_i, C_i, L_i \rangle$ such that $\langle \mathcal{D}_{i+1}, v_{i+1}, C_{i+1}, L_{i+1} \rangle$ descends from it and $L_i$ mentions $o_{n'-1}$, in which case we choose one of such occurrences as the next element in the sequence, preferring whichever satisfies property P3 if one does; (ii) if there is no such clause, but the inference step from $\mathcal{D}_i$ to $\mathcal{D}_{i+1}$ introduces $o_{n'-1}$ via $\mathsf{Nom}$, in which case there exists an occurrence of a literal $\langle \mathcal{D}_i, v_i, C_i, L_i \rangle$ with $L_i = S_{n'-2}(o_{n'-2}, x)$, $C_i$ a premise used for the $\mathsf{Nom}$ inference, and $v_{i+1} = v_i$. In the latter case, we choose $\langle \mathcal{D}_i, v_i, C_i, L_i \rangle$ as the new occurrence in the sequence and define $a_{n'-1} = i$.

We now show that the induction conditions hold for $i$. Property P2 holds for $i$ due to the way we have defined $\langle \mathcal{D}_i, v_i, C_i, L_i \rangle$. Furthermore, we have that either $L_i$ mentions $o_n$, or there exist $a_n > a_{n-1} > \cdots > a_{n''} > i$ satisfying the conditions of property P4, and $L_i$ mentions $o_{n''-1}$, where $n'' = n'$ if we did not define $a_{n'-1}$ as $i$, or $n'' = n' - 1$ otherwise.

Finally, to prove property P3, we start from the fact that by induction hypothesis, $\Gamma_{i+1} \to \Delta_{i+1} \vee [L_{i+1}] \,\hat{\in}\, \mathcal{D}$ for some $1 \leq i < m$, and we consider the clause $\Gamma'_{i+1} \to \Delta'_{i+1} \vee L_{i+1} \in \mathcal{D}$ with $\Gamma'_{i+1} \subseteq \Gamma_{i+1}$, $\Delta'_{i+1} \subseteq \Delta_{i+1}$. Consider clause $\Gamma_i \to \Delta_i \vee L_i$, which we know is in $\mathcal{D}_i$. By Lemma 23, we know $C_i \,\hat{\in}\, \mathcal{D}$. We assume that an inference rule is applied to this clause on derivation step $i$, for otherwise $C_{i+1} = C_i$ and the result follows trivially.

If $\Delta_i \vee L_i$ contains a tautology, and $L_i$ is part of this tautology, since $\mathsf{Elim}$ is applied eagerly, we have that $C_{i+1}$ does not contain $L_{i+1}$, which contradicts our assumption. If $L_i$ is not part of the tautology, then $L_{i+1} = L_i$, and since $\mathsf{Elim}$ is applied eagerly and an inference rule is applied to the occurrence of $C_i$ in $\mathcal{D}_i$, we have $\Delta_{i+1} \subseteq \Delta_i$, $\Gamma_{i+1} = \Gamma_i$, and then $\Gamma_{i+1} \to \Delta_{i+1} \vee [L_{i+1}] \,\hat{\in}\, \mathcal{D}$ implies $\Gamma_i \to \Delta_i \vee [L_i] \,\hat{\in}\, \mathcal{D}$.

Now, if $\Delta_i \vee L_i$ does not contain a tautology, suppose there is $\Gamma'_i \to \Delta'_i$ in $\mathcal{D}$ with $\Gamma'_i \subseteq \Gamma_i$ and $\Delta'_i \subseteq \Delta_i \vee L_i$. Suppose, in addition, that $L_i$ is the literal selected in the $i$-th inference step, so we have $\Gamma_i \subseteq \Gamma_{i+1}$ and $\Delta_i \subseteq \Delta_{i+1}$. If $\Delta'_i$ does not contain $L_i$, the fact that $\Gamma_i \subseteq \Gamma_{i+1}$ and $\Delta_i \subseteq \Delta_{i+1}$ leads to a violation of the induction hypothesis. Hence, $\Delta'_i$ contains $L_i$ and the desired result holds.

If $L_i$ is not the selected literal, we have $L_{i+1} = L_i$, and suppose that $\Delta'_i$ does not contain $L_i$. We then consider the other premises of the inference rule at step $i$. By Lemma 23, all of them are contained up to redundancy in $\mathcal{D}$; by the eager application of $\mathsf{Elim}$, we know that none of them contains a tautology in the head and hence all of them have subsuming clauses in $\mathcal{D}$. If the selected literal in one of them is not in the corresponding subsuming clause in $\mathcal{D}$, then such subsuming clause already subsumes the conclusion; furthermore, it does not contain $L_i$, due to the fact that $\Delta'_i$ does not contain it either, and the way we define our sequence of occurrences of literals. However, this contradicts property P3 for $i + 1$. Therefore, all selected literals are present in the corresponding subsuming clauses in $\mathcal{D}$, and since $\mathcal{D}$ is saturated, we can apply the same inference rule with those premises and $\Gamma'_i \to \Delta'_i$ instead of $C_i$ in order to obtain that a clause subsuming $C_{i+1}$ not containing $L_{i+1}$ is in contained u.t.r. in $\mathcal{D}$, which again contradicts the hypothesis.

To conclude the proof, we prove the remaining properties. Property P4 holds due to the fact that the second induction condition holds for $i = 1$; indeed, since $\mathcal{D}_0$ is a pure context structure, it does not contain any synthetic constant, so it must be the case that $a_1 > 0$. Similarly, property P5 holds because of the second induction condition holds for all $1 \leq i \leq m$.

Finally, to prove property P6, we simply observe that for any $1 \leq i \leq n$, $L_{a_i+1}$ will be of the form $x \approx o_i$, and such literal can only have a descendant of the form $S(o_i, x)$ if it is propagated at some step via $\mathsf{Pred}$, for otherwise the only other inference rule that can select this literal is $\mathsf{Eq}$, but the generated literal by this inference is a unary ground atom, which can only be selected by inference rules to produce other unary ground atoms, or produce a clause without $o_i$, thus contradicting property P5. We then take $b_i$ as the inference step where such $\mathsf{Pred}$ inference is carried out, and the claims on the form of $C_{b_i}$ follow from the form of the main premise in $\mathsf{Pred}$. □

The following lemma is a preliminary result for ground decompositions of a clause $C$; in particular, it shows which literals of ground decompositions are satisfied by $R_t^*$, when we know which literals of $C$ are satisfied by $R_t^*$.

**Lemma 26.** *Let $C$ be a clause with $C \,\hat{\in}\, \mathcal{S}_v$ for $v$ in $\mathcal{D}$, and let $\langle T, \lambda, \mu \rangle$ be a ground decomposition of $C$ such that every clause in the ground decomposition is contained u.t.r. in $\mathcal{S}_v$. Let $w_0$ be the root of $T$. Suppose that for every $w \in T$ we have $R_t^* \models \Gamma_w \sigma_t$, and $R_t^* \not\models \Delta_w \sigma_t$. Then, $R_t^* \models \Gamma_{w_0}^{\mathsf{Rt}}$ and $R_t^* \not\models \Delta_{w_0}^{\mathsf{Rt}}$.*

*Proof.* To prove this lemma, we prove by structural induction the following property: for every node $w$ in $T$, if $L \in \Gamma_w^{\mathsf{Rt}}$, then $R_t^* \models L$ if and only if $L \neq \mu(v)$; and if $L \in \Delta_w^{\mathsf{Rt}}$, then $R_t^* \models L$ if and only if $L = \mu(v)$. We do this using the fact that all clauses are contained u.t.r. in $\mathcal{S}_v$, and because of the form of their bodies, and the hypothesis $R_t^* \models \Gamma_w \sigma_t$, by Corollary 17 they must be satisfied by $R_t^*$.

For the base case, we have two options: first, if $\lambda(w) = \mu(w) \wedge \Gamma_w \to \Delta_w$, then we have that if $R_t^* \models \mu(w)$, since $R_t^* \models \Gamma_w \sigma_t$, we have $R_t^* \models \Delta_w \sigma_t$, which contradicts hypothesis $R_t^* \not\models \Delta_w \sigma_t$. Hence, $R_t^* \not\models \mu(w)$. The second option is $\lambda(w) = \Gamma_w \to \Delta_w \vee \mu(w)$. Since $R_t^* \models \Gamma_w \sigma_t$, we have $R_t^* \models \Delta_w \sigma_t \vee \mu(w)$, and since $R_t^* \not\models \Delta_w \sigma_t$, we obtain the desired result.

For the induction case, consider a node $w$ and the corresponding clause $\Gamma_w^{\mathsf{Rt}} \wedge \Gamma_w \to \Delta_w \vee \Delta_w^{\mathsf{Rt}}$. Let $A \in \Gamma_w^{\mathsf{Rt}}$ and suppose $A \neq \mu(w)$. By definition of ground decomposition, there is a successor $w'$ of $w$ such that $\mu(w') = A$ and $A$ is in the head. By induction hypothesis, we have $R_t^* \models A$, which proves the desired result. Now, let $A \in \Delta_w^{\mathsf{Rt}}$ with $\mu(w) \neq A$, and suppose $A$ is an equality. By definition of ground decomposition, there is a successor $w'$ of $w$ such that $\mu(w') = A$ and $A$ is in the body. By induction hypothesis, we have that $R_t^* \not\models A$, which is the desired result. Finally, let $A \in \Delta_w^{\mathsf{Rt}}$ with $\mu(w) \neq A$, and suppose $A$ is an inequality $u \not\approx a$. By definition of ground decomposition, there are successors $w_1', \cdots, w_l'$ of $w$ such that $\mu(w_i') = u_{i+1} \approx u_i$. Furthermore, $u_{l+1} = u$ and $u_1 = a$. By induction hypothesis on each $w_i'$, we have $R_t^* \models u_{i+1} \approx u_i$, so we conclude $R_t^* \models u \approx a$ and hence $R_t^* \not\models u \not\approx a$.

To conclude the induction case, consider $\mu(w)$, and observe that we have two possibilities: first, $\mu(w) \in \Gamma_w^{\mathsf{Rt}}$. In this case, suppose $R_t^* \models \mu(w)$; we have already shown that $R_t^* \models \Gamma_v^{\mathsf{Rt}} \backslash \mu(w)$, and we have by hypothesis that $R_t^* \models \Gamma_w \sigma_t$, so we conclude $R_t^* \models \Delta_w \sigma_t \vee \Delta_w^{\mathsf{Rt}}$; however, by hypothesis we have $R_t^* \not\models \Delta_w \sigma_t$, and we have already shown $R_t^* \not\models \Delta_w^{\mathsf{Rt}}$, and hence we reach a contradiction, so $R_t^* \not\models \mu(w)$. In the second case, $\mu(w) \in \Delta_w^{\mathsf{Rt}}$. We have already shown that $R_t^* \models \Gamma_v^{\mathsf{Rt}}$, and we have by hypothesis that $R_t^* \models \Gamma_w \sigma_t$, so we conclude $R_t^* \models \Delta_w \sigma_t \vee \Delta_w^{\mathsf{Rt}}$; however, by hypothesis

we have $R_t^* \not\models \Delta_w \sigma_t$, and we have already shown $R_t^* \not\models \Delta_w^{\mathsf{Rt}} \backslash \mu(w)$, which then implies $R_t^* \models \mu(w)$.

If we apply our result to the root node $w_0$, since $\mu(w_0) \notin \Gamma_{w_0}^{\mathsf{Rt}}$ and $\mu(w_0) \notin \Delta_{w_0}^{\mathsf{Rt}}$, we have that $R_t^* \models \Gamma_v^{\mathsf{Rt}}$ and $R_t^* \not\models \Delta_v^{\mathsf{Rt}}$. □

The following auxiliary lemma illustrates a sufficient condition for the collapse of a "chain" of synthetic constants.

**Lemma 27.** *Given $o_\rho$, let $n = |\rho|$, and define $o_i$ with $1 \leq i \leq n$ as in Lemma 25, and $o_0 = o$. If $o_\rho$ appears in $R_t^*$, but is irreducible w.r.t. $R_t^*$, then for every $0 \leq i < n$, $o_i$ is irreducible w.r.t. $R_t^*$.*

*Proof.* We proceed by reverse induction on $i$ for $1 \leq i \leq n$. The base case $i = n$ is true by hypothesis. Now, consider some $0 \leq i < n$, and assume that $o_j$ is irreducible w.r.t. $R_t^*$ for each $j$ with $i + 1 \leq j \leq n$. We now prove that $o_i$ is also irreducible w.r.t. $R_t^*$.

Suppose, for the sake of a contradiction, that $o_i$ is reducible w.r.t. $R_t^*$. Since $R_t^*$ mentions $o_n$, there must be a generative clause of the form $\Gamma \to \Delta \vee L$ where $L$ mentions $o_n$, and therefore there must be a corresponding non-ground clause $\Gamma' \to \Delta' \in \mathcal{S}_v$ where $\Delta'$ mentions $o_n$. We are in the conditions of Lemma 25, so we consider all sets of clauses as defined in that lemma.

Consider clause $C_{a_i}$ and its corresponding context $v_{a_i}$. Now, since $o_i$ is not irreducible w.r.t. $R_t^*$, due to the definition of $>_t$, by Lemma 14, we have that $o_i \approx a \in \Gamma_{\mathsf{Rt}}$ for some $a \in \Sigma_u$. Arguing as in the proof of condition L7 in Section 6.3.4.2 we can conclude that there is a clause $o_i \approx a \to o_i \approx a \,\hat{\in}\, \mathcal{S}_{v_{a_i}}$, and by Condition L6, we can assume w.l.o.g. that $o_i \approx a \to o_i \approx a \in \mathcal{S}_{v_{a_i}}$. Since $\mathcal{D}$ is saturated and hence $\mathsf{Eq}$ is not applicable to that clause and $C_{a_i}$, we have $\Gamma \wedge o_i \approx a \to \Delta \vee S(a, x) \,\hat{\in}\, \mathcal{S}_{v_{a_i}}$.

We now consider the inferences from $C_{a_i+1}$ to $C_{b_i}$. By Lemma 24, and the fact that none of the atoms in the head of $C_{b_i}$ can be bigger than $S(a, x)$ due to the way Definition 5 is defined, we have that there exists a clause $\Gamma' \wedge o_i \approx a \to \Delta' \vee S(a, x) \,\hat{\in}\, \mathcal{S}_{v_{a_i}}$, with $\Delta' \not>_{v_{a_i}} S_i(a, x)$, where $\Gamma'$ and $\Delta'$ are contained, respectively, in the body and head of a clause $C_j$ between $C_{a_i+1}$ and $C_{b_i}$. Since $\mathcal{D}$ is saturated, rule $\mathsf{Nom}$ is not applicable, so there is a clause $\Gamma' \wedge o_i \approx a \to \Delta' \vee x \approx a_1 \vee \cdots \vee x \approx a_l \,\hat{\in}\, \mathcal{S}_{v_{a_i}}$, where if $a$ is of the form $e_{\rho'}$, then $a_1, a_2, \cdots$ are of the forms $e_{\rho'.S_i^1}$, $e_{\rho'.S_i^2}$, $\cdots$, respectively.

Now observe that $C_{b_i+1}$ will have $L_{b_i+1}$ equal to $f(x) \approx o_{i+1}$, where $f$ is the label of the edge used for the $\mathsf{Pred}$ inference in step $b_i$. Since $L_{a_{i+1}}$ is of the form $S(o_{i+1}, x)$, we have that there exists $k$ with $a_{i+1} > k > b_i + 1$ where $L_k = f(x) \approx o_{i+1}$ and the inference step $k$ selects this literal for application of $\mathsf{Eq}$. We now consider

the inferences from $C_j$ to $C_{b_{i+1}}$, and by Lemma 24, and the fact that none of the literals in the head of $C_{b_{i+1}}$ can be bigger than a literal of the form $f(x) \approx a'$, with $a' \in \{a_1, \cdots, a_l\}$, we conclude that

$$o_i \approx a \wedge \Gamma'_{k'} \rightarrow \Delta'_{k'} \vee o_{i+1} \approx a_1 \vee \cdots \vee o_{i+1} \approx a_l \,\hat{\in}\, \mathcal{S}_{v_{a_{i+1}}},$$

where $\Gamma'_{k'} \subseteq \Gamma_{k'}$, and $\Delta'_{k'} \subseteq \Delta_{k'}$ for some clause $C_{k'}$ with $k \leq k' \leq b_{i+1}$.

Consider now a ground decomposition of this clause, $\langle\langle\{w\}, \emptyset\rangle, \lambda, \mu\rangle$, where $\lambda(w)$ is equal to this clause with $\Gamma_w^{\mathsf{Rt}} = \top$, $\Delta_w^{\mathsf{Rt}} = \bot$, and $\mu(w)$ is chosen arbitrarily. Consider all inferences from $C_{k'}$ to $\Gamma' \rightarrow \Delta'$ in context $C_m$. Since for any ground atom selected for an inference rule we have $L \rightarrow L$ contained u.t.r. in the corresponding context, we can use the same inferences sequentially on $\langle\langle\{w\}, \emptyset\rangle, \lambda, \mu\rangle$, considering always the natural successor whenever possible, and the alternative successor otherwise. By Lemma 24, we obtain that all clauses in the resulting ground decomposition of $C_m$ are contained u.t.r. in $\mathcal{S}_v$.

Now, we have that $R_t^* \not\models \Delta$ by Lemma 12 and by Lemma 21 we have $R_t^* \not\models \Delta'\sigma_t \backslash \Delta$; furthermore, we have $R_t^* \models \Gamma'\sigma_t$ by Lemma 21, and $R_t^* \models o_i \approx a$ by hypothesis, so we can apply Lemma 26 and conclude that $R_t^* \models o_{i+1} \approx a'_1 \vee \cdots o_{i+1} \approx a'_l$, which contradicts our induction hypothesis that $o_{i+1}$ is irreducible, and concludes the proof of the lemma.

□

Finally, the last lemma in this section complements the previous lemma and shows that if two clauses of the form in (8) corresponding to different elements in a nominal "chain" appear in the same context and have the same type, then the chain must collapse.

**Lemma 28.** *Suppose there is a clause $\Gamma \rightarrow \Delta \in \mathcal{S}_v$ mentioning a constant $o_\rho$ such that $\Gamma\sigma_t \rightarrow \pi_t(\Delta_t\sigma_t) \in N_t$. Let $n = |\rho|$, and consider all clauses defined in this case by Lemma 25. Let $i$ and $j$ with $1 \leq i < j \leq n$ be such that clauses $C_{b_i}$ and $C_{b_j}$ appear in the same context of $\mathcal{D}_{b_i}$ and $\mathcal{D}_{b_j}$, respectively, and they have the same type. Then, we have that $o_\rho$ is not irreducible w.r.t. $R_t^*$.*

*Proof.* To prove the lemma, consider the biggest constant $o_j$ in $\Sigma_u$ for which this occurs. By property P3, we have that there exist $\Gamma_{b_i} \rightarrow \Delta_{b_i} \vee x \approx o_i$ and $\Gamma_{b_j} \rightarrow \Delta_{b_j} \vee x \approx o_j$ in context $\mathcal{S}_v$ of $\mathcal{D}$ which subsume $C_{b_j}$ and $C_{b_i}$, respectively. Let $w$ be the context where clause $C_{b_j}$ is propagated in induction step $b_j$, and let $\Gamma_{b_{j+1}} \rightarrow \Delta_{b_{j+1}} \vee f(x) \approx o_j \vee \Psi_{j+1}$, where $\Psi_{j+1} = \Delta_{b_j}\{x \mapsto f(x), y \mapsto x\}$, be the conclusion of

the inference by $\mathsf{Pred}$ at step $b_j$ of the derivation, where $f$ is the label corresponding to the edge used in the inference. Notice that by hypothesis $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee [f(x) \approx o_j] \vee \Psi_{j+1} \hat{\in} \mathcal{D}$.

Now, we have that since $\mathcal{D}$ is saturated, and since $\Gamma_{b_i} \to \Delta_{b_i} \vee x \approx o_i$ and $C_{b_j}$ have the same type, we are in the preconditions of the $\mathsf{Pred}$ rule with $C_{b_i}$ as the main premise, so by Lemma 24, there exists a clause $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee \Psi_i \hat{\in} \mathcal{S}_w$, with $\Psi_i = \Delta_{b_i} \backslash \mathsf{Pr}_\tau \vee x \approx o_i\{x \mapsto f(x), y \mapsto x\}$. The head of this clause cannot contain a tautology and must contain at least one literal of $\Psi_i$, due to its form and the fact that $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee [f(x) \approx o_j] \vee \Psi_{j+1} \hat{\in} \mathcal{D}$. Notice that $\Psi_i$ must be of the form $f(x) \approx u_1 \vee \cdots \vee f(x) \approx u_l$, with $u_k \in \Sigma_u$ for $1 \leq k \leq l$.

Consider the inference steps from $b_{j+1}$ to $d_j$, the inference step where literal $f(x) \approx o_j$ is selected; we know such inference step must exist since a descendant literal of $f(x) \approx o_j$ is of the form $S_j(o_j, x)$. The premise in this step is a clause $\Gamma_{d_j} \to \Delta_{d_j} \vee f(x) \approx o_j$, with $\Gamma_{d_j} \to \Delta_{d_j} \vee [f(x) \approx o_j] \hat{\in} \mathcal{S}_w$. Considering the premises used in these inference steps and $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee \Psi_i$, by Lemma 24, there exists a clause $\Gamma'_{d_j} \to \Delta'_{d_j} \vee o_j \approx u_1 \vee \cdots \vee o_j \approx u_l \hat{\in} \mathcal{S}_w$, with no tautologies in the head and at least one literal of $o_j \approx u_1 \vee \cdots \vee o_j \approx u_l$ in the corresponding subsuming clause, since $\Gamma_{d_j} \to \Delta_{d_j} \vee [f(x) \approx o_j] \hat{\in} \mathcal{S}_w$.

Now, we can proceed as in the proof of Lemma 27, creating a decomposition for $\Gamma'_{d_j} \to \Delta'_{d_j} \vee o_j \approx u_1 \vee \cdots \vee o_j \approx u_l$ and performing on this decomposition the inferences from $C_{d_j}$ until original clause $\Gamma \to \Delta$; using an analogous argument, we conclude $R_t^* \models o_j \approx u_1 \vee \cdots \vee o_j \approx u_l$. Without loss of generality, suppose $R_t^* \models o_j \approx u_1$. If $u_1 \succcurlyeq o_j$, we contradict the initial hypothesis that $o_j$ is the biggest constant for which the conditions of the lemma hold, and if $o_j \succcurlyeq u_1$, then by Lemma 27, we have that $o_\rho$ is not irreducible.

□

#### 6.3.3.4 Main proof

In this section we use the lemmas from above to show that Lemma 22 holds in the case $|\rho| = \Lambda$, and then we use this result to prove Lemma 22 in general.

**Lemma 29.** *For any DL-clause of the form DL4 and any $o_\rho \in \Sigma_u$ irreducible w.r.t. $R_t$, where $|\rho| = \Lambda$, and $t$ is irreducible w.r.t. $R_t$ with $t \neq f(o_\rho)$ for some $f \in \Sigma_f$, we have that $R_t^* \not\models S_{B_2}(o_\rho, t)$.*

*Proof.* We use proof by contradiction. Consider the greatest constant $o_\rho$ for which this occurs. Since both $t$ and $o_\rho$ are irreducible, there exists a generative clause

$\Gamma^i \rightarrow \Delta^i \vee S(o_\rho, t)$. This implies the existence of a clause $C \in \mathcal{S}_v$ of the form $\Gamma' \rightarrow \Delta'_1 \vee \Delta'_2$ with $\Gamma'\sigma_t = \Gamma^i$, $\Delta'_1\sigma_t \subseteq \Delta^i \cup \Delta_t$ and $\Delta'_2\sigma_t = S_{B_2}(o_\rho, t)$. Now, notice that we are in the conditions of Lemma 5 so $\Delta'_1 \not\succ_v \Delta'_2$. Furthermore, since $t' \neq o_\rho$, Lemma 6 ensures that $\Delta'_2 = S_{B_2}(o_\rho, x)$.

Let $\Gamma'_\tau = \Gamma' \cap \mathsf{Su}_\tau$, and $\Delta'_\tau = \Delta'_1 \cap \mathsf{Pr}_\tau$. Notice that we are in the conditions of Lemma 25, so there exist $\Lambda$ clauses $C_{b_1}, \cdots, C_{b_\Lambda}$ satisfying the properties of that lemma contained u.t.r. in $\mathcal{D}$. Furthermore, since $o_\rho$ is not irreducible, and we are in the conditions of Lemma 25, and this in turn, allows us to apply Lemma 28, so we have that no more than $\tau$ clauses can be contained u.t.r. in each context, so by the pigeonhole principle, there exist $\tau$ such clauses contained u.t.r. in $\mathcal{S}_v$, and each of these clauses has a different type. Thus, there exists some $1 \leq j \leq \Lambda$ where $C_j$ has the same type and is contained u.t.r. in $\mathcal{S}_v$. Let us re-name this clause to $C_b$, and write it as $\Gamma_b \rightarrow \Delta_b \vee x \approx u_1 \vee \cdots \vee x \approx u_n \vee y \approx u'_1 \vee \cdots \vee y \approx u'_{n'}$, with $u_l \in \Sigma_u$ for $1 \leq l \leq n$, and $u'_l \in \Sigma_u$ for $1 \leq l \leq n'$.

Consider a ground decomposition of this clause $\langle\langle\{w\}, \emptyset\rangle, \lambda, \mu\rangle$, where $\lambda(w)$ is equal to this clause, $\Gamma_w^{\mathsf{Rt}}$ contains all ground atoms not in the body of $\Gamma'$, $\Delta_w^{\mathsf{Rt}}$ contains all ground atoms not in the head of $\Delta'$, and $\mu(w)$ is chosen arbitrarily from $\Delta_w$. Consider all inferences from $C_b$ to $\Gamma' \rightarrow \Delta'_1 \vee S_{B_2}(o_\rho, x)$ in the derivation of $\mathcal{D}$. Since for any ground atom selected for an inference rule we have $L \rightarrow L$ contained u.t.r. in the corresponding context, we can use the inferences from $C_b$ to $\Gamma' \rightarrow \Delta'_1 \vee S_{B_2}(o_\rho, x)$ to sequentially compute successor ground decompositions from $C_b$ and $\langle\langle\{w\}, \emptyset\rangle, \lambda, \mu\rangle$, choosing always the natural successor except when not possible or when a literal from $\Gamma_w^{\mathsf{Rt}}$ or $\Delta_w^{\mathsf{Rt}}$ is selected, in which case we consider the alternative successor. By Lemma 24, we obtain that all clauses in the resulting ground decomposition for $\Gamma' \rightarrow \Delta'_1 \vee S_{B_2}(o_\rho, x)$ are contained u.t.r. in $\mathcal{S}_v$.

By Lemma 26 we conclude that every ground literal in $\Gamma_b$ not in $\Gamma'$ is satisfied by $R_t^*$, and every ground literal in $\Delta_b$ not in $\Delta'_1$ is satisfied by $R_t^*$. Furthermore, if $L$ is a non-ground literal in $\Gamma_b$, it is in $\mathsf{Su}_\tau$, and hence we have $L \in \Gamma'$, because we chose $C_b$ to have the same type as $\Gamma' \rightarrow \Delta'_1 \vee S_{B_2}(o_\rho, x)$, and since $R_t^* \models \Gamma'\sigma_t$ by Lemma 21, we have $R_t^* \models L\sigma_t$. Similarly, if $L$ is a non-ground literal in $\Delta_b$, and since $C_b$ is a premise for a $\mathsf{Pred}$ rule, we have $L \in \mathsf{Pr}_\tau$, so we have $L \in \Delta'_1$ since $C_b$ has the same type as $\Gamma' \rightarrow \Delta'_1 \vee S_{B_2}(o_\rho, x)$. Thus, the fact that $R_t^* \not\models \Delta^i$ by Lemma 12 and $R_t^* \not\models \Delta'_1\sigma_t \backslash \Delta^i$ by Lemma 21 imply $R_t^* \not\models L\sigma_t$.

Thus, we have that $R_t^* \models \Gamma_b\sigma_t$, which implies $R_t^* \models \Delta_b\sigma_t \vee t \approx u_1 \vee \cdots \vee t \approx u_n \vee t' \approx u'_1 \vee \cdots \vee t' \approx u'_{n'}$, or the corresponding clause without literals $t' \approx u'_l$ if $t'$

does not exist. However, we also showed $R_t^* \not\models \Delta_b\sigma_t$, so we conclude

$$R_t^* \models t \approx u_1 \vee \cdots \vee t \approx u_n \vee t' \approx u'_1 \vee \cdots \vee t' \approx u'_{n'},$$

or the corresponding clause without equalities for $t'$ if $t$ has no predecessor. This contradicts the fact that $t$ is irreducible, which concludes the proof. □

We are now ready to prove Lemma 22 in the general case.

**Lemma 22.** *Suppose $t$ is irreducible w.r.t. $R_t$. For any DL-clause of the form DL4 and $u \in \Sigma_u$ irreducible w.r.t. $R_t$, if $t = c$ or $t = f(t')$ for some $t' \neq u$ with $f \in \Sigma_f$, then $R_t^* \not\models B_1(u) \wedge S_{B_2}(u,t)$.*

*Proof.* We prove this by contradiction. Suppose $R_t^* \models B_1(u)$. If $t = c$, since $u$ is irreducible w.r.t. $R_t$, there must be a generative clause $C^j$ of the form $\Gamma^j \to \Delta^j \vee B_1(u)$, with $B_1(u) >_t \Delta_j$, $\Gamma^j \subseteq \Gamma_t$, and $R_t \not\models \Delta^j$ due to Lemma 12. Notice that since $t = c$, we are in the conditions of Lemma 6 and there is a clause $\Gamma'' \to \Delta'' \vee B_1(u)$, with $\Gamma''\sigma_t = \Gamma^j$, $\Delta''\sigma_t \subseteq \Delta_j \cup \Delta_t$, and we have that if $A \in \Delta'' \succ_{X_t} B_1(u)$, then $A \in \Delta_Q$ because $B_1(u) >_t \Delta_j$, but by condition C2 of Theorem 2, we have that no element of $A$ can be greater than $B_1(u)$, and hence we conclude $\Delta'_1 \not\succ_{X_{t'}} B_1(u)$. If $t \neq c$, then since $u$ is irreducible w.r.t. $R_t$, Lemma 14 ensures $B_1(u)$ is also irreducible w.r.t. $R_c$, and hence $B_1(u) \in \Gamma_{\mathsf{Rt}}$, so by Condition L7 we have $B_1(u) \to B_1(u) \,\hat{\in}\, \mathcal{S}_{X_t}$. Notice once again that by condition L6, $B_1(u) \to B_1(u) \,\hat{\in}\, \mathcal{S}_{X_t}$ implies $B_1(u) \to B_1(u) \in \mathcal{S}_{X_t}$ or $\top \to B_1(u) \in \mathcal{S}_{X_t}$. In this case, let us define $\Gamma'' = B_1(u)$ or $\top$, accordingly, and $\Delta'' = \bot$;

Suppose also that $R_t^* \models S_{B_2}(u,t)$. Then, there exists a generative clause $C^i$ such that $L^i$ is of the form $S_{B_2}(u,t)$ with $u \in \Sigma_u$ irreducible, $t \neq f(u)$. Then, there exists a clause $C \in \mathcal{S}_v$ of the form $\Gamma' \to \Delta'_1 \vee \Delta'_2$ with $\Gamma'\sigma_t = \Gamma^i$, $\Delta'_1\sigma_t \subseteq \Delta^i \cup \Delta_t$ and $\Delta'_2\sigma_t = S_{B_2}(u,t)$. Now, notice that we are in the conditions of Lemma 5 so $\Delta'_1 \not\succ_v \Delta'_2$. Furthermore, since $t' \neq u$, Lemma 6 ensures that $\Delta'_2 = S_{B_2}(u,x)$.

Suppose the depth of $u$ is smaller than $\Lambda$. Since the context structure is saturated and the preconditions of the rule $\mathsf{Nom}$ are satisfied, we have that $\Gamma' \wedge \Gamma'' \to \Delta' \vee \Delta'' \bigvee_{j=1}^{N} x \approx u_j \,\hat{\in}\, \mathcal{S}_v$. Then, by Lemma 7, we have $\Gamma^i \wedge \Gamma^j \to \Delta_i \vee \Delta_j \,\hat{\in}\, N_t$ if $t \neq c$, since atoms of the form $t \approx u_j$ are in $\Delta_t$; and $\Gamma^i \wedge \Gamma^j \to \Delta_i \vee \Delta_j \vee \bigvee_{j=1}^{N} t \approx u_j \,\hat{\in}\, N_t$ otherwise. Since $C^j$ and $C^i$ are generative, Lemma 12 ensures $R_t^* \not\models \Delta^j$ and $R_t^* \not\models \Delta^i$; furthermore, in case $t = c$ we have $R_t^* \not\models t \approx u$ for any $u \in \Sigma_u$ since $t \neq u$ and $t$ is irreducible w.r.t. $R_t^*$ by hypothesis. But then we contradict Lemma 8, and hence we conclude the proof.

Finally, if the depth of $u$ is equal to $\Lambda$, then the result follows by Lemma 29. □

### 6.3.4 The Composite Model

In this section we show how to combine the fragments into a single interpretation $R^*$, and then we show that $R^*$ is a model of $\mathcal{O}$ that it disproves the query. In Section 6.3.4.1 we describe the unfolding strategy, which determines which $\mathsf{a}$-terms are included in the model and the parameters for the fragment-building method; in Section 6.3.4.2 we prove that the prerrequisites from Section 6.3.2.1 are satisfied at the time of building each fragment; finally in Section 6.3.4.3 we prove all relevant properties of the model, including the claims that it satisfies $\mathcal{O}$ and that it disproves the target query $\Gamma_Q \to \Delta_Q$.

#### 6.3.4.1 Construction of the Model

In this section we build the composite model by combining the fragments obtained from the procedure in Section 6.3.2. To do this, we use well-founded induction on HU. First we construct a "root fragment" $R^*_c$ which determines the function-free, ground atoms of our model. Then, we inductively build the fragments corresponding to elements of $\Sigma_u$ and successors of elements mentioned in previous fragments:

- If $R^*_c$ has been defined, then for any $u \in \Sigma_u$, we can attempt to construct the fragment $R^*_u$.
- If $R^*_t$ is defined for a term $t$, then for any $f \in \Sigma_f$, we can attempt to construct the fragment $R^*_{f(t)}$.

Thus, when defining a fragment $R^*_t$ for a term $t \neq c$, we assume that the fragment $R^*_c$ has already been defined and it satisfies all lemmas from the previous section. Similarly, when defining the fragment $R^*_{f(t)}$ for a term of the form $f(t)$, we assume that the fragment $R^*_t$ has already been defined and satisfies all lemmas from the previous section. Furthermore, for the construction of fragments for $t \neq c$, we use a fixed interpretation $R^*_{\mathsf{Rt}}$, a conjunction $\Gamma_{\mathsf{Rt}}$, and a disjunction $\Delta_{\mathsf{Rt}}$ defined as follows: let $R_{\mathsf{Rt}}$ be the rewrite system consisting of the rewrite rules $l \Rightarrow s$ for each $l \Rightarrow s \in R_c$ such that $l$ and $s$ are root terms; this rewrite system defines $R_{\mathsf{Rt}}$. We then define $\Gamma_{\mathsf{Rt}}$ as the set of equalities corresponding to rules in $R_c$, and $\Delta_{\mathsf{Rt}}$ as the subset of $\mathsf{Rt}$ which is satisfied by $R^*_{\mathsf{Rt}}$.

We next define partial functions $X$, $\Gamma$ and $\Delta$, which map each term $t$ to a context $X_t \in \mathcal{V}$, a set of equalities $\Gamma_t$, and a set of literals $\Delta_t$:

- For $c$, we define $X_c = q$, $\Gamma_c = \Gamma_Q\sigma_c$ and $\Delta_c = \Delta_Q\sigma_c$.

- For each $t \in \Sigma_u$, if $R_c^* \not\models t \approx u$ for some $u \in \Sigma_u$ such that $t \succcurlyeq u$, then define $X_t = v_r$, $\Gamma_t = \Gamma_{\mathsf{Rt}}$, and $\Delta_t = \Delta_{\mathsf{Rt}}$. Otherwise, $R_t$ is undefined.

- For any other $t$, the term is of the form $f(t')$ for some $f \in \Sigma_f$. We then distinguish several cases:

  - If $t$ does not occur in $R_{t'}$, then define $R_t$ as $\{t \Rightarrow c\}$.
  - If $t$ occurs in $R_{t'}$ but is not irreducible w.r.t. $R_{t'}$, then $R_t$ is undefined.
  - If $t$ occurs in $R_{t'}$ and is irreducible w.r.t. $R_{t'}$, then by construction of $R_{t'}$ we have that $t$ appears in literal $l^i \approx r^i$ in some generative clause $\Gamma^i \to \Delta^i \vee l^i \approx r^i$; the fact that $t$ is irreducible implies that if $t' = g(t'')$ for some $g \in \Sigma_f$, then $L^i$ does not contain $t''$. Let $w = X_{t'}$; since $L^i$ contains $f(t')$ for some $f \in \Sigma_f$, we are in the conditions of Lemma 5 as well as lemma Lemma 6, and hence there exists a clause $\Gamma \to \Delta_1 \vee l \approx r$ in $\mathcal{S}_w$, with $\Gamma\sigma_{t'} = \Gamma_{t'}$, $\Delta\sigma_{t'} \subseteq \Delta^i \cup \Delta_{t'}$ and $l\sigma_t = l^i$ and $r\sigma_t = r^i$, and $\Delta \not\succ_w l \approx r$. Furthermore, either $f(x)$ or $f(u)$ to appear in $l \approx r$ because $t$ appears in $l^i \approx r^i$. Since, the $\mathsf{Succ}$ rule is not applicable to $\Gamma \to \Delta_1 \vee l \approx r$, there must be an context $v$ such that $\langle w, v\rangle$ is labelled by $f$, we have $A \to A \,\hat{\in}\, \mathcal{S}_v$ for each $A \in K_2$ (hence $l \approx r \to l \approx r \,\hat{\in}\, \mathcal{S}_v$), and $\mathsf{core}_v \subseteq K_1$, for $K_1$ and $K_2$ defined as in the $\mathsf{Succ}$ rule. We choose $X_t = v$, and then define $\Gamma_t = R_t^* \cap \mathsf{Su}_t$, and $\Delta_t = \Delta_{\mathsf{Rt}} \cup (\mathsf{Pr}_t \backslash R_{t'}^*)$.

Next we define the composite rewrite system $R$. Let $R_t^u$ be the set of rules in $R_t$ of the form $u_1 \Rightarrow u_2$ for $u_1, u_2 \in \Sigma_u$. If $c$ is irreducible w.r.t. $R_c$, then $R = R_c \cup \bigcup_{t \neq c} (R_t \backslash R_t^u)$, where the union ranges over all fragments that have been defined. If $c$ is not irreducible w.r.t. $R_c$, we define $R = R_c^u \cup \bigcup_{t \neq c} (R_t \backslash R_t^u)$. We highlight two aspects of this definition: first, notice that we remove the rules in $R_t^u$ from the general system; this is to ensure that $R$ is left-reduced, and therefore Church-Rosser (see Lemma 32). Furthermore, this leaves $R^*$ unaffected, since $R_c^* \models (R_t^u)^*$. The second observation is that we exclude $R_c$ (other than rules in $R_c^u$) from $R$ when $c$ is not irreducible w.r.t. $R_c$; this is because we will use $u$ to disprove the main query $\Gamma_Q \to \Delta_Q$, where $u$ is the normal form of $c$ w.r.t. $R_c$. Removing $R_c$ will not be a problem since the definition of $\Gamma_{\mathsf{Rt}}$ and $\Delta_{\mathsf{Rt}}$ will ensure that $R_u^*$ disproves $\Gamma_Q \to \Delta_Q$ (see Lemma 38).

#### 6.3.4.2 Satisfaction of Prerequisites

We show next that during the construction of the composite model as described in the previous section, for every $t$, $X_t$, $\Gamma_t$, and $\Delta_t$, the assumptions listed in Section 6.3.2.1 are satisfied. Thus, we assume that one of the following is the case:

(i) $t = c$

(ii) $t \in \Sigma_u$ with $R_c^* \not\models t \approx u$ for every $u \in \Sigma_u$ with $t \succcurlyeq u$

(iii) $t = f(t')$ for some $f \in \Sigma_f$, and $t$ occurs in $R_{t'}$ but is irreducible by it; furthermore, $X_{t'}$ and $X_t$ are connected via an edge labelled $f$, $A \to A\,\hat{\in}\,\mathcal{S}_v$ for each $A \in K_2$, and $\mathsf{core}_{X_t} \subseteq K_1$, with $K_1$ and $K_2$ defined as in the $\mathsf{Succ}$ rule for $X_{t'}$, and such that if $X_{t'} = v_r$, then $u = t'$.

First, for $t \neq c$, we notice that since $R_c$ is Church-Rosser, then $R_{\mathsf{Rt}}$ is also Church-Rosser, because it is a subset of $R_c$. Furthermore, the rewrite rules in $R_c$ are of the form $B(u) \Rightarrow \mathsf{true}$, $S(u_1, u_2) \Rightarrow \mathsf{true}$, or $u_1 \Rightarrow u_2$, as expected. Finally, $\Gamma_{\mathsf{Rt}}$ contains $s \approx l$ for each $s \Rightarrow l \in R_{\mathsf{Rt}}$, and it is verified that $\Delta_{\mathsf{Rt}}$ contains literals exclusively in $\mathsf{Rt}$. Notice that the definition of $\Delta_{\mathsf{Rt}}$ straightforwardly ensures conditions L1 and L2 are satisfied.

Condition L3 is trivially true by definition of $\Gamma_t$ and $\Delta_t$. For condition L4, if $t \neq c$, we have that $\Delta_{\mathsf{Rt}} \subseteq \Delta_t$ is true by definition if $t \neq c$. Similarly, $\Gamma_{\mathsf{Rt}} \subseteq \Gamma_t$ is true by definition if $t \in \Sigma_u$. Otherwise, given any $L \in \Gamma_{\mathsf{Rt}}$ we have $L \in \mathsf{Su}$, so $L \in \mathsf{Su}_t$. Then, if $t' = c$, we have $R_{t'}^* \models \Gamma_{\mathsf{Rt}}$ by definition of $\Gamma_{\mathsf{Rt}}$, and otherwise by induction hypothesis we have $R_{t'}^* \models \Gamma_{\mathsf{Rt}}$ by Lemma 18. Thus, $\Gamma_{\mathsf{Rt}} \subseteq R_{t'}^* \cap \mathsf{Su}_t = \Gamma_t$. Finally, consider some $L \in \Delta_t \cap \mathsf{Rt}$ such that $L \notin \Delta_{\mathsf{Rt}}$. Since $L \notin \Delta_{\mathsf{Rt}}$, by condition L2, $R_{\mathsf{Rt}}^* \models L$, and by Lemma 18 we have $R_{t'}^* \models L$. However, by definition of $\Delta_t$, if $L \notin \Delta_{\mathsf{Rt}}$ then $R_{t'}^* \not\models L$, which contradicts our previous claim.

To see why condition L5 is satisfied, we consider the two cases covered by this condition:

- Case 1: suppose $t \in \Sigma_u$ but there exists $u \in \Sigma_u$ with $t \succcurlyeq u$ such that $t \approx u \notin \Delta_t$. Since $t, u \in \Sigma_u$, we must have that $t \approx u \in R_{\mathsf{Rt}}^*$, for otherwise we have $t \approx u \in \Delta_{\mathsf{Rt}}$ by L1, and $\Delta_{\mathsf{Rt}} \subseteq \Delta_t$ by condition L4, so we obtain a contradiction. But then, by definition of $R_{\mathsf{Rt}}$, $R_c^* \models t \approx u$, and since $t \succcurlyeq u$, we reach a contradiction with (ii).

- Case 2: suppose $t = f(t')$ but the condition is not satisfied, so there are three possible sub-cases:

– Case 2.a : there exists $u \in \Sigma_u$ with $t \succcurlyeq u$ such that $t \approx u \notin \Delta_t$. Since $t \notin \Sigma_u$, we have $t \approx u \notin \mathsf{Rt}$ and hence $t \approx u \notin \Delta_{\mathsf{Rt}}$; we also have $x \approx u \in \mathsf{Pr}$ and hence $t \approx u \in \mathsf{Pr}_t$. Thus, by definition of $\Delta_t$, we conclude $R^*_{t'} \models t \approx u$. But this contradicts (iii), since $t$ is then not irreducible w.r.t. $R^*_{t'}$.

– Case 2.b : there exists $u \in \Sigma_u$ with $t' \succcurlyeq u$ such that $t' \approx u \notin \Delta_t$. To reach a contradiction, if $t' \in \Sigma_u$, we proceed exactly as in Case 1, and otherwise (i.e. $t' \notin \Sigma_u$), we proceed as in Case 2.a, since $y \approx u \in \mathsf{Pr}$.

– Case 2.c : $t \approx t' \notin \Delta_t$. Here, we only need to note that $x \approx y \in \mathsf{Pr}$ and proceed as in Case 2.a.

To show that condition L6 holds, we consider the different forms of $t$ and reach a contradiction for each of them

- If $t = c$, then the condition is satisfied by our assumption that condition C1 holds. To see this, suppose $\Gamma_c \to \bot \,\hat{\in}\, N_c$. Thus, it must be the case that $\Gamma'_c \to \bot \in N_c$ for some conjunction $\Gamma'_c$ with $\Gamma'_c \subseteq \Gamma_c$. Hence, there is a clause $\Gamma' \to \Delta' \in \mathcal{S}_q$ with $\Gamma'\sigma_c = \Gamma'_c$ and $\Delta'\sigma_c \subseteq \Delta_c$. By definition of $\Gamma_c$ and $\sigma_c$, any $A \in \Gamma'$ must be of the form $B(x)$ for some $B(x) \in \Gamma_Q$, so $\Gamma' \subseteq \Gamma_Q$. An analogous argument applies to $\Delta'$ with respect to $\Delta_Q$. Therefore, we conclude $\Gamma_Q \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$, a claim that violates the main hypothesis of this completeness proof.

- If $t \in \Sigma_u$, notice we have $\Gamma_t = \Gamma_{\mathsf{Rt}}$, and we must have $\Gamma' \to \bot \in N_t$ for some $\Gamma' \subseteq \Gamma_t$. Thus, there exists a clause $\Gamma' \to \Delta' \in v_r$ with $\Delta' \subseteq \Delta_{\mathsf{Rt}}$. By definition of $\Delta_{\mathsf{Rt}}$, for each $A_i \in \Delta_{\mathsf{Rt}}$ there must be a generative clause $\Gamma_i \to \Delta_i \vee A_i$ in $N_c$ with $A_i >_c \Delta_i$. Thus, by Lemma 6 there must exist a clause $\Gamma'_i \to \Delta'_i \vee A'_i$ in $\mathcal{S}_q$ with $\Gamma'_i\sigma_c = \Gamma'$, $\Delta'_i\sigma_c \subseteq \Delta_i \cup \Delta_c$, and $A'_i\sigma_c = A_i$. Observe that if $L$ is a literal of $\Delta'_i$ and $\Delta_Q$, by condition C2 we have that if $L \succ_q A'_i$, then $A'_i \in \Delta_Q$, which is impossible since $A'_i$ is ground. If $L \in \Delta'_i \backslash \Delta_Q$ then $L\sigma_t \in \Delta_i$, so if $L \succ_q A'_i$ we have $\Delta_i >_c A_i$, and we reach a contradiction. We therefore conclude that $\Delta'_i \not\succ_q A'_i$. Since $A'_i$ is ground and $r$-$\mathsf{Succ}$ is not applicable, for every $u \in \Sigma_u$ mentioned in $A'_i$, there exists an edge $\langle q, v_r, u\rangle$ between $q$ and $v$. But since $\Delta_{\mathsf{Rt}} \subseteq \mathsf{Pr}^r$, with these edges we are in the conditions of the $r$-$\mathsf{Pred}$ rule for $\Gamma' \to \bot$ in $\mathcal{S}_{v_r}$. Hence, we obtain:

$$\bigwedge_{i=1}^{n} \Gamma'_i \to \bigvee_{i=1}^{n} \Delta'_i \vee \Delta' \,\hat{\in}\, \mathcal{S}_q$$

Since $\Gamma_i \to \Delta_i \vee A_i$ is generative for each $i$ with $1 \leq i \leq n$, by Lemma 12, we have $\Gamma_i \subseteq \Gamma_c$ for each $i$ with $1 \leq i \leq n$, and since $\Delta'_i$ and $\Delta'$ are either ground or in $\Delta_Q$, by Lemma 7 we have:

$$\bigwedge_{i=1}^{n} \Gamma_i \to \bigvee_{i=1}^{n} \Delta_i \vee \Delta' \,\hat{\in}\, N_c$$

Furthermore, by Lemma 8, $R_c^* \models \bigvee_{i=1}^{n} \Delta_i \vee \Delta'$. However, the fact that $\Gamma_i \to \Delta_i \vee A_i$ is generative for each $1 \leq i \leq n$, by Lemma 12, we have $R_c^* \not\models \Delta_i$ for each $i$ with $1 \leq i \leq n$. Finally, the fact that $\Delta' \subseteq \Delta_{\mathsf{Rt}}$ implies, by conditions L1 and L2, $R_c^* \not\models \Delta'$. Thus, we reach a contradiction.

- If $t = f(t')$ and $\Gamma_t \to \bot \,\hat{\in}\, N_t$, the set $N_t$ contains a clause of the form:

$$\bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C_i \to \bot \qquad \begin{array}{l} \text{with } C_i \in \Gamma_t \subseteq (R_{t'})^* \cap \mathsf{Su}_t \text{ for each } 1 \leq i \leq n, \\ \text{and } A_i \in \Gamma_{\mathsf{Rt}} \subseteq (R_{t'})^* \cap \mathsf{Su}_t, \text{ for each } 1 \leq i \leq m. \end{array} \tag{10}$$

Hence, $\mathcal{S}_v$ contains a clause of the form:

$$\bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C'_i \to \bigvee_{i=1}^{k} L'_i \qquad \begin{array}{c} \text{with } C_i = C'_i\sigma_t \text{ for each } 1 \leq i \leq m, \\ \text{and } L'_i\sigma_t \in \Delta_t \text{ for each } 1 \leq i \leq k. \end{array} \tag{11}$$

By definition of $\Delta_t$, we have that every $L'_i$ is in $\mathsf{Pr}$ or in $\Delta_{\mathsf{Rt}}$, or of the form $S(x,u)$ or $S(u,x)$ if $t' = u$ for some $u \in \Sigma_u$. Hence, this clause satisfies the preconditions for acting as main premise of a $\mathsf{Pred}$ rule inference. Next, we have that for each $i$ with $1 \leq i \leq n$, $C_i \in \Gamma_t$, and hence $C_i \in \mathsf{Su}_t$, so $C_i$ is either in $\Gamma_{\mathsf{Rt}}$, or is of the form $B(t)$, $S(t,t')$, or $S(t',t)$, with $B \in \Sigma_A$, $S \in \Sigma_S$. If $C_i \in \Gamma_{\mathsf{Rt}}$, then we have two possibilities: either $t' = c$ or $t' \neq c$.

  - In the first case ($t' = c$), by definition of $\Gamma_{\mathsf{Rt}}$ for each $C_i \in \Delta_{\mathsf{Rt}}$ there must be a generative clause $\Gamma_i \to \Delta_i \vee C_i$ in $N_c$ with $C_i >_c \Delta_i$. By Lemma 6 there must exist a clause $\Gamma'_i \to \Delta'_i \vee C_i$ in $\mathcal{S}_q$ with $\Gamma'_i\sigma_c = \Gamma'$, $\Delta'_i\sigma_c \subseteq \Delta_i \cup \Delta_c$. If $L$ is a literal of $\Delta'_i$ and $\Delta_Q$, by condition C2 we have that if $L \succ_q C_i$, then $C_i \in \Delta_Q$, which is impossible since $C_i$ is ground. If $L \in \Delta'_i \backslash \Delta_Q$ then $L\sigma_t \in \Delta_i$, so if $L \succ_q C_i$ we have $\Delta_i >_c C_i$, and we reach a contradiction. To uniformise the nomenclature, for this $i$ we define $C'_i$ as $C_i$.

– In the second case ($t' \neq c$), by condition L7, for each $C_i \in \Delta_{\mathsf{Rt}}$, there exists a clause $C_i \to C_i \,\hat{\in}\, \mathcal{S}_{X_{t'}}$, and by condition L6 we can assume, without loss of generality, that $C_i \to C_i \in \mathcal{S}_{X_{t'}}$. To uniformise the nomenclature, for this $i$ we define $\Gamma'_i$ and $C'_i$ as $C_i$.

If $C_i$ is of the form $B(t)$, $S(t,t')$, or $S(t',t)$, we have that $t'$ is irreducible by $R_{t'}$ due to Lemma 15, and $t$ is also irreducible by $R_{t'}$ due to (iii). Therefore, there exists a generative clause:

$$\Gamma_i \to \Delta_i \vee C_i \in N_{t'} \tag{12}$$

with $C_i >_t \Delta_i$ and $\Gamma_i \subseteq \Gamma_{t'}$. By definition of $N_{t'}$, for each such clause, there must be a clause in $\mathcal{S}_{X_{t'}}$ of the form:

$$\Gamma'_i \to \Delta'_i \vee C'_i \tag{13}$$

with $\Gamma_i = \Gamma'_i\sigma_{t'}$, $\Delta'_i\sigma_t \subseteq \Delta_i \cup \Delta_{t'}$ and $C_i = C'_i\sigma_{t'}$. Since $C_i$ contains $t$ and does not contain $t''$ (if it exists), we are in the conditions of Lemmas 5 and 6 and hence we have $\Delta'_i \not>_{X_{t'}} C'_i$.

We have therefore shown that for each $i$ with $1 \leq i \leq n$, there exists a clause of the form (13) with $\Gamma'_i\sigma_{t'} \subseteq \Gamma_{t'}$, $\Delta'_i\sigma_{t'} \subseteq \Delta_i \cup \Delta_{t'}$ for some $\Delta_i$ such that $R^*_{t'} \not\models \Delta_i$ (possibly $\bot$), and $C'_i\sigma_{t'} = C_i$. Such clauses satisfy the preconditions for acting as side clauses of the $\mathsf{Pred}$ rule for $i$ with $1 \leq i \leq n$. Furthermore, if $X_{t'} = v_r$ because $t' \in \Sigma_u$, we define $n'$ and $C_i$ for each $i$ with $n+1 \leq i \leq n'$ as in the $\mathsf{Pred}$ rule, that is, $\bigwedge_{i=n+1}^{n'} C_i = \mathsf{core}_v$. Since we are in case (iii), we have a clause $\top \to C_i\sigma_t \in \mathcal{S}_{v_r}$, as $\mathsf{core}_v \subseteq K_1$ for $K_1$ defined as in the $\mathsf{Succ}$ rule with respect to $X_{t'}$. For each $i$ with $n+1 \leq i \leq n'$, we then define define $\Gamma'_i = \Gamma_i = \top$, $\Delta'_i = \Delta_i = \bot$, and $C'_i = C_i\sigma_t$. Clauses $\Gamma'_i \to \Delta'_i \vee C'_i$ satisfy the preconditions for acting as side clauses of the $\mathsf{Pred}$ rule for $i$ with $n+1 \leq i \leq n'$. If $v \neq v_r$, recall that $n' = n$. Also, if $t' = c$, we have that by definition of $\Gamma_{\mathsf{Rt}}$, for each $A_i$ with $1 \leq i \leq m$, there exist generative clauses $\Gamma_{n'+i} \to \Delta_{n'+1} \vee C_{n'+1}$ in $N_c$ with $C_{n'+i} = A_i$, and by arguments analogous to those used above for $C_i \in \Gamma_{\mathsf{Rt}}$, there must exist a clause $\Gamma'_{n'+i} \to \Delta'_{n'+i} \vee C_{n'+i}$ in $\mathcal{S}_q$ with $\Gamma'_{n'+i}\sigma_c = \Gamma_{n'+i} \subseteq \Gamma_c$, $\Delta'_{n'+i}\sigma_c \subseteq \Delta_{n'+i} \cup \Delta_c$, and $\Delta_{n'+i} \not>_q C_{n'+i}$. To uniformise nomenclature, if $t' \neq c$, for $n'+1 \leq i \leq m$ we define $\Gamma'_i = \Gamma_i = A_i$, and $\Delta'_i = \Delta_i = \bot$; observe that by condition L4 we have that $A_i \in \Gamma_{\mathsf{Rt}}$ implies $A_i \in \Gamma_{t'}$. Finally, since we are in case (iii), there is an edge labelled $f$ connecting $X_{t'}$ to $X_t$.

We have shown that all preconditions of the Pred rule are satisfied, so since this rule is not applicable by hypothesis, we have:

$$\bigwedge_{i=1}^{n'} \Gamma'_i \wedge \bigwedge_{i=n'+1}^{n'+m} \to \bigvee_{i=1}^{n'} \Delta'_i \vee \bigvee_{i=n'+1}^{m} \Delta'_i \vee \bigvee_{i=1}^{k} L'_i\sigma \,\hat{\in}\, \mathcal{S}_{X_{t'}}, \tag{14}$$

where $\sigma = \{x \mapsto f(x),\ y \mapsto x\}$ if $X_{t'} \neq v_r$ and $\sigma = \{x \mapsto f(u),\ y \mapsto u\}$ otherwise, with $u = t'$. We have seen that for $1 \leq i \leq n' + m$, $\Gamma'_i\sigma_{t'} \subseteq \Gamma_t$, and the head of clause (14) is clearly ground, so by Lemma 7, we have:

$$\bigwedge_{i=1}^{n'} \Gamma_i \wedge \bigwedge_{i=n'+1}^{n'+k} \Gamma_i \to \bigvee_{i=1}^{n'} \Delta_i \vee \bigvee_{i=n'+1}^{n'+m} \Delta_i \vee \pi_{t'}\left(\bigvee_{i=1}^{k} L'_i\sigma\sigma_{t'}\right) \,\hat{\in}\, N_{t'} \tag{15}$$

For $1 \leq i \leq n$, since the clause $\Gamma_i \to \Delta_i \vee C_i$ is generative, we have by Lemma 12 that $R^*_{t'} \not\models \Delta_i$. If $n' \neq n$, for $i$ with $n+1 \leq i \leq n'$, we have that $\Gamma'_i\sigma_{t'} = \top \subseteq \Gamma_{t'}$, $\Delta'_i \vee C'_i$ is ground, so $\Gamma_i \to \Delta_i \vee C'_i \in N_{t'}$, and since $\Delta_i = \bot$, we have $R^*_{t'} \not\models \Delta'_i$. If $t' = c$, then for $i$ with $n'_1 \leq i \leq n' + k$, we have that since $\Gamma_i \to \Delta_i \vee C_i$ is generative, by Lemma 12, $R^*_{t'} \not\models \Delta_i$. If $t' \neq c$, then $\Gamma'_i \in \Gamma_{\mathsf{Rt}}$ and by condition L4 we have $\Delta_i = \bot$, so $R^*_{t'} \not\models \Delta_i$ trivially. Finally, since $L'_i\sigma\sigma_{t'} = L'_i\sigma_t \in \Delta_t$, we have that by definition of $\Delta_t$, and the fact that $R^*_{t'} \not\models \Delta_{\mathsf{Rt}}$, either by definition of $\Delta_{\mathsf{Rt}}$ (if $t' = c$) or by Lemma 18 (if $t' \neq c$), $R^*_{t'} \not\models L'_i\sigma\sigma_{t'}$. We have therefore proved that $R^*_{t'}$ does not satisfy the head of clause (15); but this contradicts Lemma 8. Thus, we obtain a contradiction, and hence we complete the proof for this case.

Finally, for condition L7, we have that for $t = c$, the condition is guaranteed by our assumption of condition C2 and the definition of $\Gamma_c$. For $t \neq c$, we consider first $A \in \Gamma_{\mathsf{Rt}}$. By definition of $\Gamma_{\mathsf{Rt}}$, there is a generating clause $\Gamma \to \Delta \vee A$ in $N_c$. By Lemma 6 there must exist a clause $\Gamma' \to \Delta' \vee A$ in $\mathcal{S}_q$ with $\Gamma'\sigma_c = \Gamma$, $\Delta'\sigma_c \subseteq \Delta \cup \Delta_c$. If $L$ is a literal of $\Delta'$ and $\Delta_Q$, by condition C2 we have that if $L \succ_q A$, then $A \in \Delta_Q$, which is impossible since $A$ is ground. If $L \in \Delta' \backslash \Delta_Q$ then $L\sigma_t \in \Delta$, so if $L \succ_q A$ we have $\Delta >_c A$, and we reach a contradiction. If $t = u$, since $r$-Succ is not applicable, we have $A \to A \,\hat{\in}\, \mathcal{S}_{v_r}$. Otherwise, we are in case (iii); if $t' = c$ we consider clause $\Gamma' \to \Delta' \vee A$ with $\Delta' \not\succ_q A$ in $X_{t'}$; otherwise by condition L7, we have $A \to A \,\hat{\in}\, \mathcal{S}_{X_{t'}}$, and since $A \in \Gamma_{\mathsf{Rt}}$, using condition L6 it is easy to see that, without loss of generality, we can assume $A \to A \in \mathcal{S}_{X_{t'}}$ (the alternative being $\top \to A$, but never $\top \to \bot$ or $A \to \bot$ so as to not violate condition L6). Furthermore, since we are in case (iii),

there must be a clause in $X_{t'}$ with a maximal literal mentioning $f$. Since $\mathsf{Succ}$ is not applicable, and $A \in \mathsf{Su}$, and either $A \to A \in \mathcal{S}_{X_{t'}}$ or $\Gamma' \to \Delta' \vee A \in \mathcal{S}_{X_{t'}}$, we conclude $A \to A \,\hat{\in}\, \mathcal{S}_{X_t}$. Finally, for $A \in \Gamma_t \backslash \Gamma_{\mathsf{Rt}}$, we must be in case (iii), and we can argue as in the proof in the previous bullet point that $A$ is irreducible w.r.t. $R_{t'}$, and there is a generative clause $\Gamma \to \Delta \vee A$ for $A$ in $R_{t'}$, and there is a clause $\Gamma' \to \Delta' \vee A'$ in $\mathcal{S}_{X_{t'}}$ where $\Gamma'\sigma_{t'} = \Gamma$, $\Delta'\sigma_{t'} \subseteq \Delta \cup \Delta_t$, $A'\sigma_t = A$ and $\Delta \not\succ_{X_{t'}} A$. Furthermore, since $A \in \Gamma_t$, by definition of $\Gamma_t$ we have $A \in \mathsf{Su}_t$, and hence $A' \in \mathsf{Su}$, so $A' \in K_2$, where $K_2$ is as specified in the $\mathsf{Succ}$ rule for $X_{t'}$. Since $\mathsf{Succ}$ is not applicable, and there is an edge $f$ connecting $X_{t'}$ and $X_t$, we have $A' \to A' \,\hat{\in}\, \mathcal{S}_v$, which implies $A \to A \,\hat{\in}\, N_t$ by Lemma 7.

#### 6.3.4.3 Properties of the Model

We now show that $R$ is Church-Rosser, and therefore we can find a single normal form w.r.t $R$ for each term in the model.

**Lemma 30.** *The rewrite system $R$ is terminating.*

*Proof.* By Lemma 1, if we define a simplification order on ground terms and show that this order embeds the rules of $R$, termination of $R$ is guaranteed. Consider an extension of the ordering $\succcurlyeq$ to all $\mathsf{p}$-function symbols in a way that makes $\mathsf{true}$ the smallest element of the order. Let $\triangleright$ be the LPO induced by this extension of $\succcurlyeq$. Since $\succcurlyeq$ is $\mathsf{a}$-admissible, it is well-founded; hence, as we mentioned in Section 2.2, this implies that $\triangleright$ is a simplification ordering. To complete the proof, we show that all rules in $R$ are embedded in $\triangleright$.

Consider an arbitrary rule $l \Rightarrow r$ of $R$. By definition of $R$, there is $t$ such that $l \Rightarrow r \in R_t$. However, if $l \Rightarrow r \in R_t$, by definition of $R_t$ condition R2, we have that $l >_t r$. If $l$ and $r$ are $\mathsf{a}$-terms, then the equality $l \approx r$ must be of one of these forms:

- $f(t) \approx g(t)$ with $f(t) >_t g(t)$. Such equality can only have been generated in $R_t$. Furthermore, $f(t) >_t g(t)$ implies $f \succcurlyeq g$, so we have $f(t) \triangleright g(t)$.
- $f(t) \approx t$ with $f(t) >_t t$, or $f(t) \approx t'$ with $f(t) >_t t'$. But we have $f(t) \triangleright t$ and $f(t) \triangleright t'$ trivially.
- $f(t) \approx u$ with $f(t) >_t u$. Since $f \succcurlyeq u$ for any $f \in \Sigma_f$ and $u \in \Sigma_u$, we have $f(t) \triangleright u$ trivially.
- $u \approx s$ with $u >_t s$. By definition of $>_t$, $s$ can only be of the form $u'$ for some $u' \in \Sigma_u$ with $u \succcurlyeq u'$, in which case we have $u \triangleright u'$.

Other forms are not possible; this follows from the fact that $>_t$ is a simplification order, the fact that $t$ and $t'$ are irreducible w.r.t. $R_t^*$ for any $t \neq c$ according to Lemma 15, and the fact that the definition of $>_t$ ensures that constants in $\Sigma_u$ are the smallest $\mathsf{a}$-terms. To conclude this proof, observe that if $l$ and $r$ are $\mathsf{p}$-terms, then $l \neq \mathsf{true}$ and $r = \mathsf{true}$. Since $\mathsf{true}$ is the smallest element in $\triangleright$ by definition, we trivially have $l \triangleright r$. □

**Lemma 31.** *The rewrite system $R$ is left-reduced.*

*Proof.* Assume that there is a rule $l \Rightarrow r$ in $R$ such that $l$ is reducible by $R' = R \backslash \{l \Rightarrow r\}$. Let $s$ be a term such that $l \Rightarrow r \in R_s$. Let $p$ be (one of) the deepest position(s) in $l$ at which $R'$ reduces $l$, so that any proper subterm of $l|_p$ is irreducible by $R'$. Let $l|_p \Rightarrow r'$ be a rule in $R'$ which reduces $l$ at $p$. Let $s'$ be a term such that $l|_p \Rightarrow r' \in R_{s'}$. We have that $s' \neq s$, for otherwise we contradict Lemma 10, which guarantees that $R_s$ is Church-Rosser, and therefore left-reduced.

If $l|_p$ is a $\mathsf{p}$-term, then $l|_p = l$ and due to the form of the rewrite systems $R_t$, we have $r' = \mathsf{true}$ and $r = \mathsf{true}$. But since $l_p \Rightarrow r' \in R'$, we have $l \Rightarrow r \in R'$, which contradicts our definition of $R'$. If $l|_p$ is an $\mathsf{a}$-term, $l|_p \Rightarrow r'$ can be only of the form $f(t) \Rightarrow g(t)$, $f(t) \Rightarrow t$, $f(t) \Rightarrow t'$, $f(t) \Rightarrow u$, or $u_1 \approx u_2$ with $u_1 \succcurlyeq u_2$, for the same reasons as in the proof of Lemma 30. In case $u_1 \Rightarrow u_2$, by definition of $R$ and $\Gamma_{\mathsf{Rt}}$, we have $u_1 \Rightarrow u_2 \in R_c$. Since $R_s$ is Church-Rosser, $l$ is irreducible w.r.t. $R_s$. In particular, $u_1$ is irreducible w.r.t. $R_s$. However, by Lemma 14 we have $R_s^* \models u_1 \approx u_2$, and since $u_1 \succcurlyeq u_2$, and hence $u_1 >_s u_2$, and the rules of $R_s$ are embedded in $>_s$, we have that $u_1$ cannot be the normal form of $u_1$ w.r.t. $R_s$, and this contradicts the fact that $u_1$ is irreducible w.r.t. $R_s$. Thus, consider the case where $l|_p$ is of the form $f(t)$. The only fragment that can contain $l_p \to r'$, then, is $R_t$, since it is the first fragment where elements of the form $f(t)$ can appear, according to the unfolding strategy, and since $f(t)$ is not irreducible w.r.t. $R_t$, no fragments are defined for $f(t)$ or its successors. Thus, $s' = t$. Now, observe that $R_s$ cannot have been defined before $R_{s'}$, for $R_{s'}$ is the first fragment where elements of the form $f(t)$ can appear. Since we also have shown that $s' \neq s$, we conclude that $R_{s'}$ has been defined before $R_s$. However, since $f(t)$ is not irreducible w.r.t. $R_t$, no further fragments containing $l|_p$ are defined after $R_t$, so $R_s$ is not defined. Hence, we obtain a contradiction. □

Lemmas 30 and 31 together imply the result we are looking for:

**Lemma 32.** *The rewrite system $R$ is Church-Rosser.*

Notice that the proof above also shows that for each $t \neq c$, the system $R_c^u \cup R_t \backslash R_t^u$ is also Church-Rosser.

Before we prove that the interpretation induced by rewrite system $R$ is a model of the ontology, we show two crucial properties of the model $R^*$. The first property shows that $R^*$ is "locally complete", that is, for any term $t \neq \Sigma_u$, we can find in $R_t^*$ the set $\mathsf{H}_t$ of all *hyperresolution triggers*, i.e. the irreducible atoms in $R^*$ which unify with the body of a clause in $\mathcal{O}$ and a substitution $\sigma$ such that $x\sigma = t$. For $t \in \Sigma_u$, a weaker condition holds: for any term $t \in \Sigma_u$, $R_t^*$ contains $\mathsf{H}_t \backslash \mathsf{V}_t$, where $\mathsf{V}_t$ contains all atoms in $\mathsf{H}_t$ of the form $S(t,s)$ or $S(s,t)$, with $s$ an $\mathsf{a}$-term of the form $s = f(s')$ for some $f \in \Sigma_f$. We will later see that each atom of the forms $S(t,s)$ or $S(s,t)$ in $\mathsf{V}_t$ is in $R_s^*$. Notice that due to the form of atoms in $R^*$, if $t \notin \Sigma_u$, $\mathsf{H}_t = \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$, and if $t \in \Sigma_u$, $\mathsf{H}_t \backslash \mathsf{V}_t = \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$.

**Lemma 33** (Local Completeness)**.** *Assume $c$ is irreducible w.r.t. $R_c$. For each ground term $t$, each $f \in \Sigma_f$, and each atom $A \in \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$ such that $R^* \models A$ and every $\mathsf{a}$-term in $A$ is irreducible by $R$, then $R_t^* \models A$.*

*Proof.* Let $t$ be a ground term, $f$ an arbitrary element of $\Sigma_f$, and $A$ an atom as described in the lemma. First, notice that if $A \in \mathsf{Rt}$, then by Lemma 18, $R_t^* \models A$. For the remainder of this proof, we assume that $A \notin \mathsf{Rt}$. Since the $\mathsf{a}$-terms of $A$ are irreducible, we have $A \Rightarrow \mathsf{true} \in R$.

- Suppose $A \in \mathsf{Su}_t$. Atom $A$ can be of the form $B(t), S(t,t'), S(t',t)$. By definition of $R$ and the fact that $A \notin \mathsf{Rt}$, $A \Rightarrow \mathsf{true}$ can occur only in $R_{t'}$ or $R_t$. However, if $A \Rightarrow \mathsf{true} \in R_{t'}$, then $A \in R_{t'}$ and hence $A \in \Gamma_t$, so by Lemma 21, we have $A \in R_t^*$.

- Suppose $A \in \mathsf{Pr}_{f(t)}$. Atom $A$ can be of the form $B(t), S(t,f(t)), S(f(t),t)$. By definition of $R$ and the fact that $A \notin \mathsf{Rt}$, $A \Rightarrow \mathsf{true}$ can only occur in $R_t$ or $R_{f(t)}$. Hence, we have that either $A \in R_t^*$ or $A \in R_{f(t)}^*$. But if $A \notin R_t^*$, then $A \in \Delta_{f(t)}$, so by Lemma 21, we have $A \notin R_{f(t)}^*$, which contradicts our previous claim.

- Suppose $A \in \mathsf{Ref}_t$. Since $A \notin \mathsf{Rt}$, $A \Rightarrow \mathsf{true}$ can only occur in $R_t$, so $A \in R_t^*$.

- Suppose $A \in \mathsf{Nom}_t$. Since $A \notin \mathsf{Rt}$, $A \Rightarrow \mathsf{true}$ can only occur in $R_t$, so $A \in R_t^*$.

□

The second property illustrates a "structural monotonicity" property of the interpretation: if an equality or inequality is satisfied in $R_t^*$, then it is also satisfied by $R^*$. To prove such property, first we introduce a preliminary lemma.

**Lemma 34.** *For any term $t$ with $t \neq c$, and any arbitrary term $s$, the normal form of $s$ w.r.t. $R_t$ is the same as the normal form of $s$ w.r.t. $R_c^u \cup R_t \backslash R_t^u$.*

*Proof.* Let $u_1'$ be the normal form of $u$ w.r.t. $R_c^u \cup R_t \backslash R_t^u$. Since the elements of $\Sigma_u$ are the smallest $\mathsf{a}$-terms in the orders $>_t$ and $>_c$, we have that $u_1'$ is the normal form of $u$ w.r.t. $R_c^u$. Hence, $R_c^* \models u \approx u'$, and by Lemma 18, $R_t^* \models u \approx u'$. Furthermore, it cannot be the case that $R_t^* \models u' \approx u''$ for some $u''$ with $u' \succcurlyeq u''$, since this violates Lemma 14. Thus, $u'$ is irreducible w.r.t. $R_t$ and hence $u'$ is the normal form of $u$ w.r.t. $R_t$. With this, we have $(R_c^u)^* = (R_t^u)^*$ due to Lemma 34 and the fact that both $R_c^u$ and $R_t^u$ mention only elements in $\Sigma_u$. Let $s'$ be the normal form of $s$ w.r.t. $R_t^*$. There exists a sequence of $n$ terms $s_1, s_2, \cdots, s_n$ such that $s_1 = s$, $s_n = s'$, and $s_i \Rightarrow s_{i+1} \in R_t$. Given any $i$ with $1 \leq i < n$, we have that if $s_i \Rightarrow s_{i+1} \in R_t \backslash R_t^u$, then $(R_c^u \cup R_t \backslash R_t^u)^* \models s_i \approx s_{i+1}$ trivially. Otherwise, we have $s_i \Rightarrow s_{i+1} \in R_t^u$, and thus $(R_t^u)^* \models s_i \approx s_{i+1}$. We have shown that this implies $(R_c^u)^* \models s_i \approx s_{i+1}$, and since $(R_c^u)^* \subseteq (R_c^u \cup R_t \backslash R_t^u)^*$, we have $(R_c^u \cup R_t \backslash R_t^u)^* \models s_i \approx s_{i+1}$. Since $(R_c^u \cup R_t \backslash R_t^u)^*$ is a congruence, we conclude that it satisfies $s \approx s'$.Suppose $s'$ is nor irreducible w.r.t $R_c^u \cup R_t \backslash R_t^u$, and let $s_n \Rightarrow s_{n+1}$ be a rule of $R_c^u \cup R_t \backslash R_t^u$ which reduces it. Since $s'$ is irreducible w.r.t. $R_t$, we have that $s_n \Rightarrow s_{n+1} \in R_c^u$. But then, $R_c^* \models s_n \approx s_{n+1}$, and by Lemma 18 we have $R_t^* \models s_n \approx s_{n+1}$. Since $s_n \succcurlyeq s_{n+1}$, for both $s_n$ and $s_{n+1}$ are in $\Sigma_u$, and $s_n \succcurlyeq s_{n+1}$, then $s_n >_t s_{n+1}$, which contradicts the fact that $s_n$ is irreducible w.r.t. $R_t^*$. This completes the proof. □

**Corollary 35.** *For any $t$, $R_t^* = (R_c^u \cup R_t \backslash R_t^u)^*$*

*Proof.* The corollary follows immediately from Lemma 34. □

**Lemma 36** (Structural monotonicity)**.** *Let $s_1$ and $s_2$ be both DL-$\mathsf{a}$-terms or DL-$\mathsf{p}$-terms such that $s_1 \bowtie s_2$ is a DL-literal, with $\bowtie \in \{\approx, \not\approx\}$, and let $\tau$ be a substitution where the terms in the range of $\tau$ are irreducible by $R$, and such that $x\tau$, $s_1\tau$, and $s_2\tau$ are ground. Suppose that for each $z_i$ such that $z_i\tau$ is ground, $z_i\tau$ is in the $\mathsf{a}$-neighbourhood of $x\tau$. Let $R_{x\tau}^*$ be such that if $c$ is not irreducible w.r.t. $R_c$, then $x\tau \neq c$. Then, if $R_{x\tau}^* \models s_1\tau \bowtie s_2\tau$, we have $R^* \models s_1\tau \bowtie s_2\tau$.*

*Proof.* Consider first the case where $\bowtie$ is an equality. If $t = c$, this is trivial since $R_{x\tau} \subseteq R$ ($c$ is irreducible w.r.t. $R_c$), and hence $R_{x\tau}^* \subseteq R^*$. Otherwise, the result

follows from Corollary 35 and the fact that $R_c^u \cup R_{x\tau} \backslash R_t^u \subseteq R$. Next, we consider the base where $\bowtie$ is an inequality. Let $t = x\tau$. Since $t$ is irreducible by $R$, according to the unfolding strategy at the beginning of Section 6.3.4.1, system $R_t$ has been defined. Let $s_1'$ and $s_2'$ be the normal forms of $s_1\tau$ and $s_2\tau$ w.r.t. $R_t$, respectively. If $t \neq c$, by Lemma 34, they are also the normal forms of $s_1\tau$ and $s_2\tau$ w.r.t. $R_c^u \cup R_{x\tau} \backslash R_t^u$. If $t \neq c$, $R_c^u \cup R_t \backslash R_t^u \subseteq R^*$, and otherwise by hypothesis we have that $c$ is irreducible w.r.t. $R_c$ so $R_c \subseteq R^*$. Thus, to show $R^* \not\models s_1\tau \approx s_2\tau$, it suffices to show that both $s_1'$ and $s_2'$ are irreducible w.r.t. $R$, since in that case they will still be the normal forms of $s_1\tau$ and $s_2\tau$ (respectively) w.r.t. $R$.

Due to the form of DL-literals, we have that $s_1$ and $s_2$ must be of the forms $f(x)$, $x$, or $z_i$. Furthermore, since $s_1\tau$ and $s_2\tau$ are in the $\mathsf{a}$-neighbourhood of $t$, they are of the form $t, t'$, $f(t)$ for some $f \in \Sigma_f$, or $u$. Furthermore, since $R_t$ does not contain $\mathsf{a}$-terms outside of the $\mathsf{a}$-neighbourhood of $t$, terms $s_1'$ and $s_2'$ are also of one of the forms appearing in that list. We finish the proof by showing if any term of such form is irreducible in $R_t$, then it is also irreducible in $R$.

We already have that $t$ is irreducible by $R$ and since $t'$ is a subterm of $t$, we have that $t'$ is also irreducible by $R$. For the case $u \in \Sigma_u$, if $t = c$, the fact that $u$ is irreducible w.r.t. $R_t$ implies that $u$ is irreducible w.r.t. $R_c$, and since elements of $\Sigma_u$ are smallest in $>_c$, this implies that $u$ is irreducible w.r.t. $R_c^u$. If $t \neq c$, the fact that $u$ is irreducible w.r.t. $R_t$ implies it is irreducible w.r.t. $R_c^u \cup R_t \backslash R_t^u$ due to Lemma 34, and since elements of $\Sigma_u$ are smallest in $>_t$ and $>_c$, this implies that $u$ is irreducible w.r.t. $R_c^u$. Thus, by construction of $R$, the fact that $u$ is irreducible w.r.t. $R_c^u$ implies $u$ is irreducible w.r.t. $R$. Finally, for the case $f(t)$, if $f(t)$ is irreducible by $R_t$, then by the unfolding strategy at the beginning of Section 6.3.4.1, the only other fragments where $f(t)$ can appear are $R_{f(t)}$ or $R_{g(f(t))}$ for some $g$. But in such fragments, Lemma 15 implies that $f(t)$ is irreducible by $R_{f(t)}$ and $R_{g(f(t))}$. Hence, $f(t)$ must be irreducible by $R$. □

We are finally ready to show that $R^*$ is a model of the ontology.

**Lemma 37.** *For each DL-clause $\Gamma \to \Delta \in \mathcal{O}$, we have $R^* \models \Gamma \to \Delta$.*

*Proof.* Let $\tau'$ be an arbitrary substitution such that $\Gamma\tau' \to \Delta\tau'$ is ground. Let $\tau$ be the substitution obtained by replacing each ground term in the range of $\tau'$ by its unique normal form w.r.t. $R$; such unique normal form exists since we know that $R$ is Church-Rosser by Lemma 32. Since $R^*$ is a congruence, $R^* \models \Gamma\tau' \to \Delta\tau'$ if and only if $R^* \models \Gamma\tau \to \Delta\tau$. Thus, we only need to assume $R^* \models \Gamma\tau$ and show $R^* \models \Delta\tau$. To prove $R^* \models \Delta\tau$, we proceed in three steps: first, we find a model fragment that

contains all atoms in $\Gamma\tau$; second, we show that this fragment satisfies some literal in $\Delta\tau$; finally, we show that $R^*$ also satisfies this literal. Notice that if $c$ appears, then it is irreducible w.r.t. $R_c$ and we can use all the results proved above for this case.

Consider an arbitrary atom $A \in \Gamma$. By definition of DL-clauses, $A$ is of the form $B(x)$, $S(x,x)$, $S(x,z_i)$, or $S(z_i,x)$. Since terms in the range of $\tau$ are irreducible by $R$, $A\tau$ cannot be reduced by $R$ in proper positions. This, together with the fact that $A \in \Gamma$, implies $A\tau \Rightarrow \mathsf{true} \in R$. If we denote $x\tau$ by $t$, by construction of $R$, we have that $A\tau$ must be of the form $B(t)$, $S(t,t)$, $S(t,f(t))$, $S(f(t),t)$, $S(t,t')$, $S(t',t)$, $S(t,u)$, $S(u,t)$, $S(s,t)$, or $S(t,s)$, where $s$ is an arbitrary term not in $\Sigma_u$; the last two cases occur only if $t \in \Sigma_u$.

If $t \in \Sigma_u$ and $A\tau$ is of the form $S(s,t)$ or $S(t,s)$, with $s \notin \Sigma_u$, and also $s \neq f(t)$, we have that $S(s,t) \Rightarrow \mathsf{true}$ appears in the fragment $R_s$ due to Lemma 33, since $A\tau \in \mathsf{Nom}_s$. Due to the form of $\mathcal{ALCHOIQ}^+$ DL-clauses, it is easy to see that if $A\tau$ is of the form $S(t,s)$, then $S$ is a numerically restricted predicate, but this contradicts Lemma 22. In case $A\tau$ is of the form $S(s,t)$, then due to the form of $\mathcal{ALCHOIQ}^+$ DL-clauses, for every other atom $A' \in \Gamma$, we have that $A'\tau$ is of the form $B(t)$ (with $t$ irreducible, by hypothesis), and so by Lemma 33, $B(t) \in R_s$ since $B(t) \in \mathsf{Su}_s$. Hence, all atoms in $\Gamma\tau$ are contained in fragment $R^*_s$.

If $t$ is either not in $\Sigma_u$, or we have $t \in \Sigma_u$ but $A\tau$ is not of the form $S(s,t)$ or $S(t,s)$ with $s \notin \Sigma_u$ or of the form $f(t)$, we have the following possibilities:

- $A = B(x)$, so $A\tau = B(t)$. But if $A = B(x)$, then $B(x) \in \mathsf{Su}$. Hence, $B(t) \in \mathsf{Su}_t$.
- $A = S(x,x)$, so $A\tau = S(t,t)$. But if $A = S(x,x)$, then $A\tau = S(t,t) \in \mathsf{Ref}_t$.
- $A = S(z_i,x)$, so $A\tau = S(t',t)$, or $S(f(t),t)$, or $S(u,t)$. Notice that since $A$ appears in the body of an ontology clause, we know $S(y,x) \in \mathsf{Su}$. Therefore, we have $S(t',t) \in \mathsf{Su}_t$ and $S(f(t),t) \in \mathsf{Pr}_{f(t)}$. Finally, we have $S(u,t) \in \mathsf{Nom}_t$.
- $A = S(x,z_i)$, which is symmetric to the case above.

Consider the case where $A_i\tau \in \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$ for every $A_i$. We have that by Lemma 33, $A_i\tau \in R_t$, so $N_t$ contains a generative clause of the form

$$\Gamma_i \rightarrow \Delta_i \vee A_i \text{ with } A_i >_t \Delta_i \text{ and } \Gamma_i \subseteq \Gamma_t$$

Let $v = X_t$. If $A_i \in \mathsf{Rt}$, by Lemma 14 we have $R^*_c \models A_i$, so due to conditions L6 and L7 we can assume we have $A_i \rightarrow A_i \in \mathcal{S}_v$ without loss of generality. Otherwise, we are in the conditions of Lemma 5. Furthermore, if $t' \in \Sigma_u$ and $t'$ appears in $A_i$,

we have that $A_i \in \mathsf{Su}_t$, and by definition, $A_i \in \Gamma_t$, so by conditions L6 and L7 we can assume, without loss of generality, that $A_i \to A_i \in \mathcal{S}_v$. Otherwise, we are in the conditions of Lemma 6. Therefore, for every $1 \leq i \leq n$, there exists a clause in $\mathcal{S}_v$ of the form $\Gamma'_i \to \Delta'_i \vee A'_i$ with $\Gamma'_i\sigma_t = \Gamma_i$, $\Delta'_i\sigma_t \subseteq \Delta'_i \cup \Delta_t$, and $A'_i\sigma_t = A_i$, with $\Delta'_i \not\succ_v A'_i$. Since the $\mathsf{Hyper}$ rule is not applicable to the ontology DL-clause and these clauses (this includes the case where $n = 0$), we have that

$$\bigwedge_{i=1}^{n} \Gamma'_i \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta'_i \,\hat{\in}\, \mathcal{S}_v,$$

with $\sigma$ the substitution where we replace $t$ in the domain by $x$, and if $n = 0$ then $\bigwedge_{i=1}^{n} \Gamma'_i$ and $\bigvee_{i=1}^{n} \Delta'_i$ are replaced by the empty conjunction and disjunction, respectively. Since $\Gamma_i \subseteq \Gamma_t$ and $\Delta_i \vee A_i$ is ground, by Lemma 7, we have

$$\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\tau \vee \bigvee_{i=1}^{n} \Delta_i \,\hat{\in}\, N_t$$

and by Corollary 17, we have $R^*_t \models \Delta\tau \vee \bigvee_{i=1}^{n} \Delta_i$. However, by Lemma 12 or the fact that $\Delta_i = \bot$, we have that $R^*_t \not\models \Delta_i$, which implies $R^*_t \models \Delta\tau$. Then, by Lemma 36, we conclude that $R \models \Delta\tau$.

To complete the proof, we consider the case where we have $A_i\tau \in \mathsf{Su}_s \cup \mathsf{Pr}_{f(s)} \cup \mathsf{Ref}_s \cup \mathsf{Nom}_s$ for every $A_i$. In this case, due to the form of $\mathcal{ALCHOIQ}^+$ DL-clauses, the ontology clause is of the form DL3, DL4a, DL7, DL8, or DL9. An analogous argument to the one above shows $R^*_s \models \Delta\tau$, and since for these axioms $\Delta$ consists exclusively of equalities, by Corollary 35 we trivially have $R \models \Delta\tau$. □

The only remaining task is to show that the model is a counterexample of the input query.

**Lemma 38.** *$R^* \not\models \Gamma_Q \to \Delta_Q$.*

*Proof.* Observe that Lemma 21 implies $R^*_c \models \Gamma_c$ and $R^*_c \not\models \Delta_c$. If $c$ is irreducible w.r.t. $R_c$, we have $R_c \subseteq R$, which guarantees $R^* \models \Gamma_c$. However, observe also that for each $B(x) \in \Delta_Q$, by definition of $\mathsf{Pr}$ we have $B(y) \in \mathsf{Pr}$, so $B(c) \in \mathsf{Pr}(f(c))$, and hence by Lemma 33, we have that $R^*_c \not\models \Delta_c$ implies $R^* \not\models \Delta_c$. Thus, element $c$ disproves $\Gamma_Q \to \Delta_Q$.

If $c$ is not irreducible w.r.t. $R_c$, then let $u$ be the normal form of $c$ w.r.t. $R_c$. Let $B$ be an arbitrary predicate such that $B(c) \in \Gamma_c$ and let $B'$ be an arbitrary predicate such that $B'(c) \in \Delta_c$. Since we have $R^*_c \models \Gamma_c$ and $R^*_c \not\models \Delta_c$, as well as $R^*_c \models c \approx u$, we have $R^*_c \models B(u)$ and $R^*_c \not\models B'(u)$. Thus, we have $B(u) \in \Gamma_{\mathsf{Rt}}$, and $B'(u) \in \Delta_{\mathsf{Rt}}$. By

Lemma 18, $R^*_u \models B(u)$ and $R^*_u \not\models B'(u)$, so by Corollary 35 we conclude $R^* \models B(u)$. Finally, as argued above, $B'(u) \in \mathsf{Pr}(f(u))$, and hence by Lemma 33, the fact that $R^*_u \not\models B'(u)$ implies $R^* \not\models B'(u)$. Thus, we have $R^* \models \Gamma_Q\sigma_u$ and $R^* \not\models \Delta_Q\sigma_u$; therefore $\Gamma_Q \to \Delta_Q$ is disproved. □

## 6.4 Proofs for the Variant of the Calculus

**Theorem 3** (Soundness)**.** *Suppose $\mathcal{O}$ is a Horn ontology. Given a context structure $\mathcal{D}$ which is sound for $K$, the application to $\mathcal{D}$ of a rule from Table 5.3, a rule from Table 5.1 other than* Hyper*, or a rule from Table 5.2 other than $r$-*Succ*, with the eager context strategy, yields a context structure which is sound for $K$.*

The proof of Theorem 3 relies on the following auxiliary lemma, which shows that if $\mathcal{D}$ is a context structure which is sound for $K$, then every model of $\mathcal{O}$ where the interpretation of $K$ is not empty verifies that the interpretation of $\mathsf{core}_v$ for each $v \in \mathcal{V}$ is also not empty.

**Lemma 39.** *Let $K$ be a conjunction of atoms of the form $B_1(x) \wedge \cdots \wedge B_n(x)$, with $B_i \in \Sigma_A$ for $1 \leq i \leq n$; let $\mathcal{D}$ be a context structure $\langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \succcurlyeq, \theta \rangle$ which is sound for $K$, and let $v \in \mathcal{V}$ be an arbitrary context. For every model of $\mathcal{I}$ of $\mathcal{O}$ such that $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \to \bot$, we have $\mathcal{I} \not\models \mathsf{core}_v \to \bot$.*

*Proof.* Consider an arbitrary $K$ and $\mathcal{D}$. By condition Z1, there exists a context $w$ with $\mathsf{core}_w = K$ and such that for every $v \in \mathcal{V}$ with $v \neq v_r$, context $v$ is reachable from $w$ or $v_r$. We can therefore use structural induction on the context structure to prove the lemma.

For the base case $v = w$, we have that for every model $\mathcal{I}$ of $\mathcal{O}$ such that $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \to \bot$, then $\mathcal{I} \not\models \mathsf{core}_w \to \bot$ trivially since $\mathsf{core}_w = K$. For the base case $v = v_r$, $\mathsf{core}_{v_r} = \top$, and we have that for every model for every model $\mathcal{I}$ of $\mathcal{O}$ such that $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \to \bot$, $\mathcal{I} \not\models \top \to \bot$ trivially.

Now, suppose that $v \in \mathcal{V}$ is an arbitrary context such that for every model of $\mathcal{I}$ of $\mathcal{O}$ with $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \to \bot$, we have $\mathcal{I} \not\models \mathsf{core}_v \to \bot$. Let $v'$ be a successor of $v$ such that we have $\langle v, v', f \rangle \in \mathcal{E}$ for some $f \in \Sigma_u$. Since $\mathcal{D}$ is sound for $K$, if $v \neq v_r$ we have that $\mathcal{I} \models \mathsf{core}_v \to \mathsf{core}_{v'}\{x \mapsto f(x), y \mapsto x\}$. Therefore, if $\mathcal{I} \models \mathsf{core}_{v'} \to \bot$, then $\mathcal{I} \models \mathsf{core}_v \to \bot$, which contradicts the induction hypothesis; hence, $\mathcal{I} \not\models \mathsf{core}_{v'} \to \bot$. If $v = v_r$, condition Z3 ensures $\mathcal{I} \not\models \mathsf{core}_{v'} \to \bot$. □

With this lemma, we can now prove Theorem 3.

*Proof.* The proof is similar to the proof of Theorem 1 in section 6.2. For all inference rules other than Succ and $r$-Succ from Table 5.3, condition Z3 is proved exactly in the same way as conditions S1 and S2 from Theorem 1: we consider an arbitrary model $\mathcal{I}$ of $\mathcal{O} \cup \mathcal{C}_\mathcal{D}$ such that $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \rightarrow \bot$, and we use soundness of the hyperresolution, or the fact that $\approx$ is a congruence, in order to obtain that $\mathcal{I}$ satisfies every clause added to the context structure by the rule, or the clause $\mathsf{core}_v \rightarrow \mathsf{core}_w\{x \mapsto f(x), y \mapsto x\}$ for every edge with label $f \in \Sigma_f$ added to the context structure. In the case of Succ, the same argument applies to prove the first two properties of condition Z3. To prove the third property, suppose Succ is applied to $v_r$ and it creates an edge $\langle v_r, v, f\rangle$. Since $\mathcal{D}$ is sound for $K$, by condition Z2 we have that for each literal $L \in K_1$, clause $\top \rightarrow L$ is contained in $\mathcal{S}_{v_r}$. Furthermore, by condition Z3 we have that $\mathcal{I} \models \top \rightarrow L$ for each $L \in K_1$. Since we use the eager expansion strategy, $K_1 = \mathsf{core}_v$, so we conclude $\mathcal{I} \not\models \mathsf{core}_v \rightarrow \bot$.

To prove condition Z3 when the applied rule is $r$-Succ from Table 5.3, let $\top \rightarrow A$, with $A$ a ground atom, be a context clause added to $\mathcal{S}_{v_r}$ by this rule; since $\mathsf{core}_{v_r}$ is empty, we have to prove that $\mathcal{I} \models \top \rightarrow A$. Let $v$ the context where this clause is triggered. By condition Z2, the $r$-Succ rule is applied on a context clause of the form $\top \rightarrow A \in \mathcal{S}_v$. By Lemma 39, we have that $\mathcal{I} \not\models \mathsf{core}_v \rightarrow \bot$, and by condition Z3, we have $\mathcal{I} \models \mathsf{core}_v \rightarrow A$, which together ensure $\mathcal{I} \models \top \rightarrow A$.

To prove condition Z1, observe that if no new context is added to $\mathcal{D}$, then condition Z1 holds trivially in the new context structure because it already holds for $\mathcal{D}$. Thus, suppose that the rule applied to $\mathcal{D}$ is Succ, and that it introduces a new context $v'$. Let $v$ be the context where Succ is triggered. Since condition Z1 holds for $\mathcal{D}$, we have that $v$ is reachable from $w$ or $v_r$, where $w$ is the context defined in condition Z1 such that $\mathsf{core}_w = K$. Since the application of Succ introduces an edge $\langle v, v', f\rangle$ for some $f \in \Sigma_f$, we have that $v'$ is reachable from $w$ or $v_r$, and hence condition Z1 holds in the new context structure.

Finally, to show that condition Z2 holds in the new context structure, we consider each inference rule and show that any conclusion of these inference rules is a context clause, except in the case where Hyper applies to a non-root context with $\sigma(x) = u \in \Sigma_u$ and a literal of the form $S(x, u)$ or $S(u, x)$. However, looking at the form of ontology clauses in Table 2.4, it is easy to see that for any $z_i$, we have $\sigma(z_i) = x$ or $\sigma(z_i) \in \Sigma_u$, and therefore every new literal introduced in the inference can only be of the form $B_2(x), S_{B_2}(x, u)$, $S_{B_2}(u, x)$, $u \approx o$ or $x \approx o$ for some $o \in \Sigma_u$, all of which are context literals. Therefore, we only have to show that for any new clause $\Gamma \rightarrow \Delta$ added to $\mathcal{D}$, $\Gamma = \top$ and $\Delta$ contains at most one literal. Clauses added by the Core

and $r$-Succ rules already have the required form. For rules Hyper, Eq, Ineq, and Nom, it is easy to see that if the bodies of the premises are empty, and the heads of the premises have one literal, which is guaranteed by the fact that $\mathcal{O}$ is Horn and that condition Z2 holds for $\mathcal{D}$, then the body of the conclusion is empty, and the head has a single literal. The Factor rule is never triggered. For the Succ rule, condition Z2 for $\mathcal{D}$ ensures that $K_2 = K_1$, and since we use the eager strategy, we have that $\mathsf{core}_w = K_1$, where $w$ is the context selected by the expansion strategy. Therefore, no clauses are added to $w$. For the Pred rule, by condition Z2 the main premise in context $v$ must be of the form $\top \to L_1$ for a literal $L_1 \in \mathsf{Pr}$. Furthermore, in this application of the rule there are either no side premises, if the predecessor context $w$ is distinct from $v_r$, or a side premise $\top \to C_i\sigma$, where $\sigma$ is the corresponding substitution, for each atom $C_i$ in the core of $v$. Therefore, the conclusion is of the form $\top \to L_1\sigma$, with $\sigma$ being the corresponding substitution. An analogous argument applies to $r$-Pred. □

**Theorem 4** (Completeness)**.** *Let $\mathcal{D}$ be a sound context structure for $K$ derivable from a context structure with no auxiliary constants, and let $\omega$ be the number of contexts in $\mathcal{D}$. Suppose that the parameter $\Lambda$ used in the* Nom *rule is set to be greater or equal than $|\mathsf{Su}_\tau|^2 \cdot \omega$, and that no rule from Table 5.3, or Table 5.1 except* Hyper*, or Table 5.2 except $r$-*Succ*, can be applied to $\mathcal{D}$. Then, for each query clause $K \to \Delta_Q$ such that $\mathcal{O} \models K \to \Delta_Q$, and each context $q \in \mathcal{V}$ satisfying the properties in condition Z1 of Definition 12, and such that all of the following conditions hold:*

*C1. for each $A \in K$, we have $\top \to A \,\hat{\in}\, \mathcal{S}_q$;*

*C2. for each context atom $A \approx \mathsf{true} \in \Delta_Q$ and each context term $s$ not mentioning $y$ and distinct from $x$ and* true *such that $A \succ_q s$, we have $s \approx \mathsf{true} \in \Delta_Q$;*

*we have that $K \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ also holds.*

*Proof.* Notice that if no rule from Table 5.3, or Table 5.1 except Hyper, or Table 5.2 except $r$-Succ, can be applied to $\mathcal{D}$, then no rule from Table 5.1 or Table 5.2 is applicable to $\mathcal{D}$, except $r$-Succ on a context clause in any context $v$ where all preconditions of the rule except $A \to A \,\hat{\in}\, \mathcal{S}_v$ are satisfied, and the clause contains a maximal atom of the form $S(x, u)$ or $S(u, x)$ for $u \in \Sigma_u$ and $S \in \Sigma_S$. Indeed, if all preconditions of the $r$-Succ rule from Table 5.1 except $A \to A \,\hat{\in}\, \mathcal{S}_v$ are satisfied, and the premise does not contain a maximal atom of the form $S(x, u)$ or $S(u, x)$ for $u \in \Sigma_u$ and $S \in \Sigma_S$, and we have that $r$-Succ from Table 5.3 is not applicable, then it must be the case that $\top \to A \,\hat{\in}\, \mathcal{S}_{v_r}$ for premise $\Gamma \to \Delta \vee A\sigma$, but this implies $A \to A \,\hat{\in}\, \mathcal{S}_{v_r}$ and hence rule $r$-Succ from 5.2 is also not applicable.

We can therefore prove Theorem 4 using an argument analogous the proof of Theorem 2 in section 6.3, in the special case $\Gamma_Q = K$. The proof of Theorem 2 only uses the fact that $\mathcal{D}$ is saturated via $r$-$\mathsf{Succ}$ from Table 5.1 with respect to a context clause in a context $v$ where all preconditions of the rule except $A \to A \,\hat{\in}\, \mathcal{S}_v$ are satisfied, and the premise contains a maximal atom of the form $S(x, u)$ or $S(u, x)$ for $u \in \Sigma_u$ and $S \in \Sigma_S$, in the final paragraph of lemma 37, whenever $A'_i$ is of the form $S(x, t)$ or $S(t, x)$ for some $1 \leq i \leq n$. But then, instead of using the argument that $\mathcal{D}$ is closed by $r$-$\mathsf{Succ}$ in Table 5.2, we can use the fact that $\mathsf{Hyper}$ from Table 5.3 is now applicable on clauses $\Gamma'_i \to \Delta'_i \vee A'_i$ for $1 \leq i \leq n$ to treat this case exactly as the case $A_i\tau \in \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$ in the previous paragraph of the proof, and therefore prove $R^*_s \models \Delta\tau$ also in this case.

To account for the reduction in the depth limit $\Lambda$, we observe that due to the form of context clauses in $\mathcal{D}$, which is sound for $K$, the type of "blocking" clauses $C_{b_i}$ introduced in the proof of Lemma 22 is always $\langle \top, \bot \rangle$, so we can use Lemma 28 to show that there can only be at most one such clause per context. Furthermore, if the length $n = |\rho|$ of the "chain" defined for $o_\rho$ in the proof of Lemma 22 is greater than $|\mathsf{Su}_\tau|^2$, and since the expansion strategy introduces at most one context $v_{K_1}$ for each subset $K_1$ of $\mathsf{Su}_\tau$, we have that there are at least two contexts $v_{b_i}, v_{b_j}$ with $1 \leq b_i < b_j \leq n$ such that $\mathsf{core}_{v_{b_i}} \subseteq \mathsf{core}_{v_{b_j}}$; thus, an argument analogous to the proof of Lemma 28 shows that $o_j$ is equal to $o_i$, and hence by Lemma 27, the topmost constant $o_\rho$ in the chain is no longer irreducible, which leads to a contradiction.

□

# Chapter 7

# Solving DL Reasoning Problems Using the Calculus

Let $\mathcal{O}$ be an arbitrary $\mathcal{ALCHOIQ}^+$ ontology and let $\mathbf{B} \subseteq \Sigma_A$. This section shows how the calculus introduced in Chapter 5 can be used to decide the DL reasoning tasks presented in Section 2.4.3, which we now express in DL clausal notation:

- *Consistency checking:* decide whether there exists an interpretation $\mathcal{I}$ that satisfies $\mathcal{O}$.
- *Subsumption:* given concepts $C$ and $D$, decide whether each instance of $C$ is also an instance of $D$ in each model of $\mathcal{O}$.
- *Classification:* decide whether $\mathcal{O} \models B_1(x) \rightarrow B_2(x)$ for each $B_1, B_2 \in \mathbf{B}$.
- *Instance retrieval:* compute all constants $o_\epsilon \in \Sigma_u$ that are instances of a given concept $C$ in each model of $\mathcal{O}$.
- *Realisation:* for each constant $o_\epsilon \in \Sigma_u$, compute the subset $\mathbf{B}_o^+ \subseteq \mathbf{B}$ consisting of all $B \in \mathbf{B}$ such that $o_\epsilon$ is an instance of $B$ in each model of $\mathcal{O}$. This problem is sometimes defined with an additional requirement: if $B_1 \in \mathbf{B}_o^+$, there must be no $B_2 \in \mathbf{B}_o^+$ such that $\mathcal{O} \models B_2(x) \rightarrow B_1(x)$.

In Section 7.1 we propose a general algorithm which, given a set of subsumptions, returns those entailed by the ontology; we then establish soundness, completeness, and termination of the algorithm, discuss its worst-case complexity, and show that the parameters of the algorithm can be adjusted to ensure pay-as-you-go behaviour for $\mathcal{ALCHIQ}^+$, $\mathcal{ALCHOQ}$, $\mathcal{ALCHOI}$, and $\mathcal{ELH}$ ontologies. Finally, we show how the algorithm can be applied to solve the reasoning tasks listed above. Then, in Section 7.2, we describe a variant of the algorithm based on the modified calculus

described in Section 5.4, and we show that this variant of the algorithm is worst-case optimal for classification in Horn fragments of $\mathcal{ALCHOIQ}^+$ with nominals. In Section 7.3, we discuss the relationship between our calculi and algorithms and other CB procedures for DLs with nominals, as well as with resolution and tableau calculi for $\mathcal{SHOIQ}$. Finally, in Section 7.4, we prove the termination, complexity, and pay-as-you-go results for our algorithms.

## 7.1 General Algorithm for Subsumption Checking

Define a *query* $\mathbf{Q} = \{Q_1, \ldots, Q_n\}$ as a finite set of query clauses. Given $\mathbf{Q}$, Algorithm 1 returns the subset $\mathbf{Q}_+ \subseteq \mathbf{Q}$ such that $Q_i \in \mathbf{Q}_+$ if and only if $\mathcal{O} \models Q_i$.

**Algorithm 1** Given $\mathbf{Q}$, this algorithm decides whether $\mathcal{O} \models \Gamma_{Q_i} \to \Delta_{Q_i}$ for each $Q_i \in \mathbf{Q}$

A1. Create a context structure $\mathcal{D}$ with $\mathcal{V} = \{v_r\}$, $\mathcal{E} = \emptyset$, and $\mathcal{S}_{v_r} = \emptyset$. This step includes the selection of an $\mathsf{a}$-admissible order $\succcurlyeq$ and a choice of context order $\succ_{v_r}$ compatible with $\succcurlyeq$.

A2. For each $Q_i \in \mathbf{Q}$, either:

- Select a context $v_i \in \mathcal{D}$ with $\mathsf{core}_{v_i} \subseteq \Gamma_{Q_i}$ with a context order $\succ_{v_i}$ which satisfies Condition C2 of Theorem 2 w.r.t. $\Delta_{Q_i}$,

OR

- Add a context $v_i$ to $\mathcal{D}$ with $\mathsf{core}_{v_i} \subseteq \Gamma_{Q_i}$, and define the order $\theta(v_i) = \succ_{v_i}$ satisfying Condition C2 of Theorem 2 w.r.t. $\Delta_{Q_i}$.

A3. Select an expansion strategy $\mathsf{strat}$ and a depth limit $\Lambda$ for nominal labels.

A4. Apply the inference rules from Table 5.1 and Table 5.2 to $\mathcal{D}$ and $\mathcal{O}$ until no new inferences are possible.

A5. Return $\mathbf{Q}_+ = \{Q_i \in \mathbf{Q} \mid \Gamma_i \to \Delta_i \,\hat{\in}\, \mathcal{S}_{v_i}\}$.

Algorithm 1 generalises the algorithm in [20] for checking query clause entailment in $\mathcal{ALCHIQ}^+$. It starts by initialising the context structure with the root context $v_r$ containing no clauses. Step A3 selects an expansion strategy $\mathsf{strat}$ and a depth limit $\Lambda$ to be used in inferences by the $\mathsf{Succ}$ rule during Step A4. Steps A1 and A2 require the creation of new context orders, which must verify Condition C2 of Theorem 2; in 6.1 we show that given an $\mathsf{a}$-admissible order $\succcurlyeq$ and a query clause $Q$, it is always possible to introduce a context order satisfying Condition C2 with respect to $Q$.

Soundness and completeness of Algorithm 1 are a consequence of Theorems 1 and 2, but they depend on the initialisation of the context structure, as well as the

parameters chosen in Step A3. Given a context structure $\mathcal{D}$ and an expansion strategy $\mathsf{strat}$, let $\omega(\mathcal{D}, \mathsf{strat})$ be the maximum number of contexts in a context structure derivable from $\mathcal{D}$ using rules from Table 5.1 and 5.2 with the expansion strategy $\mathsf{strat}$ and an arbitrary depth limit $\Lambda$. Observe that $\omega(\mathcal{D}, \mathsf{strat})$ may not exist: for instance, if $\mathsf{strat}$ introduces a new context each time the $\mathsf{Succ}$ rule is triggered, the algorithm will introduce infinitely many contexts if $\mathcal{D}$ is initialised with a context with core $A(x)$ and $\mathcal{O}$ contains a clause of the form $A(x) \to A(f(x))$. On the other hand, in the case of the eager strategy introduced in Section 5.2, it is easy to see that $\omega(\mathcal{D}, \mathsf{eager})$ is smaller or equal than the number of contexts in $\mathcal{D}$ plus the number of subsets of $\mathsf{Su}$ for $\mathcal{O}$, and it is independent of the selected depth limit $\Lambda$.

**Corollary 40** (Soundness and Completeness)**.** *Let* $\mathbf{Q}$ *be a query, and let* $\mathbf{Q}^+$ *be the output of Algorithm 1 on input* $\mathbf{Q}$*, where* $\mathcal{D}_0$ *is the context structure in Steps A1 and A2, and* $\Lambda \geq \tau \cdot \omega(\mathcal{D}_0, \mathsf{strat})$*. For each* $Q \in \mathbf{Q}$*, we have that* $Q \in \mathbf{Q}^+$ *if and only if* $\mathcal{O} \models Q$*.*

*Proof.* Assume that $Q \in \mathbf{Q}^+$ and let $\mathcal{I}$ be a model of $\mathcal{O}$. Let $\mathcal{D}$ be the saturated context structure obtained after Step A4. Let $\mathcal{J}$ be the extension of $\mathcal{I}$ to $\Sigma_u$ defined inductively as given next:

- $o_\epsilon^{\mathcal{J}} = o_\epsilon^{\mathcal{I}}$;
- $o_\rho^{\mathcal{J}}$ with $\rho = S^i_{B_2} \cdot \rho'$ is defined as follows: if there are exactly $m$ domain elements $a_1, \dots a_m$ in $\mathcal{I}$ such that $(o_{\rho'}^{\mathcal{I}}, a_i) \in S^{\mathcal{I}}_{B_2}$ for each $1 \leq i \leq m$, then we define $o_\rho^{\mathcal{J}} = a_i$ if $1 \leq i \leq m$, and we define $o_\rho^{\mathcal{J}}$ arbitrarily if $i > m$. Otherwise, i.e. if no such elements exist or if there exist infinitely many, $o_\rho^{\mathcal{J}}$ is defined arbitrarily.

Clearly, $\mathcal{J} \models \mathcal{O} \cup \mathcal{C}_{\mathcal{D}}$, with $\mathcal{C}_{\mathcal{D}}$ as in Definition 9. Thus, by Theorem 1, we have $\mathcal{J} \models Q$, and since $\mathcal{J}$ is a conservative extension of $\mathcal{I}$, and $Q$ does not contain auxiliary constants, we have $\mathcal{I} \models Q$ and thus $\mathcal{O} \models Q$.

Finally, assume $\mathcal{O} \models Q$. Let $q$ be the context chosen in Step A2 for $Q$, and let $\mathcal{D}$ be the saturated context structure. Notice that $\mathcal{D}$ is derived from context structure $\mathcal{D}_0$, which contains no auxiliary constants, and if $\omega$ is the total number of contexts in $\mathcal{D}$, we have $\omega(\mathcal{D}_0, \mathsf{strat}) \geq \omega$, so $\Lambda \geq \tau \cdot \omega(\mathcal{D}_0, \mathsf{strat}) \geq \tau \cdot \omega$, and Theorem 2 is applicable, and we can conclude $Q \,\hat{\in}\, \mathcal{S}_q$. Step A5 of Algorithm 1 then ensures $Q \in \mathbf{Q}^+$. □

Termination of Algorithm 1 also depends on the chosen expansion strategy. As already mentioned, if $\mathsf{strat}$ introduces a new context each time the $\mathsf{Succ}$ rule is triggered, the algorithm may not terminate. Termination is, however, guaranteed whenever the

expansion strategy can only introduce finitely many contexts on any input. Furthermore, if the number of contexts generated by the strategy can be bound exponentially in the size of $\mathcal{O}$, and we ensure $\Lambda \leq \tau \cdot \omega(\mathcal{D}_0, \mathsf{strat})$ for any context structure $\mathcal{D}_0$ created according to Steps A1 and A2, we obtain a triple exponential upper bound on the complexity of the algorithm. It is easy to see that both the trivial and eager strategies introduce at most exponentially many contexts for any context structure initialised according to Steps A1 and A2. Such triple exponential complexity bound is in line with the bounds obtained for the tableau and resolution-based procedures discussed in Section 7.3 [75, 55]. We refer the reader to Section 7.4 for a proof of Theorem 5.

**Theorem 5** (Termination)**.** *Algorithm 1 is terminating if the strategy* strat *selected in Step A3 introduces a finite number of contexts. Furthermore, Algorithm 1 runs in worst-case triple exponential time in the size of $\mathcal{O}$ if the strategy* strat *selected in Step A3 introduces at most exponentially many contexts on the size of $\mathcal{O}$, and the parameter $\Lambda$ selected in Step A3 is smaller or equal than $\tau \cdot \omega(\mathcal{D}_0, \mathsf{strat})$ for any context structure $\mathcal{D}_0$ created in Steps A1 and A2.*

Observe that we can obtain a complete algorithm that satisfies the triple exponential complexity bound if we choose $\Lambda$ equal to the maximum value of $\omega(\mathcal{D}_0, \mathsf{strat})$ for context structures $\mathcal{D}_0$ created according to Steps A1 and A2. Our algorithm is therefore not worst-case optimal for $\mathcal{ALCHOIQ}^+$, which is an NExpTime-complete logic [85]. This is due to the fact that the Nom rule can introduce a doubly exponential number of auxiliary constants in the size of the input ontology. If, however, rule Nom is not triggered at any point during saturation, Algorithm 1 will run in worst-case exponential time if the strategy introduces at most exponentially many contexts. Rule Nom fires only when an inverse role interacts with a number restriction and a nominal simultaneously; hence, Algorithm 1 is worst-case optimal for fragments of $\mathcal{ALCHOIQ}^+$ which lack either inverse roles, number restrictions, or nominals. Furthermore, in the case of $\mathcal{ALCHIQ}^+$ ontologies, the inferences of our calculus mimic those of the $\mathcal{ALCHIQ}^+$ calculus in [20]. By carefully choosing the expansion strategy, the algorithm can also be worst-case exponential (and therefore optimal) for classification in the DL $\mathcal{ELH}$.

**Theorem 6** (Pay-as-you-go Behaviour)**.** *Let* strat *be an expansion strategy introducing at most exponentially many contexts on any input. If Step A3 selects* strat*, then Algorithm 1 runs in exponential time in the size of $\mathcal{O}$ if it is either $\mathcal{ALCHIQ}^+$,*

*$\mathcal{ALCHOQ}$, or $\mathcal{ALCHOI}$; furthermore, if $\mathcal{O}$ is $\mathcal{ELH}$, the algorithm runs in polynomial time with the eager strategy.*

We next show how, by adjusting the initialisation parameters, we can exploit Algorithm 1 to solve each of the DL reasoning tasks presented at the beginning of this section.

To solve consistency checking, it suffices to set $\mathbf{Q} = \{\top \rightarrow \bot\}$. Since query clause $\top \rightarrow \bot$ is inconsistent, $\mathcal{O} \models \top \rightarrow \bot$ if and only if $\mathcal{O}$ is also inconsistent, and the algorithm returns $\mathbf{Q}^+ = \emptyset$ if and only if $\mathcal{O}$ is consistent.

To check whether $C$ is subsumed by $D$ with respect to $\mathcal{O}$, we can use Algorithm 1 directly in the obvious way provided that both $C$ and $D$ are named concepts. Otherwise, we extend $\mathcal{O}$ with the normalisation (and clausification) of axioms $C \sqsubseteq B_C$ and $B_D \sqsubseteq D$ to obtain $\mathcal{O}'$, where $B_C$ and $B_D$ are fresh and in $\Sigma_A$. This leads to a polynomial increase in the size for $\mathcal{O}$. Finally, we run Algorithm 1 on $\mathcal{O}'$ and query $\{B_C(x) \rightarrow B_D(x)\}$ since $\mathcal{O} \models C \sqsubseteq D$ if and only if $\mathcal{O}' \models B_C \sqsubseteq B_D$.

To solve classification, we run Algorithm 1 with the query $\mathbf{Q} = \{B_i(x) \rightarrow B_j(x) \,|\, \{B_i, B_j\} \subseteq \mathbf{B}\}$. The result $\mathbf{Q}^+$ consists of all subsumptions of the form $B_i(x) \rightarrow B_j(x)$ entailed by $\mathcal{O}$, and it can be used to build a taxonomy of $\mathbf{B}$. To improve performance, it is often helpful to introduce a single context $v_i$ with core $B_i(x)$ for each $B_i \in \mathbf{B}$ in Step A2 and ensure that the corresponding context order $\succ_i$ makes all atoms of the form $B_j(x)$ for each $B_j \in \mathbf{B}$ pairwise incomparable.

To compute all instances of a (possibly complex) concept $C$, we extend $\mathcal{O}$ with the normalisation and clausification of axiom $C \sqsubseteq B_C$, where $B_C$ is fresh and in $\Sigma_A$. We then introduce a fresh unary $B_o$ for each $o_\epsilon \in \Sigma_u$, and extend $\mathcal{O}$ also with clauses

$$\mathcal{C}_u = \bigcup_{o_\epsilon \in \Sigma_u} \{\, B_o(x) \rightarrow x \approx o,\ \top \rightarrow B_o(o) \,\}$$

to obtain $\mathcal{O}'$. We can then run Algorithm 1 on $\mathcal{O}'$ and query $\mathbf{Q} = \{B_o(x) \rightarrow B_C(x) \,|\, o_\epsilon \in \Sigma_u\}$ since $\mathcal{O}' \models B_o(x) \rightarrow B_C(x)$ if and only if $\mathcal{O} \models C(o)$. To recover the instances of $C$, output each constant $o$ such that $B_o$ is mentioned in $\mathbf{Q}^+$.

Finally, to solve realisation, we extend $\mathcal{O}$ with the aforementioned clauses $\mathcal{C}_u$ to obtain $\mathcal{O}'$ and run Algorithm 1 on $\mathcal{O}'$ and $\mathbf{Q} = \{B_o(x) \rightarrow B(x) \,|\, o_\epsilon \in \Sigma_u, B \in \mathbf{B}\}$. It holds that $\mathcal{O} \cup \mathcal{C}_u \models B_o(x) \rightarrow B(x)$ if and only if $\mathcal{O} \models B(o)$; hence, for each $o_\epsilon \in \Sigma_u$, we know that $B_o(x) \rightarrow B(x)$ is in the output $\mathbf{Q}^+$ if and only if $B$ realises $o$. We can use these results to construct the set $\mathbf{B}_o^+$. Additionally, we can take into account the requirement that for every $B_1 \in \mathbf{B}_o^+$ there must be no $B_2 \in \mathbf{B}_o^+$ such that $\mathcal{O} \models B_2(x) \rightarrow B_1(x)$. To do so, we extend $\mathbf{Q}$ with the set $\{B_i(x) \rightarrow$

$B_j(x) \mid \{B_i, B_j\} \subseteq \mathbf{B}\}$, so that we can use $\mathbf{Q}^+$ to compute the taxonomy of $\mathbf{B}$, exactly as we did for classification. The taxonomy of $\mathcal{O}$ can then be used to remove from $\mathbf{B}_o^+$ all concepts that subsume other concepts in $\mathbf{B}_o^+$.

It is important to note that a saturated context structure $\mathcal{D}$ used to solve any of these tasks can be re-used to solve other tasks, in order to reduce the computational workload. For this, one can simply run Algorithm 1 by replacing the initialisation of a new context structure in Step A1 with the saturated context structure $\mathcal{D}$ from the previous task. Then, instead of introducing a new context in Step A2, once can simply choose one of the contexts that already exist and relax the context order to ensure that Condition C2 of Theorem 2 is satisfied. Soundness of this method is guaranteed because Theorem 1 does not depend on the initialisation of the context structure, and the conditions in the completeness theorem can always be satisfied for a new query by relaxing the context order of $\mathcal{D}$.

## 7.2 Modified Algorithm for Horn Ontologies and Single Subsumptions

The algorithm presented in the previous section runs in triple exponential time if $\mathcal{O}$ is in Horn-$\mathcal{ALCHOIQ}^+$; this is so because the $\mathsf{Nom}$ rule can fire. Furthermore, the algorithm runs in exponential time for $\mathcal{ELHO}$ ontologies, even with the eager strategy, because the number of ground atoms in the body of context clauses can be exponential in the size of the signature. In order to achieve worst-case optimal behaviour for such cases we introduce in this section a variant of Algorithm 1 which is based on the calculus from Section 5.4 for Horn ontologies. Our algorithm only works for queries of the form $\{K \to \Delta_1, K \to \Delta_2, \cdots, K \to \Delta_n\}$ having the same body $K$ consisting of a conjunction of unary atoms of the form $B(x)$.

In order to ensure that the context structure generated by the algorithm is sound for $K$, Algorithm 2 must initialise the context structure with just a single context $v$ other than $v_r$, and set $\mathsf{core}_v = K$. Furthermore, the $\mathsf{eager}$ strategy is fixed as the expansion strategy of Algorithm 2. This ensures that any context structure derived by the algorithm preserves soundness for $K$.

**Corollary 41** (Soundness and Completeness)**.** *Suppose that $\mathcal{O}$ is Horn. Let $\mathbf{Q}$ be a query of the form $\{K \to \Delta_{Q_1}, \cdots, K \to \Delta_{Q_n}\}$, where $K$ is a conjunction of atoms of the form $B(x)$ with $B \in \Sigma_A$, let $\mathsf{Su}_\tau$ be the number of atoms in $\mathsf{Su}$ of the form $S(y, x)$, and let $\mathbf{Q}^+$ be the output of Algorithm 2 on input $\mathbf{Q}$, where $\mathcal{D}_0$ is the context*

**Algorithm 2** Given $\mathbf{Q}$ of the form $\{K \to \Delta_{Q_1}, \cdots, K \to \Delta_{Q_n}\}$, where $K$ is a conjunction of atoms of the form $B(x)$ with $B \in \Sigma_A$, this algorithm decides whether $\mathcal{O} \models K \to \Delta_{Q_i}$ for each $Q_i \in \mathbf{Q}$, for a Horn ontology $\mathcal{O}$.

A1. Create a context structure $\mathcal{D}$ with $\mathcal{V} = \{v_r\}$, $\mathcal{E} = \emptyset$, and $\mathcal{S}_{v_r} = \emptyset$. This step includes the selection of an $\mathsf{a}$-admissible order $\succcurlyeq$ and a choice of context order $\succ_{v_r}$ compatible with $\succcurlyeq$.

A2. For each $Q_i \in \mathbf{Q}$:

- If $i = 1$ add a context $v_1$ to $\mathcal{D}$ with $\mathsf{core}_{v_1} = K$, and define the order $\theta(v_1) = \succ_{v_1}$ satisfying Condition C2 of Theorem 4 w.r.t. $\Delta_{Q_1}$.
- If $i \neq 1$, select context $v_1$ and weaken the context order $\succ_{v_1}$ so that it satisfies Condition C2 of Theorem 4 w.r.t. $\Delta_{Q_i}$,

A3. Select a depth limit $\Lambda$ for nominal labels.

A4. Apply the inference rules from Table 5.3, Table 5.1 minus Hyper, and Table 5.2 minus $r$-Succ, to $\mathcal{D}$ and $\mathcal{O}$, with the eager expansion strategy, until no new inferences are possible.

A5. Return $\mathbf{Q}_+ = \{Q_i \in \mathbf{Q} \mid \Gamma_i \to \Delta_i \,\hat{\in}\, \mathcal{S}_{v_i}\}$.

*structure in Steps A1 and A2, and* $\Lambda \geq |\mathsf{Su}_\tau|^2 \cdot \omega(\mathcal{D}_0, \mathsf{strat})$. *For each* $Q \in \mathbf{Q}$, *we have that* $Q \in \mathbf{Q}^+$ *if and only if* $\mathcal{O} \models Q$.

*Proof.* Assume that $Q \in \mathbf{Q}^+$ and let $\mathcal{I}$ be a model of $\mathcal{O}$. The proof of soundness is analogous to that of Corollary 40: we simply observe that if $\mathcal{J} \models K \to \bot$, then $\mathcal{O} \models Q$ trivially, and otherwise we can apply Theorem 3 to obtain $\mathcal{J} \models Q$. Similarly, the proof of completeness is analogous to that in Corollary 40, because $\mathcal{D}_0$ is clearly sound for $K$ by Theorem 3, and therefore $\mathcal{D}$ is sound for $K$, so Theorem 4 is applicable. □

The termination and relevant complexity results for Horn DLs are stated in Theorem 7; the proof is given in Section 7.4.

**Theorem 7** (Termination & Pay-as-you-go Behaviour)**.** *Algorithm 2 is terminating if the strategy* strat *selected in Step A3 introduces a finite number of contexts. Furthermore, if the parameter* $\Lambda$ *selected in Step A3 is smaller or equal than* $|\mathsf{Su}_\tau|^2 \cdot \omega(\mathcal{D}_0, \mathsf{strat})$ *for the context structure* $\mathcal{D}_0$ *created in Steps A1 and A2, then Algorithm 2 runs in worst-case exponential time. Finally, if* $\mathcal{O}$ *is* $\mathcal{ELHO}$ *and the* Hyper *rule is applied eagerly, then Algorithm 2 runs in polynomial time.*

It is easy to see that Algorithm 2 can be used to solve consistency checking and concept subsumption in the same way as Algorithm 1, since the queries in these

problems contain only a single subsumption each. To solve classification, however, we need to run the algorithm once for each concept $B \in \mathbf{B}$, using the query $\{B(x) \rightarrow B'(x) \mid B' \in \mathbf{B}\}$. Similarly, for instance retrieval and realisation, the algorithm must be run once for each $o_\epsilon \in \Sigma_u$ (plus once for each $B \in \mathbf{B}$, in the case of realisation, if we require that concept names in the output should be as specific as possible). Therefore, even though Algorithm 2 is worst-case optimal, it may not be faster in practice, as it cannot reuse context structures across subsumption tests.

Hence, there exists a trade-off between worst-case optimal complexity for classification of Horn ontologies with nominals (as achieved by Algorithm 2), and one-pass classification and maximal re-use context structures (as achieved by Algorithm 1). Practical considerations should dictate the choice between Algorithm 1 and Algorithm 2 when using our consequence-based approach for solving DL reasoning problems in Horn ontologies with nominals. In particular, we have observed that most $\mathcal{ELHO}$ ontologies contain a small number of nominals or use them trivially, and hence it pays off to use Algorithm 1 for one-pass classification despite the fact that the algorithm is worst-case exponential.

## 7.3 Discussion

In this section, we discuss the relationship between our calculus and other CB calculi for DLs with nominals, as well as with resolution and tableau calculi for $\mathcal{SHOIQ}$.

### 7.3.1 Relation to CB Calculi for $\mathcal{ELHO}$ and Horn-$\mathcal{SROIQ}$

The way in which our calculus handles nominals differs from the approach used in the $\mathcal{ELHO}$ and Horn-$\mathcal{SROIQ}$ calculi from [72] and [105], respectively. These calculi address the problem of non-local clauses discussed in Section 4.2 by explicitly deriving non-local consequences. For instance, the $\mathcal{ELHO}$ calculus can derive consequences of the form $G : C \sqsubseteq D$ to represent the fact that a subsumption $C \sqsubseteq D$ holds in all models of the input ontology $\mathcal{O}$ where the interpretation of concept $G$ is non-empty. In this approach, clause (31) from Section 4.2 would be represented as $A_1 \sqcap C : A_2 \sqsubseteq B$.

Deriving non-local consequences makes it easier to perform inferences involving arbitrarily distant contexts. For instance, the nominal rule $\mathsf{R}_{\{\}}$ in the $\mathcal{ELHO}$ calculus uses non-local premises $G \rightsquigarrow C$ and $G \rightsquigarrow D$, which represent the claim that the interpretation of $C$ (resp. $D$) is non-empty in every model where the interpretation of $G$ is non-empty, together with two other non-local premises, $G : C \sqsubseteq \{o\}$ and $G : D \sqsubseteq \{o\}$, to derive $G : C \sqsubseteq D$. An analogous inference rule in our calculus would

come with significant overhead, as it might require discovering paths from a context $v_G$ with core $G(x)$ to contexts $v_C$ and $v_D$ with cores $C(x)$ and $D(x)$, respectively, and then resolving clauses of the form $G(u) \to x \approx o$ in $v_C$ and $v_D$ with a clause of the form $\top \to G(u)$ in $v_G$. The derivation of non-local consequences, however, leads to the loss of properties 1-3 discussed in Section 4.2. The authors in [72] already point out the difficulties of reusing consequences across different queries and avoiding the derivation of very similar clauses. As a result, their $\mathcal{ELHO}$ calculus cannot achieve one-pass classification. In contrast, our calculus can re-use derived consequences across multiple queries, and in particular Algorithm 1 can classify ontologies in a single run. As already discussed, however, this comes at the price of losing worst-case optimality for Horn ontologies with nominals. The variant of the calculus for Horn ontologies presented in Section 5.4 is similar to the calculus in [72] in that it allows us to recover worst-case optimality for $\mathcal{ELHO}$ at the cost of losing one-pass classification. We believe that the optimisation techniques in [72] targeted towards re-use of clauses across queries may also be applicable in our setting.

The discussion above also applies to the Horn-$\mathcal{SROIQ}$ calculus in [105], which only ensures refutational completeness and cannot achieve one-pass classification. In contrast, Algorithm 1 can classify a Horn-$\mathcal{SROIQ}$ ontology in a single run, albeit with triply exponential running time. As in the previous case, Algorithm 2 can be used to obtain a worst-case optimal subsumption procedure for Horn-$\mathcal{SROIQ}$ ontologies, but at the price of losing one-pass classification.

The Horn-$\mathcal{SROIQ}$ calculus in [105] must also deal with the problem discussed in Section 4.3 concerning the interaction between inverse roles, number restrictions, and nominals. For this, the calculus relies on an inference rule that is triggered whenever an instance of a concept $C$ is connected via an inverse-functional role to an instance of a concept subsumed by a nominal. The rule derives a consequence $\mathsf{same}(C,C)$, which means that $C$ must behave like a nominal in any model of the ontology; this amounts to introducing a new nominal $\{a_C\}$ to represent the single instance of $C$ in each model where $C$ is non-empty. This approach is closely related to ours: on Horn-$\mathcal{SROIQ}$ ontologies, an application of the $\mathsf{Nom}$ rule from our calculus also introduces a single nominal. The main difference lies in the label of the new nominal: the calculus in [105] uses the name of the concept that collapsed, while we use the names of the linked constant and the role establishing the link.

### 7.3.2 Relation to Tableau and Resolution Calculi for $\mathcal{SHOIQ}$

When equipped with the trivial strategy, our calculus bears a resemblance to the resolution calculus in [75]. This calculus also ensures termination by strongly constraining the form of derivable inferences, and it also relies on paramodulation to reason with nominals. In contrast to our calculus, however, it uses the non-deterministic *splitting* technique for guessing which ground atoms hold. Although non-determinism can help reduce the number of derived clauses, it makes it harder to reuse derived clauses across different queries. The calculus in [75] addresses the simultaneous interactions between inverse roles, "at-most" number restrictions, and nominals, by means of specialised inference rules which introduce new constants, similarly to our Nom rule. Furthermore, the form of the closures in [75] which block the application of such inference rules in order to ensure termination is analogous to the form of the context clauses that we define in Section 6.3.3 to show that $x$ is still made equal to a constant in cases where $\Lambda$ "blocks" the application of the Nom rule in our calculus.

Our calculus also presents similarities with the tableau procedure for $\mathcal{SHOIQ}$ introduced in [59]. The completeness proof of our calculus is based on the ability to use context clauses to build a counter-model of every target query that is not entailed by the ontology. Models built this way have a similar structure to the models built by the tableau algorithm: they consist of (i) a central "cloud" of arbitrarily interconnected nominals, which can be either present in the ontology or created during execution, (ii) a number of (possibly infinitely deep) trees of unnamed domain elements rooted in a nominal, and (iii) role links from elements in the tree-shaped parts to the nominals in the central cloud. The rule that introduces new nominals in the tableau algorithm is similar to our Nom rule; in particular, our labelling convention for artificial nominals closely resembles that used in the tableau algorithm. Our rule, however, is deterministic, as it introduces a (possibly larger than needed) set of new constants which can later be made equal; furthermore, derived consequences involving these constants affect all models of the ontology. In contrast, the tableau algorithm *guesses* a number of elements which satisfy the relevant number restriction, and introduces them only in a single model of the ontology.

## 7.4 Proofs of Termination & Pay-as-you-go Behaviour

This section presents our proof of Theorems 5 to 7, which state the termination and complexity results for the algorithms of Chapter 7.

**Theorem 5** (Termination)**.** *Algorithm 1 is terminating if the strategy* strat *selected in Step A3 introduces a finite number of contexts. Furthermore, Algorithm 1 runs in worst-case triple exponential time in the size of $\mathcal{O}$ if the strategy* strat *selected in Step A3 introduces at most exponentially many contexts on the size of $\mathcal{O}$, and the parameter $\Lambda$ selected in Step A3 is smaller or equal than $\tau \cdot \omega(\mathcal{D}_0, \mathsf{strat})$ for any context structure $\mathcal{D}_0$ created in Steps A1 and A2.*

*Proof.* Define $k$ as the total number of DL-clauses in $\mathcal{O}$; let $m_1$ be the maximum number of body atoms in a DL-clause in $\mathcal{O}$, and $m_2$ the maximum number of literals in a context clause during execution. Observe that the former is linear in $\mathcal{O}$ since the number of variables is fixed. Furthermore, only a finite number of constants can be introduced by the algorithm, since $\mathcal{D}_0$ does not contain auxiliary constants, and the depth of auxiliary constants introduced by Nom is bounded by $\Lambda$.

To prove the first claim of the theorem, simply notice that since the number of possible context clauses per context is bounded, then if the number of contexts that are introduced is finite, the total number of context clauses in the context structure is bounded. Observe that after an inference rule has been applied, the preconditions of such inference never satisfied again, for eliminating a clause of $\mathcal{D}$ is only possible if another clause that subsumes it is derived in the same context. This ensures that there are only finitely many possible inferences, and hence the algorithm terminates.

To prove the second claim, we assume that the context strategy introduces at most exponentially many contexts, and we bound the number of inferences for each rule within each context or pair of contexts. First, observe that $\tau$ is exponential in the size of $\mathcal{O}$, and therefore by definition of $\Lambda$, the maximum length of an auxiliary constant appearing in $\mathcal{D}$ is exponential in the size of $\mathcal{O}$. Hence, there are is at most a double exponential number of constants in $\Sigma_u$ introduced in $\mathcal{O}$. This ensures that $m_2$ is at most double exponential, since we can have cubically many atoms on the number of predicates and constants. Let $m$ be the maximum between $m_1$ and $m_2$. Let $\mathcal{P}$ be the number of context clauses that can be constructed using the symbols in $\Sigma_u$. Notice that this is at most exponential on the size of $\Sigma_A$, $\Sigma_S$, and the number of constants in $\Sigma_u$ introduced in $\mathcal{D}$; hence, it is triple exponential on the size of $\mathcal{O}$.

For rules Core, Eq, Ineq, Factor, Elim, Join, and Nom, the number of context clauses participating in each inference is fixed, and it is not greater than 2, so the total number of inferences per context is bounded by $\mathcal{P}^2 \cdot m^2$, and hence the total number of possible inferences using these rules is triple exponential on the size of $\mathcal{O}$. In the case of the rule Hyper, the number of inferences that can be carried out in each context is bounded by $k \cdot \mathcal{P}^{m_1} \cdot m_2^{m_1}$, so the total number of inferences remains triple exponential on the size of $\mathcal{O}$. For the Pred and $r$-Pred rules, given a pair of contexts, the number of possible inferences by each of these rules between each pair of contexts is bounded by $\mathcal{P} \cdot \mathcal{P}^{m_2} \cdot {m_2}^{m_2}$, which is still triple exponential, and hence the total number of such inferences is triple exponential in the size of $\mathcal{O}$. Finally, for the Succ and $r$-Succ rules, we have that in each context, the rule can be triggered each time a clause with a new function symbol is derived in the context, or when a clause that modifies the set $K_2$ is added to the context. Thus, the rule can be applied at most $\mathcal{P}^2 \cdot m_2^{2m_1}$ times, which is still triple exponential on the size of $\mathcal{O}$. □

**Theorem 6** (Pay-as-you-go Behaviour)**.** *Let* strat *be an expansion strategy introducing at most exponentially many contexts on any input. If Step A3 selects* strat*, then Algorithm 1 runs in exponential time in the size of $\mathcal{O}$ if it is either $\mathcal{ALCHIQ}^+$, $\mathcal{ALCHOQ}$, or $\mathcal{ALCHOI}$; furthermore, if $\mathcal{O}$ is $\mathcal{ELH}$, the algorithm runs in polynomial time with the eager strategy.*

*Proof.* Suppose $\mathcal{O}$ is an $\mathcal{ALCHIQ}^+$ ontology. We then have that if $\mathcal{D}$ has no occurrences of elements of $\Sigma_u$, then no inference rule will introduce a clause with elements of $\Sigma_u$. Hence, if we initialise the calculus according to Algorithm 1, the rule Nom will never be triggered, and therefore no constants of $\Sigma_u$ will appear other than original constants. This prevents the double exponential increase in the size of $m_2$ in the proof of Theorem 5, instead making $m_2$ linear. Therefore, following the same argument than the one given in this proof, we obtain that the number of total inferences in the context structure per rule is, at most, exponential in the size of $\mathcal{O}$. Furthermore, we see that since no elements of $\Sigma_u$ can appear in $\mathcal{D}$, then rules Join, $r$-Pred, $r$-Succ, and Nom will not be applied to the algorithm, and the remaining rules will never be applied to $v_r$, or the Eq rule to an equality $x \approx u$, so the inferences will correspond exactly to those performed by the calculus in [20]. This also implies that if $\mathcal{O}$ is $\mathcal{ELH}$, then the algorithm runs in at most polynomial size in the size of $\mathcal{O}$, since this result was proved for the $\mathcal{ALCHIQ}^+$ calculus in [20].

Next, consider the case where $\mathcal{O}$ is an $\mathcal{ALCHOI}$ ontology. Once again, it is easy to see that the Nom rule will not be triggered, since there are no clauses of the form

DL4 in the ontology. This prevents the double exponential increase bound for $m_2$, reducing it instead to a polynomial-size bound. Applying an argument analogous to that in the proof of Theorem 5 proves that the algorithm terminates in at most time exponential with the size of $\mathcal{O}$.

Now, suppose $\mathcal{O}$ is in $\mathcal{ALCHOQ}$ ontology. As in the previous cases, the $\mathsf{Nom}$ rule will not be triggered because no atom of the form $S(u, x)$ can be derived, and hence the $\mathsf{Nom}$ rule will not be triggered. As in the previous case, this reduces $m_2$ from being at most double exponential on the size of $\mathcal{O}$, to being polynomial in the size of $\mathcal{O}$, and we have already seen that this proves that the algorithm terminates in at most exponential time in the size of $\mathcal{O}$.

Finally, if $\mathcal{O}$ is an $\mathcal{ELHO}$ ontology, the $\mathsf{Nom}$ is clearly never triggered, so $m_2$ has a polynomial upper bound, and since we use the eager strategy, the argument from the proof of Theorem 5 yields that fact that the algorithm terminates in exponential time.

□

**Theorem 7** (Termination & Pay-as-you-go Behaviour)**.** *Algorithm 2 is terminating if the strategy* $\mathsf{strat}$ *selected in Step A3 introduces a finite number of contexts. Furthermore, if the parameter* $\Lambda$ *selected in Step A3 is smaller or equal than* $|\mathsf{Su}_\tau|^2 \cdot \omega(\mathcal{D}_0, \mathsf{strat})$ *for the context structure* $\mathcal{D}_0$ *created in Steps A1 and A2, then Algorithm 2 runs in worst-case exponential time. Finally, if* $\mathcal{O}$ *is* $\mathcal{ELHO}$ *and the* $\mathsf{Hyper}$ *rule is applied eagerly, then Algorithm 2 runs in polynomial time.*

*Proof.* The proof of termination is identical to that for Theorem 5, so we focus on proving the claims of pay-as-you-go behaviour.

If $\mathcal{O}$ is in Horn-$\mathcal{ALCHOIQ}^+$, and the eager strategy is used, at most an exponential number of contexts will be derived. Furthermore, in this case $\Lambda$ is polynomial, rather than exponential; therefore, the upper bound on the size of $m_2$ will be exponential, rather than double exponential. Furthermore, consider the argument used for the proof of Theorem 5, and observe that since $\mathcal{D}$ is sound for some $K$, where $K$ is the conjunction in the body of all query clauses, context clauses have empty bodies. This ensures that only a linear number of clauses participate in each inference by the $\mathsf{Pred}$ and $r$-$\mathsf{Pred}$ rules, so following the same argument we conclude that the algorithm terminates in time at most exponential on the size of $\mathcal{O}$.

Finally, if $\mathcal{O}$ is $\mathcal{ELHO}$, the bound on $m_2$ is once again polynomial. Next, observe that due to the form of the clauses in $\mathcal{O}$, we have that in each context, the first time that the $\mathsf{Succ}$ rule is applied with some $S(y, x) \in K_1$, for $S \in \Sigma_S$, we have that

$K_1$ is either of the form $\{S(y,x)\}$ or $\{S(y,x), B(x)\}$ for some $B \in \Sigma_A$. Then, the fact that the Hyper rule is applied eagerly implies that if Succ is applied again to the same context, then $K_1$ will be of the form $\{S'(y,x) \mid S(y,x) \to S'(y,x) \in \mathcal{O}\}$ or $\{B(x)\} \cup \{S'(y,x) \mid S(y,x) \to S'(y,x) \in \mathcal{O}\}$. This, together with the fact that the initial context structure only contains two contexts, results in the fact that only a polynomial number of contexts can be introduced, at most, by the algorithm. With these bounds on $m_2$ and the total number of contexts, and the bound on the number of clauses participating in inferences by the Pred and $r$-Pred rules from the previous paragraph, we can use the argument from Theorem 5 to conclude that the algorithm terminates in time at most polynomial. □

# Part III

# Implementation

# Chapter 8

# System Description

We have implemented the algorithm described in Chapter 7 as an extension of Sequoia[1], a CB reasoner for $\mathcal{SRIQ}$ presented in [19]. Our implementation has been released as version v.0.7. It supports the reasoning tasks described in Chapter 7 for $\mathcal{SROIQ}$ ontologies, and is available through the OWL API interface.

In Section 8.1 we overview the architecture and main features of Sequoia v.0.6, many of which carry over to the latest version. In Section 8.2 we describe the implementation of the new features which enable Sequoia to reason with nominals. Finally, in Section 8.3, we present several novel optimisations that have been incorporated in the latest version of the reasoner.

## 8.1 Overview of Sequoia v.0.6

Figure 8.1 summarises the architecture of Sequoia. The reasoner does not use the OWL API representations internally, so it includes a set of *OWL API Bindings* which can translate OWL elements back and forth between the two representations.

The *Sequoia Reasoning Engine* is the core component of the system. Sequoia v.0.6 accepts as input OWL 2 DL ontologies without nominals, ABoxes, datatypes and `HasKey` axioms. By default, Sequoia will throw an exception if it encounters any of these constructs in the input, but the reasoner can be configured to work in best-effort mode and ignore unsupported axioms.

The *Ontology Loader* converts OWL axioms to normalised DL-clauses. For this, it first encodes away role inclusion axioms (*RIA Encoding* phase) using a variant of the algorithm in [55], and then it applies the *Clausification* sub-routine for computing a set of DL-clauses using a variant of the structural transformation [104]. Like many

[1] `http://www.cs.ox.ac.uk/isg/tools/Sequoia/`

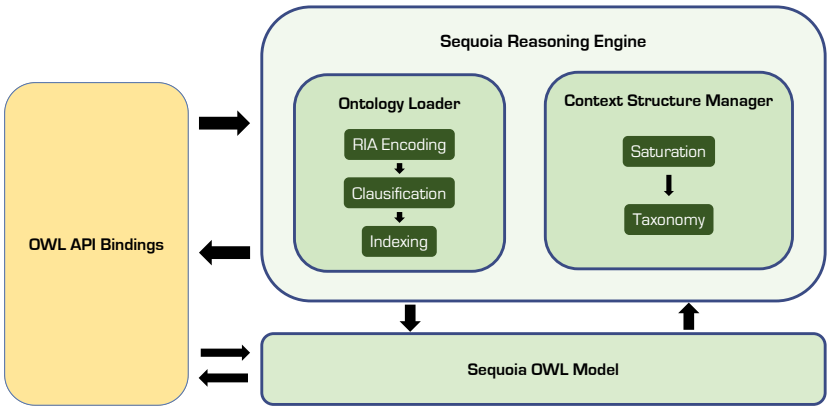



Figure 8.1: Architecture of Sequoia. Boxes represent key components, and arrows indicate flow of information.

theorem provers, Sequoia relies on indices to identify clauses that may participate in an inference or a simplification rule; however, Sequoia uses custom indexing techniques adapted to the specifics of ontology and context clauses. The *Ontology Indexing* phase is applied to DL clauses immediately after they are generated by the Clausification sub-routine.

Once an ontology $\mathcal{O}$ has been loaded, transformed, and indexed, the *Context Structure Manager* creates and saturates a context structure to classify the ontology. Following Step A2 of Algorithm 1, the Context Structure Manager initialises a context $v_B$ with core $B(x)$ for each named concept $B$ in $\mathcal{O}$, and it defines the context order for each context so that all named concepts (excluding auxiliary predicates) cannot be compared with each other. As discussed in Chapter 7, this condition is required for completeness. Following [20], the context order requires also that all that atoms with function terms are greater than atoms with no function terms, and that fresh concepts introduced during normalisation are smaller than those in the input ontology; these restrictions improve the performance of the reasoner.

Sequoia uses the *safe central* expansion strategy, defined as $\mathsf{safeCentral}(f, K_1, \mathcal{D}) = \langle v_{K_1}, K_1, \succ_{K_1} \rangle$ if $K_1$ has no named concepts, where $v_{K_1}$ and $\succ_{K_1}$ are as in the eager strategy; and $\mathsf{safeCentral}(f, K_1, \mathcal{D}) = \langle v_{\beta(K_1)}, \beta(K_1), \succ_{\beta(K_1)} \rangle$ otherwise, where $\beta$ takes the named concepts in $K_1$. This strategy strikes a balance between the eager and trivial strategies with respect to the number of contexts introduced.

Each context is implemented as a fibre. When a context $v$ is created, an empty

set $\mathcal{S}_v$ of context clauses is initialised, together with an auxiliary (empty) set of *unprocessed* clauses $\mathcal{U}_v$. Furthermore, a set $\mathcal{P}_v$ is created to hold clauses received from successors of $v$ that may be relevant for Pred inferences in $v$. After a context $v$ is created, Sequoia applies to $v$ the steps in Algorithm 3. The Ineq rule is applied as a simplification immediately after each new clause $C$ is derived. Furthermore, set $\mathcal{S}_v$ uses a redundancy index to ensure that no clause $C$ is added to $\mathcal{S}_v$ (and $\mathcal{U}_v$) if it is subsumed by a clause $C' \in \mathcal{S}_v$. Furthermore, clauses in $\mathcal{S}_v$ subsumed by $C$ are dropped when $C$ is added to $\mathcal{S}_v$. This implements the Elim rule. Since the derivation of new inferences within each context is independent from the state of other contexts, this part of the algorithm is easily parallelisable (see [20] for details).

By default, Sequoia uses all threads available in the system, but it can be configured to use a specific number of threads, if available. When all contexts are sleeping, the context structure has become saturated. Sequoia then runs the *Taxonomy* subroutine to read all relevant concept inclusions from the saturated context structure and compute their transitive reduction.

Sequoia v.0.6. encodes each predicate, term, and literal using an integer, called the *Unique Integer Identifier*, or UID for short. This yields a compact representation of clauses, and the natural ordering of numbers can be used as a term context order by representing $y$ as 0, $x$ as 1, and using number $i$ to represent function symbol $f_i$ in $\Sigma_f$ starting at $i = 2$. This also allows Sequoia to represent literals as 64-bit integers. The reasoner assumes that each sequence of 64 bits is partitioned at fixed positions into three subsequences: the first one represents the form of the literal (e.g., unary atom, equality, etc.), while the second and third correspond to the UIDs of all predicates or terms in the literal other than $x$. The first partition uses only 3 bits, and it is assumed that at most 30 bits are used to represent each UID corresponding to a predicate or term in the literal. Equalities and inequalities are represented as pairs, where the first element is strictly greater than the second. This compact representation is possible in the absence of nominals since at most two UIDs are necessary to express each context literal. This association between long integers and literal is also very helpful during redundancy checks.

To implement forward and backward redundancy checking, Sequoia maintains a redundancy index in each context inspired by *feature vector indexing* from the theorem prover E [120]. In every context, each clause is transformed into a distinct sequence of integers and sorted in ascending order. The result is inserted into a trie data structure (see [20] for details). With this encoding, forward redundancy for a clause $C$ can be checked by transforming $C$ into its corresponding integer sequence $\vec{s}_C$ and

**Algorithm 3** Sequence of actions performed on a context $v$.

P1. Apply the Core rule; add any resulting clauses to $\mathcal{U}_v$ and $\mathcal{S}_v$.

P2. Apply the Hyper rule with ontology clauses of the form $\top \to \Delta$; add all derived clauses to $\mathcal{U}_v$ and $\mathcal{S}_v$.

P3. While $\mathcal{U}_v$ is not empty:

   (a) Pick a clause $C$ from $\mathcal{U}_v$; let $\mathcal{L}$ be the set of maximal literals in $C$.

   (b) Apply all Hyper inferences involving a literal in $\mathcal{L}$, ontology clauses, and clauses in $\mathcal{S}_v$. Add inferences to sets $\mathcal{U}_v$ and $\mathcal{S}_v$.

   (c) Apply all inferences with the Eq and Factor rules involving a literal in $\mathcal{L}$ and clauses in $\mathcal{S}_v$; add inferences to sets $\mathcal{U}_v$ and $\mathcal{S}_v$.

   (d) Erase $C$ from $\mathcal{U}_v$.

P4. For each new clause added to $\mathcal{S}_v$, propagate any relevant inference(s) to the neighbour context as determined by the safe central strategy, using Succ. Clauses are propagated through communication channels between contexts. If a connection cannot be established with a context, this means that the context has not been created yet, so an appropriate context is initialised by the Context Structure Manager.

P5. For every new clause added to $\mathcal{S}_v$, if the clause may trigger an inference via Pred involving a predecessor context $w$, propagate this clause to the look-up set $\mathcal{P}_w$ in $w$.

P6. *Sleep* until a clause $C$ is received through an incoming communication channel. If this happens, then:

   (a) If $C$ has been propagated via Succ from a predecessor context $w$, perform the relevant substitution to obtain a clause $C'$. If $C'$ is redundant, propagate to the set $\mathcal{P}_w$ all clauses in $\mathcal{S}_v$ which can trigger an inference by Pred involving $w$. If $C'$ is not redundant, then add $C$ to $\mathcal{U}_v$ and $\mathcal{S}_v$, and proceed to Step 3.

   (b) If $C$ has been propagated via Pred from a successor context $w$, perform the relevant substitution to obtain a clause $C'$. Then carry out all hyperresolution inferences with $C'$ as a main premise and clauses of $\mathcal{S}_v$ as side premises. Add all conclusions to $\mathcal{U}_v$ and $\mathcal{S}_v$. Then, add $C'$ to $\mathcal{P}_v$ and proceed to Step 3.

traversing the tree downwards from the root while making sure that all node labels appear in $\vec{s}_C$ and in the same order. Similarly, for backward redundancy elimination, one can (recursively) traverse all downward paths from the root, which have $\vec{s}_C$ as a subsequence of their sequence of node labels. Bodies and heads of clauses are stored in a sorted form, which greatly speeds up redundancy checks.

We conclude this section by discussing the indexing techniques in Sequoia. These techniques exploit the special syntax of ontology and context clauses. For instance, the Hyper rule can be applied only to atoms of the form $B(x)$, $S(x, x)$, $S(x, *)$, and $S(*, x)$; where $*$ is distinct from $x$. These forms are called *unification patterns*, and Sequoia keeps indices of clauses in every context using these patterns as keys. Body atoms in ontology clauses are also restricted to these unification patterns; thus, whenever a clause $C$ with maximal literal $L$ is selected for application of the Hyper rule in a context $v$, Sequoia identifies the unification pattern of $L$ and retrieves only ontology clauses that contain a body atom with the same unification pattern using the ontology index. Conversely, once an ontology clause has been selected to participate in a Hyper inference, Sequoia identifies the unification pattern of each atom in the body of the ontology clause and then uses a context index to identify context clauses in $v$ with a maximal literal that has the same unification pattern. Similar techniques are used rules Pred and Eq; see [20] for details.

## 8.2 Implementing Support for Nominals

We next describe the modifications and extensions to the system described in Section 8.1 in order to implement support for nominals. Additionally, Sequoia v.0.7 supports all tasks discussed in Chapter 7, as well as novel features such as interruption, debugging mode, and extension of a loaded ontology with new axioms.

### 8.2.1 Representation and Ordering of Terms and Literals

Due to the presence of constants in context clauses, we now store 3 UIDs per literal (e.g., to represent atoms of the form $S(u_1, u_2)$). As a result, representing literals as 64-bit integers becomes impractical, and Sequoia v.0.7. represents terms and literals as standard objects. Redundancy indices still are represented as integer tries, but since literals are no longer associated to integers, each redundancy index must define its own mapping from literals to integers. Pairs of terms occurring together in an a-equality or inequality are no longer totally ordered; for instance, in $y \approx u$, with $u \in \Sigma_u$, we have that $y \succ_v u$ and $u \succ_v y$ are both forbidden by condition (4) of

Definition 5. Thus, we can no longer use a canonical representation of (in)equalities where the first term is guaranteed to be bigger than the second; instead, we require only that the first element is not smaller than the second.

As a consequence of these modifications, the handling of clauses has become significantly more involved in Sequoia v.0.7. For instance, since each redundancy trie defines its own correspondence from literals to integers, the bodies and heads of clauses must be sorted for each redundancy check. Similarly, clause subsumption becomes harder to check.

### 8.2.2 Fragmentation of the Root Context

Instead of using a single root context $v_r$, our implementation introduces a single *nominal context* $v_o$ with core $B_o(x)$ (see Chapter 7) for each individual $o \in \Sigma_u$. To ensure that all relevant inferences with Join can be carried out, clause $\top \rightarrow x \approx o$ is then added to this context, and we replace $o$ by $x$ in any clause added to this context via $r$-Succ. This allows us to use the Hyper rule as in any non-root context by unifying $x$ in the context clause with $x$ in the ontology clause. This method has several advantages. First, by splitting $v_r$ into a context $v_o$ for each nominal $o$, we can perform Hyper inferences for atoms of the form $B(o)$ in the corresponding context $v_o$ using the unification pattern $B(x)$ and the indexing strategy defined in Section 8.1 for all Hyper inferences with atoms of the form $B(o)$. Second, since constant $o$ is replaced by variable $x$, we reduce the number of UIDs required to represent literals in $v_o$ which contain $o$. This simplifies the application of many inference rules. Finally, the root context may yield many inferences, especially for ontologies with large ABoxes; ensuring that those inferences happen in different contexts makes the algorithm easier to parallelise.[2]

Sequoia implements an additional inference rule to propagate information between nominal contexts and preserve completeness: if a clause $\top \rightarrow L_o$ is derived in a nominal context $v_a$ with core $B_a(x)$ for $a \in \Sigma_u$, and $L_o$ contains a constant $o \neq a$ with $o \in \Sigma_u$, then clause $\top \rightarrow L_o[x/a][o/x]$ is added to $v_o$. The rules governing the interaction between non-root nodes and $v_r$ have also been modified so that information can be exchanged between non-root nodes and the relevant nominal nodes. For instance, if $r$-Succ is triggered in context $v$ by an atom of the form $B(o)$, the rule propagates

[2]The current version of Sequoia does not yet support parallel reasoning, but we believe that it should be relatively easy to add this feature in future versions. Some design decisions, such as this one, have already been made with that goal in mind.

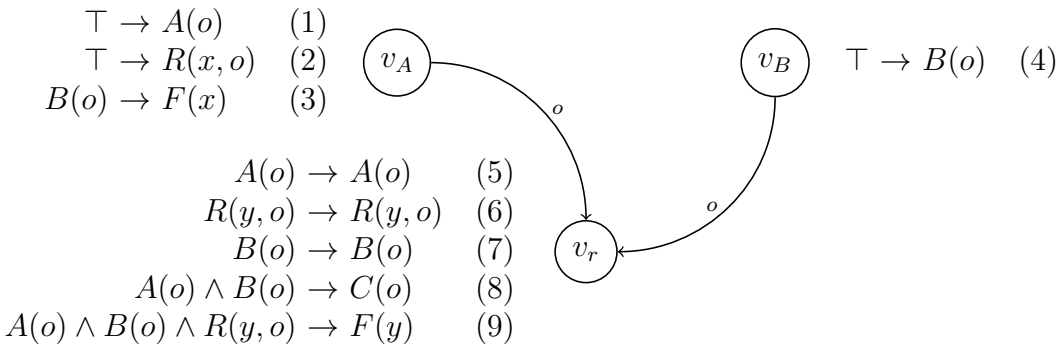



Figure 8.2: Example of irrelevant inference by $r$-Pred. Context $v_A$ has core $A(x)$, and context $v_B$ has core $B(x)$; context clauses are attached to the corresponding node.

clause $B(x) \to B(x)$ to nominal context $v_o$ (instead of $v_r$), and adds an edge from $v$ to $v_o$ (if required) labelled with $o$.

### 8.2.3 Implementation of Inference Rules

Rule Nom has been implemented in Step A3 together with other intra-context inference rules, just after A3c. In order to reduce the number of auxiliary constants introduced in the context structure, each application of Nom in context $v$ is preceded by a search for context clauses in $\mathcal{S}_v$ which would already force $x$ to be made equal to a constant. In particular, we search for a clause of the form $\Gamma \to \Delta \vee \Psi \in \mathcal{S}_v$ with $\Gamma \subseteq \Gamma_1 \cup \Gamma_2$, $\Delta \subseteq \Delta_1 \cup \Delta_2$, and such each literal $L \in \Psi$ is of the form $x \approx u$, $y \approx u$, or $x \approx y$, for some $u \in \Sigma_u$.

Rule Join is implemented just after Step P3a in Algorithm 3: after we have chosen $C$, we apply all inferences with Join that involve a literal $L \in \mathcal{L}$ and a clause in $\mathcal{S}_v$ with $L$ in the body. We have included an additional index in each context to find all clauses that contain a particular ground atom in the body.

Rules $r$-Pred and $r$-Succ have been implemented analogously to rules Pred and Succ in Section 8.1. We have observed in practice, however, that rule $r$-Pred often leads to the derivation of many irrelevant clauses. For instance, suppose that the context structure in Figure 8.2 has been obtained while classifying an ontology $\mathcal{O}$ that contains the following clauses:

$$A(x) \wedge B(x) \to C(x) \quad (10) \qquad C(x) \wedge R(z, x) \to F(z) \quad (11)$$

Clauses (1), (2) and (4) are propagated by $r$-Succ to clauses (5) through (7), respectively. Rule Hyper then generates clauses (8) and (9) from the previous clauses and clauses (10) and (11), and the latter is then propagated via $r$-Pred to $v_A$ as

clause (3). Since $v_A$ does not contain a clause with atom $B(o)$ in a maximal position, clause (3) (as well as all the clauses derived from it) become irrelevant. More generally, we have encountered that a large number of clauses of the form $A \wedge \Gamma \rightarrow \Delta$, with $A$ a ground atom, may be propagated to contexts $v$ such that neither $v$ nor any of its ancestors have a clause with $A$ in a maximal position in the head; such clauses and their consequences are irrelevant.

To prevent the generation of such clauses, we have strengthened the preconditions of rule $r$-Pred so that a clause $C$ in the root context may only be propagated to a non-root context $v$ via $r$-Pred if *all* body atoms appear as maximal atoms in heads of context clauses of $v$. In our example, this ensures that clause (9) is not propagated to $v_A$. To retain completeness, we also modify $r$-Succ such that, if ground atom $A$ is selected in a non-root context $v$, a clause $A \rightarrow A$ is then propagated to all its successors.

Finally, the application of the Eq rule in our calculus may involve equalities of the form $x \approx o$ with $o \in \Sigma_u$, and result in the replacement of $x$ with $o$ (or vice-versa) in other clauses. The completeness proof in 6.3, however, shows that such equalities are only required in contexts introduced upon initialisation based on the query, and are only necessary for paramodulation inferences on query atoms. Therefore, many of the aforementioned inferences are irrelevant and our implementation incorporates a mechanism for blocking them.

## 8.3 Optimisations

Next, we describe a suite of optimisation techniques that we have incorporated in Sequoia v.0.7.

### 8.3.1 Ordering of Query Atoms

As discussed in Chapter 7, classification requires that predicates in $\mathbf{B}$ are incomparable in context orders assigned to *query contexts* (contexts initialised based on the query). This has a negative impact on performance: $\mathbf{B}$ typically contains most ontology predicates, many inferences occur in query contexts, and the resolution of long context clauses with many predicates in $\mathbf{B}$ within a query context can lead to inefficient behaviour. Consider an ontology $\mathcal{O}_3$ which contains, among others, the following clauses:

$$A(x) \to A_1(x) \vee \cdots \vee A_n(x) \quad (12) \qquad A_i(x) \to B(f_i(x)) \quad 1 \leq i \leq n \quad (13)$$
$$B(x) \to \bot \quad (14)$$

Let $A_i \in \mathbf{B}$ for each $1 \leq i \leq n$, in which case no two atoms of in $\{A_i(x) \mid 1 \leq i \leq n\}$ can be compared in any query context. Consider a query context $q$ containing clause $\top \to A(x)$. We have $\mathcal{O} \models A(x) \to \bot$, so we expect to derive $\top \to \bot$ in $q$. Using clause $\top \to A(x)$, an inference by Hyper leads to:

$$\top \to A_1(x) \vee \cdots \vee A_n(x). \quad (15)$$

Since all atoms in the head of this clause are incomparable in $q$, Sequoia can select each $A_i(x)$ when the clause is processed, and therefore Algorithm 3 will add $n$ clauses to $\mathcal{U}_v$. It is possible that each of these $n$ clauses can be resolved in the same round with clauses of the form $B(f_i(x)) \to \bot$ in $\mathcal{P}_v$, derived via Pred from a successor context with a context clause $B(x) \to \bot$. In that case, for each $1 \leq i \leq n$, Sequoia will derive

$$\top \to A_1(x) \vee \cdots \vee A_{i-1}(x) \vee A_{i+1}(x) \vee \cdots \vee A_n(x) \quad (16)$$

Although clause (16) makes clause (15) redundant, clauses of that form stay in $\mathcal{U}_v$ and may produce further clauses. Each clause of this form will in turn lead to the derivation of $n - 1$ clauses, and by the time Sequoia derives $\top \to \bot$, it will have derived a clause for every subset of $\{A_1(x), \cdots, A_n(x)\}$, thus producing an exponential number of clauses.

To address this source of inefficiency, we take advantage of the fact that query clauses in classification tasks are Horn. If there are at least two query atoms in the head of the same context clause, then the clause will be irrelevant unless one of those atoms participates in an inference. Therefore, we can block application of inferences on any other query atoms occurring in the head of the same clause. This is exploited in Sequoia to define an order on query atoms which can be used to reduce the number of inferences during classification. In our previous example, Sequoia can derive $\top \to \bot$ in linear time; this pattern appears in several ontologies, which Sequoia v.0.7 can now classify.

### 8.3.2 Controlled Propagation of the ABox

The application of rule Hyper ensures that ground atoms $A$ occurring in ontology clauses of the form $\top \to A$ (corresponding to DL ABox assertions) are propagated to every context; however, such clauses are only used in a small number of inferences.

Such propagation of ground facts increases memory usage, and has proved problematic in practice for Sequoia. Clauses of this form can, however, still be useful for redundancy elimination. Indeed, clauses of the form $\Gamma \to \Delta \vee A$ can often be generated in many contexts, and it would be ideal to derive clauses of the form $\top \to A$ in such contexts, so that we can carry out redundancy checks.

Our approach to this issue has been to block the application of Hyper or $r$-Pred to ontology clauses of the form $\top \to A$, with $A$ a ground fact, in all contexts except nominal contexts. This ensures that ground facts are not propagated to arbitrary contexts, thereby reducing memory use and preventing the degradation of the performance of redundancy index checks. Later, if a clause of the form $\Gamma \to \Delta \vee A'$ is derived in a context $v$, where $A'$ is a ground atom that mentions a constant $o \in \Sigma_u$, Sequoia immediately applies Hyper to $v$ for all clauses of the form $\top \to A$ in $\mathcal{O}$ such that $A$ mentions $o$, to prevent the derivation in $v$ of redundant clauses mentioning $o$. This optimisation has been critical for performance in ontologies with large ABoxes.

Ontologies, however, often contain complex combinations of axioms that generate clauses of the aforementioned form, and which are not addressed by our optimisation. For instance, consider the following clauses.

$$\top \to A(o) \quad (17) \qquad A(x) \to R(x, f(x)) \quad (18)$$
$$A(x) \to B(f(x)) \quad (19) \qquad B(x) \to C(x) \quad (20)$$
$$R(z_1, x) \wedge C(x) \to D(z_1) \quad (21)$$

It is easy to check that any ontology that contains these clauses implies $\top \to D(o)$. However, by the time $\top \to D(x)$ has been generated in the nominal context for $o$, redundant clauses containing $D(o)$ have already been generated in other contexts.

To address this issue, we have divided the saturation phase in Algorithm 3 into two phases. In the first one (the *Horn phase*), inference rules are never applied to non-Horn context clauses; in the second one, all inference rules are applied as usual. Non-Horn clauses derived during the Horn phase are stored in the queue $\mathcal{U}_v$ defined in Algorithm 3, but they are blocked so that they will be skipped by Step P3. This allows Sequoia to seamlessly initiate the second phase from the state reached upon completion of the Horn phase.

Splitting the saturation phase can also prevent the derivation of irrelevant clauses in cases where no ground facts are involved. Indeed, the derivation of Horn clauses can help reducing the number of non-Horn clauses through redundancy elimination in certain ontologies containing combinations of axioms similar to the one described above.

# Chapter 9

# Evaluation

This section describes our evaluation of the performance of Sequoia on classification of real-world ontologies. In our experiments, we follow the methodology by Steigmiller et al. [132]. In Section 9.1 we describe the methodology of our evaluation, including the input dataset and the systems utilised. In Section 9.2 we summarise the experiment results and present a critical assessment of our evaluation.

## 9.1 Methodology

The corpus of ontologies selected for our experiment is the Oxford Ontology Repository, which consists of 799 ontologies.[1] Sizes of the ontologies range from ∼1KB to ∼1GB, and their expressiveness range from lightweight languages such as DL-Lite and $\mathcal{EL}$ to $\mathcal{SROIQ}(\mathcal{D})$. Ontologies are sourced from a diverse range of repositories, such as the Gardiner Corpus [44], the OBO Foundry [127], and the Phenoscape project [17]. We processed each ontology by performing the following steps in the given order.

- *Syntax Sanitisation.* We verified that each ontology could be successfully loaded by the OWL API v.3.5. If this was not the case, we manually edited the ontology to fix syntax errors.[2] We also eliminated all empty ontologies.
- *OWL 2 DL Profile Check.* We checked that each ontology satisfied the global restrictions of the OWL 2 DL language on the use of non-simple object properties, which ensure decidability. A total of 7 ontologies required removal of axioms violating these restrictions.

[1] `http://www.cs.ox.ac.uk/isg/ontologies/`

[2] Ontologies with ID 158, 160, 291, and 785 required editing due to syntax errors.

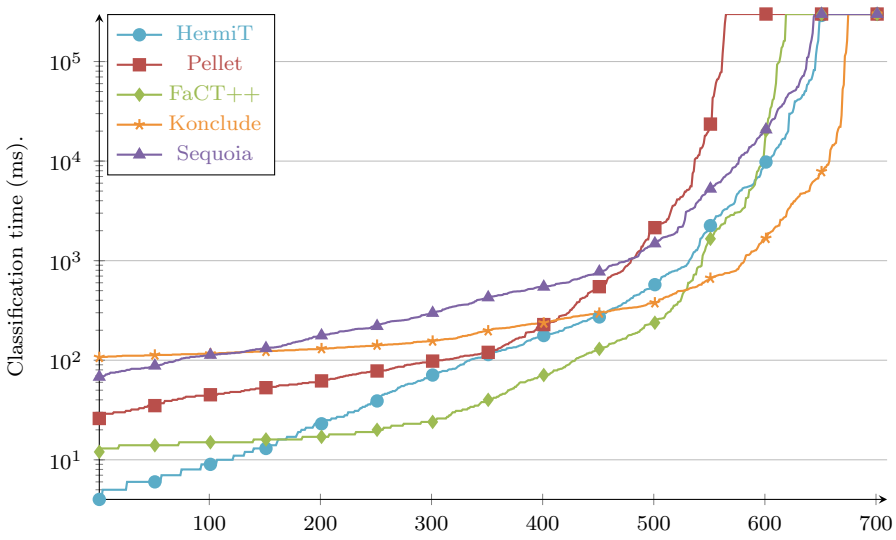



Figure 9.1: Classification Times for All Ontologies

- *Elimination of Datatypes.* Sequoia does not currently support datatype reasoning; thus, we followed the common approach where each datatype (or datatype restriction) is replaced with a fresh class, each data range intersection, union, complement, or enumeration with a class intersection, union, complement, or enumeration (respectively), each data property with a fresh object property, and each literal with a fresh named individual. Multiple occurrences of a datatype, data property, or literal were matched to the same fresh class, object property, or individual, respectively. We also removed all `HasKey` axioms.

- *ABox Elimination.* We removed the ABox in ontologies that do not already contain nominals in the TBox, which allowed us to focus our evaluation on classification. Indeed, it is well-known that, in the absence of TBox nominals, it is possible to eliminate the ABox from satisfiable knowledge bases without affecting the result of classification.

The result of this process is a corpus of 703 ontologies, out of which 97 contain nominals in the TBox.

We used version v.0.7 of Sequoia in our evaluation, which has been described in Chapter 8. The other reasoners used in the evaluation were HermiT 1.3.8, Konclude 0.6.2, FaCT++ 1.6.5, and Pellet 2.4.0. We ran all experiments on a Dell server with 512 GB of RAM and two Intel CPU E5-2640 V3 2.60 GHz processors, with eight cores per processor and two threads per core. The system OS was Fedora 26, kernel version 4.11.9-300.fc26.x86_64, and Java 1.8.0 update 151. To streamline the tests,

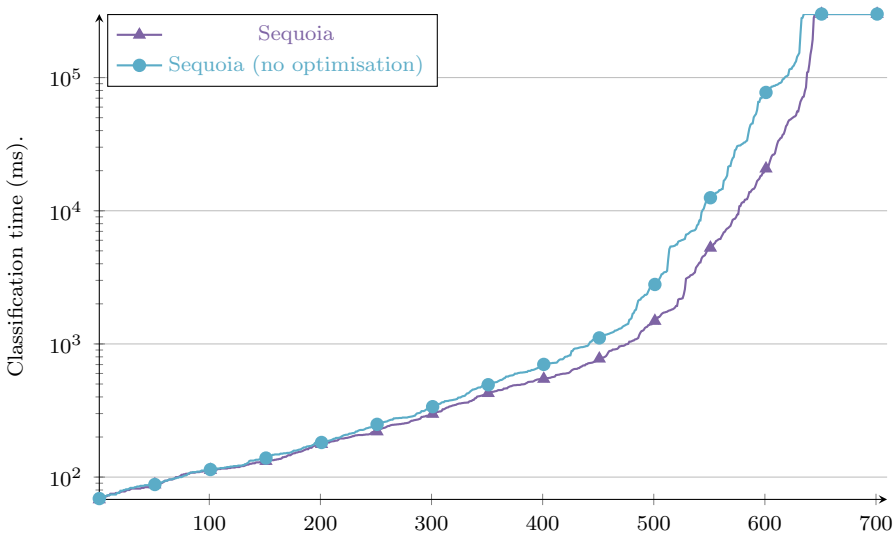



Figure 9.2: Classification Times for Sequoia, with and without Query Atom Ordering

we accessed all reasoners through the OWL API interface. We used versions 3 and 4 of the OWL API interface, since not every reasoner supported the same versions. Konclude does not offer direct access through the OWL API, so we accessed it via the OWLlink OWL API adapter. We do not expect this to have had a significant impact on performance, since axioms were loaded to the OWLlink server before the test started. The reasoner Konclude offers support for parallel reasoning; however, this feature is not yet available in the current version of Sequoia, so we compared all reasoners in single-threaded mode.

For each ontology in the processed corpus, and for each reasoner, we created a fresh process that used the OWL API to: (i) load the ontology, (ii) create a new instance of the reasoner, (iii) load the ontology to the reasoner, and (iv) ask the reasoner to classify the ontology. Step (iv) was allowed to run for 5 minutes; if this threshold was reached, the process was destroyed and we recorded a timeout. Otherwise, we created another fresh process to repeat Steps (i)-(iv) up to three times. We measured the wall-clock duration of the classification task (i.e. Step (iv)) in each repetition. If all repetitions finished before the timeout, we recorded the average classification time for that ontology; otherwise, we recorder the result as a failure. Taxonomies were hashed to verify correctness. All systems and ontologies used in the experiment are available online.[3]

[3] `http://krr-nas.cs.ox.ac.uk/2019/DL-sequoia/`

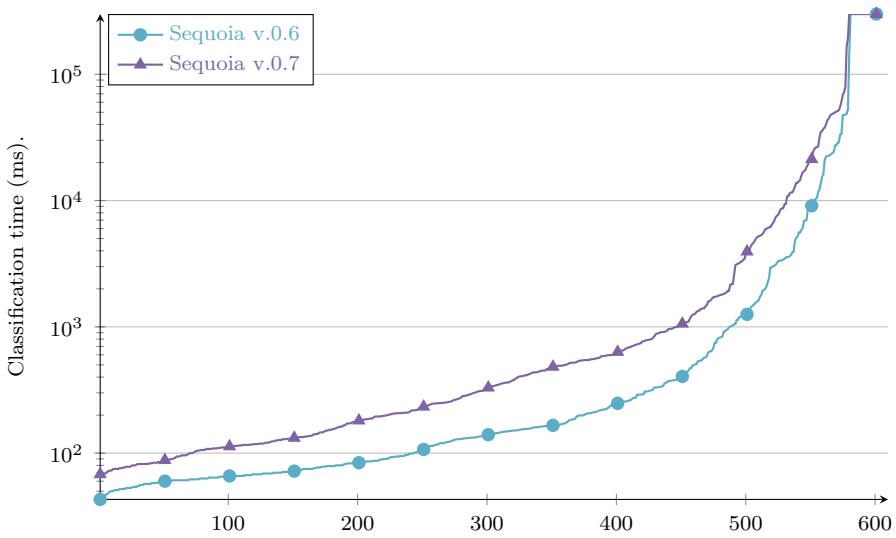



Figure 9.3: Classification Times for Sequoia v.0.7 and v.0.6. on Ontologies without Nominals

## 9.2 Results and Discussion

The (averaged) classification times for the full corpus are summarised in Figure 9.1. For each reasoner, we sorted the classification times in ascending order; a value point $(n, m)$ in the figure represents that the $n$-th smallest average classification time for that reasoner was $m$ milliseconds. Points where $m = 300s$ represent timeouts. Hash values for Sequoia agreed with the results for at least one other reasoner in almost all cases. When disagreement occurred, we manually verified that taxonomies were equivalent, and the discrepancies were due to outputting a different selection of (equivalent) entailments sets[4]. Please see the companion website for a full list of the test results and ontology hashes.

Figure 9.1 shows that the performance of Sequoia is competitive with that of well-established DL reasoners. Sequoia was the only reasoner capable of classifying ontologies with ID 472 (in 20s), and ID 483 (in ∼3min). Sequoia incurred a measurable overhead on ontologies with classification times up to 10s, which we believe is related to the management of the context structure and the communication between contexts. Most timeouts (38 out of 49) occurred on non-Horn ontologies, where saturation produced many clauses with large numbers of disjuncts in the head, which is a well-known source of performance problems in CB reasoning [20]. These problems

[4]We observed that reasoners do not agree on which entailment sets constitute a taxonomy; in particular, their outputs correspond to different categories of entailment sets among those listed in [16].

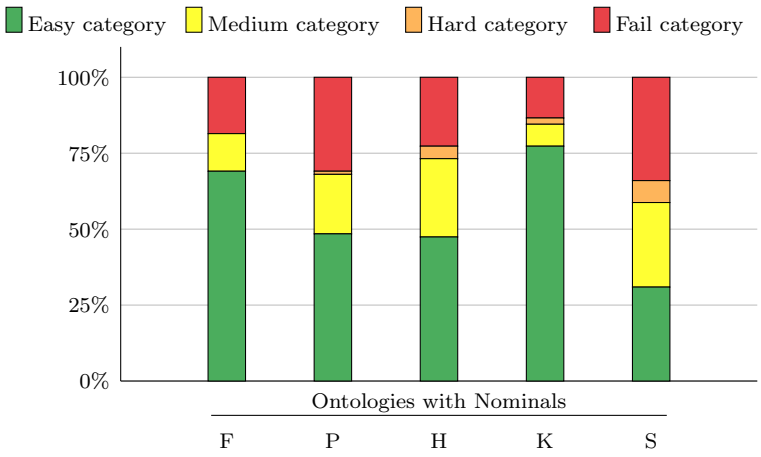



Figure 9.4: Percentage per Category of Easy (< 10s), Medium (< 1m), Hard (< 5m), and Fail (≥ 5 m) Ontologies with Nominals, for FaCT++ (F), Pellet (P), HermiT (H), Konclude (K), and Sequoia (S).

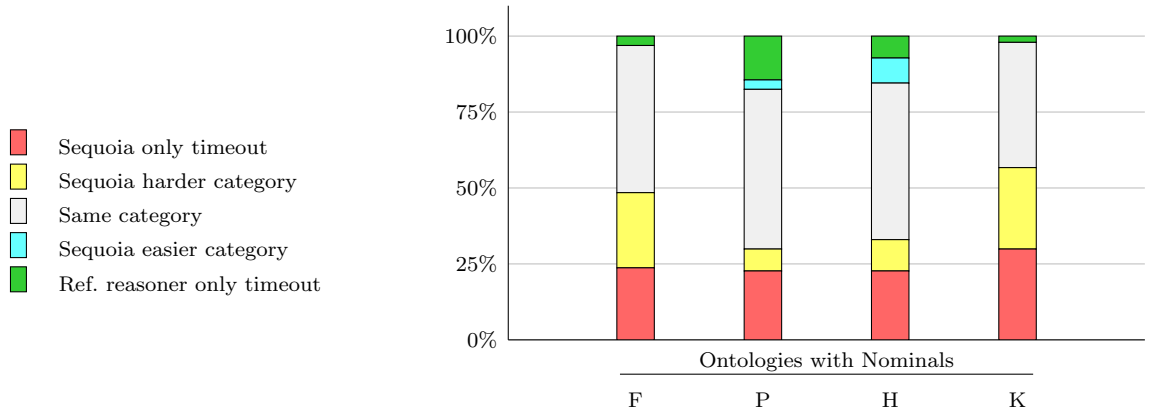



Figure 9.5: Pairwise Comparison of Overlap in Categories between Sequoia and each of the reasoners FaCT++ (F), Pellet (P), HermiT (H), and Konclude (K), for Ontologies with Nominals.

are exacerbated for ontologies where an "at-most" number restriction is applicable to a context with a large number of successors for the same role; this leads to the generation of an exponential number of clauses with quadratically many equalities in the head; this was, for instance, the case with the Pizza ontology (ID 799).

Although the overall performance of Sequoia and HermiT over the corpus appears to be similar, a more fine-grained analysis shows that a significant number of ontologies were easy for one reasoner, but very hard for the other. For instance, HermiT was able to classify 21 ontologies where Sequoia timed out. Of these ontologies, 8 were classified by HermiT in less than 1 second each; another ontology was classified in 41 seconds, and the rest were classified in approximately 5 to 10 seconds each. Con-

versely, Sequoia successfully classified 16 ontologies in which HermiT timed out. Four of these ontologies were classified in more than a minute by Sequoia; one additional ontology was classified in 1.5 seconds, and the remaining ontologies were classified in approximately 10 to 20 seconds. These disparities can be explained by the fact that the calculi underlying these reasoners are very different from each other. We see this as evidence that each of these reasoners has its unique strengths, which manifest in different types of input ontologies, and therefore they can be seen as complementary to each other.

Figure 9.4 considers the 97 ontologies with nominals in the TBox. For each reasoner, we further divided these ontologies into four categories: Easy (classification time $< 1s$), Medium (between $1s$ and $60s$), Hard (between $60s$ and $300s$), and Fail (timeout). Figure 9.4 shows that reasoning with nominals is still challenging for Sequoia, and further optimisation is needed. It is interesting to notice, however, that none of the ontologies required the generation of fresh nominals by the Nom rule during classification; this supports the widespread belief within the community that occurrences of simultaneous interaction of nominals, number restrictions, and inverse roles are rare. Sequoia's performance, however, was in line with that of the other reasoners on unit test ontologies that included a small number of interactions of this kind.

Figure 9.5 complements Figure 9.4 by showing the overlap between difficulty categories for Sequoia versus each of the other reasoners. In each column, we select a reasoner to compare against Sequoia; then, for the 97 ontologies with nominals, we show the percentage where only Sequoia timed out (red), where neither reasoner timed out but the classification time for Sequoia was in a harder category (yellow), where the times of both reasoners were in the same category (grey), where neither reasoner timed out but the time for Sequoia was in an easier category (cyan), and ontologies where only the reference reasoner timed out. The observations that we made in the general case for HermiT hold also when we consider only ontologies with nominals: HermiT can classify 21 ontologies where Sequoia times out, and it performs considerably better in 10 additional ontologies. Conversely, Sequoia could classify 7 ontologies where HermiT timed out, and it performed considerably better in 8 additional ontologies.

To measure the effect of the optimisations described in Section 8.3, we disabled them one by one and ran the resulting modified versions of Sequoia on the entire corpus. Furthermore, in order to observe the effect of the implementation overhead

described in Section 8.2.1, we ran Sequoia v.0.6 on the subset of ontologies in the corpus that do not contain nominals.

The optimisation described in Section 8.3.1 improved the classification time of most ontologies in the corpus. Moreover, it allowed Sequoia to classify 11 additional ontologies. This can be seen in Figure 9.2, which compares the average classification times of Sequoia with those of a modified version of the reasoner where this optimisation has been disabled. This result emphasised the importance of ordered resolution for restricting the number of inferences in our calculus. The optimisations described in Section 8.3.2 allowed Sequoia to classify 29 additional ontologies (e.g. ID (005), ID (078), ID (316), ID (507)); furthermore, classification times for ontologies that did not require these optimisations were not significantly affected. Not surprisingly, the effect of these optimisations manifested almost entirely on ontologies with nominals and ABoxes (27 out of 29).

Figure 9.3 compares the performance of Sequoia v.0.7 and Sequoia v.0.6 on ontologies that do not contain nominals. We can observe that v.0.7. incurs in some overhead due to a less efficient representation of terms, literals, and clauses (see the discussion in Section 8.2.1), as well as the increased difficulty of comparing clauses during redundancy checks. We also observe, however, that the effect of this overhead is reduced for hard ontologies; in fact, Sequoia v.0.7 can classify a higher number of ontologies than the previous version thanks to the additional optimisations.

Bate et al. [20] compared the performance of Sequoia v.0.6 with that of ELK on ontologies in the EL profile of OWL 2; their evaluation showed that the theoretical pay-as-you-go behaviour of the reasoner also translates into practice. None of the OWL 2 EL ontologies supported by ELK in the corpus contains nominals, and our evaluation on v.0.7 does not significantly alter the results in [20] despite the representation overhead in v.0.7. It is interesting to observe that Sequoia is the fastest amongst all the fully-fledged OWL 2 reasoners to classify the $\mathcal{ELH}$ version of SNOMED (ID 798); in particular, Konclude takes twice as long, and the other reasoners time out.

In summary, our implementation shows promising performance despite not being as mature as other well-established reasoners. Our approach has proved to complement nicely the performance of other reasoners due to underlying differences in their respective calculi. Reasoning with nominals or ontologies with number restrictions can still be challenging for Sequoia, which suggests promising directions for further optimisation. Furthermore, there appears to be plenty of room for optimisation via the design of more efficient data structures, representations, and indexing techniques.

# Chapter 10

# Conclusions and Future Work

In this chapter, we review the contributions of this thesis, and we look at possible future lines of research opened by the work that we have presented.

## 10.1 Summary

The two main contributions of this thesis have been the following:

- **The Calculus.** We have presented the first consequence-based reasoning algorithm for a DL featuring all Boolean operators, inverse roles, nominals, and number restrictions. Our calculus can be extended to $\mathcal{SROIQ}$ by means of standard transformations; this DL covers the semantic web language OWL 2 DL, with the exception of datatypes.

  We have first addressed the challenge of applying the consequence-based approach to reasoning, which is based on derivation of local clauses, to DLs with nominals, which allow us to impose global restrictions on models. We have developed a technique which allows ground atoms and individual names in context clauses, and we have produced suitable inference rules to exchange information between contexts that are not connected to each other via edges labelled by successor function symbols. We have also addressed the challenge that emerges when inverse roles, number restrictions, and nominals interact with each other. Our calculus introduces new constants to represent elements that behave like nominals in forest-like models of the ontology, and it establishes a bound on the number of such constants in order to ensure termination.

  The calculus has been built over the framework for CB reasoning for $\mathcal{ALCHIQ}^+$ presented by Bate and colleagues [20], so that it behaves precisely like this calculus when input ontologies do not contain nominals. More generally, we

have shown that our calculus exhibits pay-as-you-go behaviour: it is worst-case optimal for the proper fragments of $\mathcal{ALCHOIQ}^+$, and for the full logic except in those rare cases where disjunctions, nominals, number restrictions and inverse roles interact simultaneously. We have shown that the calculus retains the capacity to fully classify an ontology in a single pass. To ensure worst-case optimality in Horn ontologies with nominals, we have devised a variant of the calculus that decides in a single pass a number of subsumptions that share the same expression on the left-hand side.

We have also shown how our calculus can be applied to solve key DL reasoning problems, including ontology consistency checking, subsumption, and classification. Furthermore, and for the first time, we have shown how CB reasoning for expressive DLs can be used to solve instance retrieval and realisation. Furthermore, since our calculus is fully deterministic and highly modular, we have shown that context structures that have been created and saturated for answering a specific reasoning problem can be re-used to solve a different query; this saves us the task of computing many inferences a second time.

- **The Implementation.** We have implemented our calculus as an extension of the $\mathcal{SHIQ}$ reasoner Sequoia. We have introduced new data structures to represent the context clauses of our calculus, since they may include ground atoms and constants. We have shown how the root context used in our calculus can be implemented as a collection of smaller contexts that communicate with each other using special rules for exchanging information. This speeds up redundancy checks and makes the algorithm more suitable to parallelisation.

  We have also introduced a number of optimisation techniques in order to address pre-existing performance issues with Sequoia, as well as novel hurdles that emerge when we cover ontologies with nominals. For instance, we have shown that for queries consisting of single atom subsumptions, such as the ones that are used in classification, the completeness conditions of the calculus can be relaxed so that a more stringent literal ordering can be introduced. This ameliorates the performance problems of Sequoia in ontologies where the algorithm derives many clauses with a large number of disjuncts in the head. We have also introduced an optimisation which regulates the propagation of ground facts across contexts in order to strike a balance between saving memory and preventing the derivation of redundant clauses.

Finally, we have tested Sequoia on the task of ontology classification against other well-established DL reasoners. Our results show that Sequoia offers competitive performance, and that the strengths of our reasoner nicely complement those of (hyper-)tableau based reasoners. We have also shown that the optimisations that we have introduced produce a significant improvement on performance. The tests show that non-Horn ontologies with nominals or number restrictions can still be hard for Sequoia; however, the current implementation allows ample room for further optimisation.

## 10.2 Future Work

We see many open challenges to be addressed in future work, which we briefly discuss next:

- **Worst-Case Optimality.** We have shown that when there is a simultaneous interaction between nominals, inverse roles and number restrictions, our algorithm is worst-case triple exponential in time, when it should be possible to devise a double exponential time algorithm. We believe, however, that deriving such tighter upper bound would require a significant modification of our approach. For instance, a non-deterministic algorithm that uses the splitting rule for handling ground atoms, similar to the resolution-based calculus for $\mathcal{ALCHOIQ}$ used by Kazakov and Motik [76], might lead to a worst-case optimal algorithm for the full logic. Alternatively, we speculate that if contexts can be modified to resemble more closely the "star-types" used in the $\mathcal{ALCHOIQ}$ algorithm by Le Duc [82], this could produce a context structure of double-exponential size, instead of triple-exponential.

- **Datatypes.** Another outstanding task is extending our calculus with datatypes in order to cover all of OWL 2 DL; there is still little progress in the area of consequence-based reasoning in expressive DLs with concrete domains, but we believe that this step will not be excessively challenging. Work in this area should also include an implementation of datatype-handling capabilities in the reasoner Sequoia.

- **Optimisations for Sequoia.** Our evaluation has shown that there is still plenty of room for optimisation; in particular, Sequoia still struggles with several non-Horn ontologies with number restrictions. Nominals can also be a

source of problems for Sequoia due to the derivation of large numbers of ground clauses in many contexts. We will be working on further optimisation techniques to address the outstanding practical limitations. Another issue affecting our reasoner is the problem of repeating inferences across contexts. Switching to strategies that re-use contexts more often can alleviate this issue, but this can cause a proliferation of irrelevant clauses, as we have explained in Section 4.1. Thus, we would like to find alternative solutions that are not subject to this trade-off.

- **Parallel Reasoning.** We believe that it should be relatively easy to implement support for parallel reasoning, which seems particularly compatible with the underlying theoretical calculus. This should be made easier by the fact that some optimisations and implementation choices presented in this thesis have already been designed with an eye to supporting parallel reasoning.

- **Incremental Reasoning.** Sequoia currently creates a fresh context structure for each query. We have shown in this thesis, however, that context structures can be re-used in future queries to prevent the repetition of inferences. The implementation of Sequoia could be modified so that the reasoner creates and manages a single context structure, which is queried and extended as necessary when new queries are posed, or when the input ontology is modified. It would be very easy to implement support for incremental reasoning when the input ontology is extended with additional clauses. The next step, however, would pose a further challenge: we should study how a context structure can be minimally modified in cases where clauses are removed from the input ontology. Perhaps it is possible, for each set of clauses removed from the input, to find a tight bound on the subset of the context structure which may change. Then, context clauses outside this subset would be guaranteed to remain sound entailments for the new, reduced ontology. For this task, we might draw inspiration from other incremental reasoning approaches for materialisation-based methods for different fragments of first-order logic [95].

- **ABox Summarisation.** We also observe that in order to handle ontologies with a large number of nominals, it seems wasteful to create a context for each nominal; instead, we believe that Sequoia would benefit from ABox abstraction mechanisms such as the one described in [28]; in particular, we believe that it should be easy to use a standard context to represent a large set of individuals and succintly perform most or all relevant inferences on them.

- **Query Answering.** The application of consequence-based calculi to the problem of conjunctive query answering in DLs [48] remains also a fairly unexplored topic. A solid basis for this line of research is provided by "combined" approaches [79, 42, 33] to query answering that start with a materialisation phase similar to saturation in consequence-based reasoning, which is then followed by query-rewriting techniques.

- **New Unifying Frameworks.** The framework introduced in Section 3.2 does not completely cover the hybrid approaches from Section 3.3, which combine non-deterministic or tableau-like reasoning with consequence-based reasoning. Therefore, the question of whether there can be a more general unifying theoretical framework that is capable of capturing such techniques remains open.

All these observations lead us to firmly believe that consequence-based reasoning will continue being a dynamic area of research, one which holds promise for delivering the next generation of robust, efficient, and scalable reasoners for Description Logics.

# Appendix A

# Survey and Comparison of Consequence-Based Calculi

This chapter recapitulates key CB calculi from the literature and offers a detailed comparison between them. This analysis helps us identify the common features of CB calculi as shown in Section 4.1 and discusses aspects of interest for each calculus. In Appendix A.1 we discuss the CB calculus for $\mathcal{EL}$ by Baader et al.; we show the calculus is not complete for ontologies with nominals and we identify the cause of this problem. In Appendix A.2 we present the rules for the Horn-$\mathcal{SHIQ}$ CB calculus by Kazakov, and discuss how they relate to the $\mathcal{EL}$ calculus. In Appendices A.3 and A.4 we discuss the $\mathcal{ELHO}$ and Horn-$\mathcal{SROIQ}$ CB calculi, respectively, and relate them to the previous calculi. Next, in Appendix A.5 we introduce the rules of the $\mathcal{ALCH}$ calculus by Simančík et al., and compare it with prior CB calculi. Finally, in Appendix A.6, we introduce all the rules in the CB calculus for $\mathcal{ALCHI}$ by the same authors, which includes a unifying framework for CB reasoning in general, and then we show how to recover previous CB calculi in this framework.

## A.1 The Calculi for the $\mathcal{EL}$ Family

This section focuses on the CB calculus presented in [4] by Baader et al., which takes an $\mathcal{EL}{+}{+}(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ ontology $\mathcal{O}$ as input, and classifies it in polynomial time. In other words, for each pair $\langle A, B\rangle$ of concept names in $\mathsf{B} \cup \{\top, \bot\}$, with $\mathsf{B}$ the set of concept names, nominals, and concrete domains in $\mathcal{O}$, it determines in polynomial time whether $\mathcal{O} \models A \sqsubseteq B$. Unless we explicitly say so, we will maintain this use of $A$ and $B$, and also $C$ and $D$ for concept names throughout this entire text. A detailed overview of the calculus has already been given, in Section 3.1.1. Here, we give the

| | |
|---|---|
| CR1 | If $B \in \mathcal{S}(A), B \sqsubseteq C \in \mathcal{O}$, then add $C$ to $\mathcal{S}(A)$. |
| CR2 | If $B, C \in \mathcal{S}(A), B \sqcap C \sqsubseteq D \in \mathcal{O}$, and $D \notin \mathcal{S}(A)$ then add $D$ to $\mathcal{S}(A)$. |
| CR3 | If $B \in \mathcal{S}(A), B \sqsubseteq \exists R.C \in \mathcal{O}$, then add $(A, C)$ to $\mathcal{E}(R)$. |
| CR4 | If $(A, B) \in \mathcal{E}(R)$, $B \in \mathcal{S}(C)$, and $\exists R.C \sqsubseteq D \in \mathcal{O}$, then add $D$ to $\mathcal{S}(A)$. |
| CR5 | If $(A, B) \in \mathcal{E}(R)$ and $\bot \in \mathcal{S}(B)$, then add $\bot$ to $\mathcal{S}(A)$. |
| CR6 | If $\{o\} \in \mathcal{S}(A) \sqcap \mathcal{S}(B)$, and $A \rightsquigarrow_R B$, then add all elements in $\mathcal{S}(B)$ to $\mathcal{S}(A)$. |
| CR7 | If $\mathsf{con}_j(\mathcal{S}(A))$ is unsatisfiable in $\mathcal{D}_j$, then add $\bot$ to $\mathcal{S}(A)$. |
| CR8 | If $\mathsf{con}_j(\mathcal{S}(A))$ implies $p(f_1, \cdots, f_k) \in \mathsf{B}$ in $\mathcal{D}_j$, then add $p(f_1, \cdots, f_k)$ to $\mathcal{S}(A)$. |
| CR9 | If $p(f_1, \cdots, f_k)$ ,$p'(f'_1, \cdots, f'_{k'}) \in \mathcal{S}(A), p \in \mathcal{P}^{\mathcal{D}_j}$, $p' \in \mathcal{P}^{\mathcal{D}_l}, j \neq l, f_s = f'_t$ for some $s, t$, then add $\bot$ to $\mathcal{S}(A)$. |
| CR10 | If $(A, B) \in \mathcal{E}(R)$ and $R \sqsubseteq S \in \mathcal{O}$, then add $(A, B)$ to $\mathcal{E}(S)$. |
| CR11 | If $(A, B) \in \mathcal{E}(R_1)$, $(B, C) \in \mathcal{E}(R_2)$, and $R_1 \circ R_2 \sqsubseteq R_3 \in \mathcal{O}$, then add $(A, C)$ to $\mathcal{E}(R_3)$. |

Figure A.1: Rules of the $\mathcal{EL}{+}{+}(\mathcal{D}_1, \ldots \mathcal{D}_n)$ proceedure, Baader et al. (2005). We have $\mathsf{con}_j(\Gamma) = \bigwedge_{p(f_1,\cdots,f_k), p \in \mathcal{P}^{\mathcal{D}_j}} p(f_1, \cdots, f_k)$; $\mathcal{D}_j$ is a concrete domain with domain $\Delta^{\mathcal{D}_j}$ and set of predicate names $\mathcal{P}^{\mathcal{D}_j}$, and $f_1, \ldots, f_k$ are *features* which map elements of the general interpretation domain to elements of the concrete domain$\Delta_j$. Relation $\rightsquigarrow_R$ is defined as: $C \rightsquigarrow_R D$ iff there are $C_1, C_2, \ldots, C_k \in \mathsf{B}$ such that: $C_1 = C$ or $C_1 = \{o\}$ for an individual name $o, C_k = D$, and $\langle C_i, C_{i+1} \rangle \in \mathcal{E}(R_i)$, for some $R_i \in \mathcal{E}(R_i)$ for $1 \leq i < k$.

full set of inference rules of the calculus (see Figure A.1) and discuss why this calculus is not complete for ontologies with nominals.

In general, the main difficulty of consequence-based reasoning algorithms is to prove completeness. To do this, one must show that if an arbitrary subsumption $A \sqsubseteq B$ is entailed by the ontology, then *it must be derived* during the execution of the algorithm. In this calculus, we must have $B \in \mathcal{S}(A)$ after the saturation of the set $\mathcal{S}$.

The proof follows works by constructing a *canonical model* that disproves any subsumption that has not been derived by the calculus. An interesting feature of this proof is that the method for building the canonical model does not depend on the inclusion to be disproved, because the canonical model disproves *all* non-entailed inclusions betweens pairs of concept names. But this is an exclusive property of this description logic; it is easy to prove, for more expressive logics like $\mathcal{ALCO}$, that there is no "canonical" model that can disprove all non-entailed inclusions.

The canonical model is built from the saturated structure: first, for each $A \in \mathsf{B}$,

the canonical model $\mathcal{I}$ introduces an element $x_A$ in the interpretation domain $\Delta^{\mathcal{I}}$. Then, we add $x_A$ to $B^{\mathcal{I}}$ if and only if $B \in \mathcal{S}(A)$; furthermore, for each $A, B \in \mathsf{B}$ and $R \in \mathbf{S}$, the model is such that $\langle x_A, x_B \rangle \in R^{\mathcal{I}}$ if and only if $\langle A, B \rangle \in \mathcal{E}(R)$. The idea behind this is that, since $\mathcal{S}(A)$ contains every concept name subsuming $A$, every element contained in $A^{\mathcal{I}}$ is of the form $x_C$ for some $C$ such that $A \in \mathcal{S}(C)$, that is, some $C$ such that $C \sqsubseteq A$ is entailed. If the algorithm is complete, $C \sqsubseteq B$ will have been derived, so we will add $x_C$ also to $B^{\mathcal{I}}$, and $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ will be satisfied; the burden of the proof consists in showing that this actually occurs, thanks to the action of the inference rules. For instance, for axioms of the form $A \sqsubseteq B$, we have that if $x_C$ is any element of $A^{\mathcal{I}}$, then $A \in \mathcal{S}(C)$. But if $A \in \mathcal{S}(C)$, rule $\mathsf{CR1}$ has been applied with axiom $A \sqsubseteq B$, so $B \in \mathcal{S}(C)$, and hence $x_C \in B^{\mathcal{I}}$ by definition of the model. A similar idea is used for each of the possible types of axioms in the ontology: for axioms of form $A_1 \sqcap A_2 \sqsubseteq B$ we use $\mathsf{CR2}$, for axioms of the form $C \sqsubseteq \exists R.D$, we use $\mathsf{CR3}$ and for those of the form $\exists R.C \sqsubseteq D$ we use $\mathsf{CR4}$. The remaining rules are used to deal with role inclusion axioms, or to prove intermediate results having to do with concrete predicates and nominals.

In order to take care of nominals, the calculus defines the relation $\sim$ on $\mathsf{B}$ as follows:

$$A \sim B \text{ iff } A = B \text{ or } \{a\} \in \mathcal{S}(A) \cap \mathcal{S}(B) \text{ for some individual name } a$$

Then, the proof claims that for any $A, B \in \mathsf{B}$ such that $A \sim B$, we have $\mathcal{S}(A) = \mathcal{S}(B)$ due to rule $\mathsf{CR6}$. However, this is not true: consider an ontology consisting of a single GCI $A \sqsubseteq \{a\}$. The execution of the algorithm on this ontology yields $\mathcal{S}(A) = \{A, \{a\}\}$ $\mathcal{S}(\{a\}) = \{\{a\}\}$, so by definition of $\sim$ we have that $A \sim \{a\}$, while $\mathcal{S}(A) \neq \mathcal{S}(\{a\})$. Indeed, rule $\mathsf{CR6}$ does not help here, for in order to add $A$ to $\mathcal{S}(\{a\})$, we need the relation $\{a\} \rightsquigarrow_R A$ to hold, and this only happens, according to the algorithm definition, if $\langle \{a\}, A \rangle \in \mathcal{E}(R)$ for some $R \in \mathbf{S}$, which is impossible, since $\mathbf{S}$ is empty. This leads to incompleteness, as we can see in the following example:

$$\begin{array}{ll} A \sqsubseteq \exists R.B_1 & A \sqsubseteq \exists S.B_2 \\ B_1 \sqsubseteq \{o\} & B_2 \sqsubseteq \{o\} \\ B_1 \sqsubseteq C & \exists .C \sqsubseteq \bot \end{array}$$

Figure A.2: Counterexample ontology $\mathcal{O}_1$

The concept $A$ is unsatisfiable in this $\mathcal{O}_1$ from Figure A.2, that is, its interpretation must be empty in every model. To see this, suppose $\mathcal{I}$ is a model with an element

$$\mathsf{IR1}\ \frac{}{A \sqsubseteq A} \qquad \mathsf{IR2}\ \frac{}{A \sqsubseteq \top} \qquad \mathsf{C1}\ \frac{A \sqsubseteq B \qquad B \sqsubseteq C \in \mathcal{O}}{A \sqsubseteq C}$$

$$\mathsf{C2}\ \frac{A \sqsubseteq B \qquad A \sqsubseteq C \qquad B \sqcap C \sqsubseteq D \in \mathcal{O}}{A \sqsubseteq D}$$

$$\mathsf{C3}\ \frac{A \sqsubseteq B \qquad B \sqsubseteq \exists R.C \in \mathcal{O}}{A \sqsubseteq \exists R.C} \qquad \mathsf{C4}\ \frac{A \sqsubseteq \exists R.B \qquad R \sqsubseteq S \in \mathcal{O}}{A \sqsubseteq \exists S.B}$$

$$\mathsf{C5}\ \frac{A \sqsubseteq \exists R.B \qquad B \sqsubseteq C \qquad \exists R.C \sqsubseteq D \in \mathcal{O}}{A \sqsubseteq D}$$

Figure A.4: Inference rules for $\mathcal{EL}\mathcal{H}$ – Version 2 [Kazakov et al. 2009]

$x \in A^{\mathcal{I}}$. Without loss of generality, let us assume $o^{\mathcal{I}} = o$. By axiom $A \sqsubseteq \exists R.B_1$ , there is an element $y \in B_1^{\mathcal{I}}$ with $\langle x, y\rangle \in R^{\mathcal{I}}$, and by axiom $B_1 \sqsubseteq \{o\}$, we have $y = o$. Analogously, because of axioms $A \sqsubseteq \exists S.B_2$ and $B_2 \sqsubseteq \{o\}$, we have $\langle x, o\rangle \in S^{\mathcal{I}}$. However, by axiom $B_1 \sqsubseteq C$, we have $o \in C^{\mathcal{I}}$, but this violates axiom $\exists S.C \sqsubseteq \bot$ for $x$. Thus, we have reached a contradiction. However, the algorithm fails to derive $A \sqsubseteq \bot$. Indeed, the full execution of the algorithm builds the following structure, where $\bot \notin \mathcal{S}(A)$:

| | |
|---|---|
| $\mathcal{S}(A) = \{A\}$ | $\mathcal{S}(C) = \{C\}$ |
| $\mathcal{S}(B_1) = \{B_1, C, \{o\}\}$ | $\mathcal{S}(B_2) = \{B_2, \{o\}\}$ |
| $\mathcal{S}(\{o\}) = \{\{o\}\}$ | |
| $\mathcal{E}(R) = \{\langle A, B_1\rangle, \langle A, \{o\}\rangle, \langle A, C\rangle\}$ | $\mathcal{E}(S) = \{\langle A, B_1\rangle, \langle A, \{o\}\rangle\}$ |

Figure A.3: Saturated structure

Relation $\rightsquigarrow_R$ holds for the following pairs: $\{\langle A, \{o\}\rangle, \langle A, B_1\rangle, \langle A, B_2\rangle, \langle A, C\rangle\}$ It is easy to check that this structure is saturated by the rules of Table A.1.

The algorithm is incomplete because the "reachability" relation $\rightsquigarrow_R$, used in rule CR6, relates only elements that are in the same "branch" of the model (seen as a tree). However, two elements in different branches can both collapse into the same nominal, and therefore the algorithm is missing a rule that can deal with these situations.

## A.2 The Calculus for Horn-$\mathcal{SHIQ}$

The paper introducing the Horn-$\mathcal{SHIQ}$ CB calculus from Section 3.1.2 first presents an alternative form of the $\mathcal{EL}^{++}$ calculus from the previous section, focused on the

$$\mathsf{I1}\ \frac{}{M \sqcap A \sqsubseteq A} \qquad \mathsf{R1}\ \frac{M \sqsubseteq A_i \qquad \bigsqcap A_i \sqsubseteq C \in \mathcal{O}}{M \sqsubseteq C}$$

$$\mathsf{I2}\ \frac{}{M \sqsubseteq \top} \qquad \mathsf{R2}\ \frac{M \sqsubseteq \exists R.N \qquad N \sqsubseteq \bot}{M \sqsubseteq \bot}$$

$$\mathsf{R3}\ \frac{M \sqsubseteq \exists R_1.N \qquad M \sqsubseteq \forall R_2.A \qquad R_1 \sqsubseteq_{\mathcal{O}} R_2}{M \sqsubseteq \exists R_1.(N \sqcap A)}$$

$$\mathsf{R4}\ \frac{M \sqsubseteq \exists R_1.N \qquad N \sqsubseteq \forall R_2.A \qquad R_1 \sqsubseteq_{\mathcal{O}} R_2}{M \sqsubseteq A}$$

$$\mathsf{R5}\ \frac{\begin{array}{llll} M \sqsubseteq \exists R_1.N_1 & N_1 \sqsubseteq B & R_1 \sqsubseteq_{\mathcal{O}} S & \\ M \sqsubseteq \exists R_2.N_2 & N_2 \sqsubseteq B & R_2 \sqsubseteq_{\mathcal{O}} S & N_1 \sqsubseteq \leq 1S.B \end{array}}{M \sqsubseteq \exists R_1(N_1 \sqcap N_2)}$$

$$\mathsf{R6}\ \frac{\begin{array}{lll} M \sqsubseteq \exists R_1.N_1 & N_1 \sqsubseteq \exists R_2(N_2 \sqcap A) & R_1 \sqsubseteq_{\mathcal{O}} S^- \\ M \sqsubseteq B & N_2 \sqcap A \sqcap B & R_2 \sqsubseteq_{\mathcal{O}} S \qquad N_1 \sqsubseteq \leq 1S.B \end{array}}{\begin{array}{ll} M \sqsubseteq A & M \sqsubseteq \exists R_2^-.N_1 \end{array}}$$

Figure A.5: Inference rules for Horn-$\mathcal{SHIQ}$

$\mathcal{ELH}$ fragment. We reproduce this calculus in Figure A.4, so that later we can compare it with the method for Horn-$\mathcal{SHIQ}$. The equivalence of these rules with those in Figure A.1 can be readily checked: rule IR1 and IR2 take care of the initialisation described in Appendix A.1. Rule C1 corresponds to CR1, and in the same way we have a correspondence between C2 and CR2, between C3 and CR3, and between C4 and CR10, and C5 and CR4.

We turn now towards the Horn-$\mathcal{SHIQ}$ procedure. We have already given an overview of this calculus in the main text (Section 3.1.2), so here we present the full set of rules of the calculus, given in Figure A.5. It is easy to see that most of the rules of the Horn-$\mathcal{SHIQ}$ approach are analogous to the rules of $\mathcal{ELH}$. Indeed, I1 is like R1, and I2 similarly corresponds to IR2, but allowing for the new form of inferences, with long conjunctions on the left hand side. R1 combines C1 and C2, and also extends to axioms of the ontology with longer conjunctions in the left-hand side, which are now allowed in the normalisation of the ontology. Moreover, because the variable $C$ can also be an existential restriction, it also covers C4. Observe that this inference rule, if we would consider the clausal form of consequences, would essentially correspond to hyperresolution [111].

Next we have R2, which has no direct analogous in Figure A.4, but corresponds to CR5 for $\mathcal{EL}$++. Rule R3 is new; it deals with universal restrictions, drawing the kind of inferences discussed above which use universal restrictions as premises. R4 is a combination of CR5 and CR4: observe that the second axiom can be written $\exists R_2'.N \sqsubseteq A, R_1 \sqsubseteq R_2'$, where note that $R_2'$ replaces $R_2^-$. Rules R5 and R6 have no analogous, because they deal with functional restrictions. In particular, R5 deals with cases where two role successors of some concept collapse into each other, while R6 deals with the combination of functional restrictions with inverse roles: when a concept $N_2$ is the $R^-$-role successor of an $R$-successor of a concept $M$, and $R$ is restricted functionally, then it must contain all instances of $M$.

The calculus maintains the same good properties that characterised the $\mathcal{ELH}$ procedure: it is worst-case optimal, it allows for one-pass classification, and interestingly, it becomes analogous to the $\mathcal{ELH}$ procedure on a $\mathcal{ELH}$ ontology, so it also worst-case optimal for $\mathcal{ELH}$ ontologies. Thus, the method displays pay-as-you-go behaviour.

## A.3 The Calculus for $\mathcal{ELHO}$

Similarly to the Horn-$\mathcal{SHIQ}$ discussed above, the paper introducing the first complete CB calculus for $\mathcal{ELHO}$ [73] contributes by presenting yet another version of the

$$\mathsf{R}_{\sqsubseteq}\ \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \in E \qquad \mathsf{R}^{-}_{\sqcap}\ \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}$$

$$\mathsf{R}^{-}_{\exists}\ \frac{C \sqsubseteq \exists R.D}{D \sqsubseteq D} \qquad \mathsf{R}^{+}_{\top}\ \frac{C \sqsubseteq C}{C \sqsubseteq \top} : \top \text{ occurs in } \mathcal{O}$$

$$\mathsf{R}^{+}_{\sqcap}\ \frac{C \sqsubseteq D_1 \quad C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O}$$

$$\mathsf{R}^{+}_{\exists}\ \frac{C \sqsubseteq \exists R.D \quad D \sqsubseteq E}{C \sqsubseteq \exists R.E} : \exists R.E \text{ occurs in } \mathcal{O}$$

Figure A.6: Inference rules for $\mathcal{ELH}$ – Version 3, Simančík et al. (2011)

calculus for $\mathcal{ELH}$ which is further simplified. Indeed, the new version does not require normalisation, although authors have noted that some limited form of normalisation can help improve the performance in practice. We present these rules in here, and we briefly compare them with those in Figure A.4.

In the figure, $R \sqsubseteq_O S$ represents that there is a chain of role inclusions from $R$ to $S$ in the ontology. Observe that the main difference with respect to the procedures already discussed is that derived consequences allow conjunctions and existential restrictions (where inner concepts can be general) in *both* sides. This representation is motivated by performance, but we will not discuss this further since this issue is beyond the scope of this thesis. Rule $R_{\sqsubseteq}$ corresponds to C1, C2 and C3 and part of C5 – the latter, only in the special case where $B = C$. No rule that can take an arbitrary number of premises, like C2, is necessary in this calculus, because we can directly combine an ontology axiom $A_1 \sqcap A_2 \sqcap \cdots \sqcap A_n \sqsubseteq \ldots$ with a derived consequence that has $A_1 \sqcap \cdots \sqcap A_n$ as its right-hand side. Such consequences are generated with rule $R^{+}_{\sqcap}$. An optimisation for the calculus introduced in the paper consists in only applying this rule if the resulting conjunction appears in the ontology (for otherwise, no new consequences will follow); this anticipates the goal-oriented behaviour of contexts in later calculi.

With regard to the other rules, we have that rule $R^{-}_{\sqcap}$ allows us to break large conjunctions, which is helpful whenever we only need to resolve a part of such conjunctions. Rule $R^{+}_{\top}$ corresponds to IR2. Rule $R^{-}_{\exists}$ is actually an optimised form of rule IR1: instead of producing every inclusion of the form $C \sqsubseteq C$, we do it only for concepts that are "relevant" i.e. concepts that appear in some existential restriction, and we also add $C \sqsubseteq C$ as part of the initialisation. In particular, for classification, we add $A \sqsubseteq A$ for each concept name, as in IR1. Finally, rule $R^{+}_{\exists}$, combined

with $R_{\sqsubseteq}$, generates the rest of C5. This shows that the rationale behind rule C5 is sound for concepts of any form, not just concepts that follow the normal form used in Figure A.4, which helps motivate this new form of the calculus. In order to further optimise the method, this rule is only allowed when the resulting concept appears in the ontology.

The calculus for $\mathcal{ELHO}$ is built on top of that described in Figure A.6. Observe that because concepts of the form $C$ and $D$ can have any form, the calculus can be naturally extended by allowing nominals in these concepts. However, as we have argued in Section 4.2, the calculus must keep track of additional, non-local restrictions which cannot be directly and explicitly expressed in DLs. The rules of the calculus are a modification of those in Figure A.6: $\mathsf{R}_{\sqsubseteq}$, $\mathsf{R}^{-}_{\sqcap}$, $\mathsf{R}^{+}_{\sqcap}$ are simply altered so that the conclusion and all previously inferred consequences are now preceded by '$G$:'. The remaining rules are:

$$\mathsf{R}^{+}_{\rightsquigarrow}\ \frac{G \rightsquigarrow C \quad G : C \sqsubseteq \exists R.D}{G \rightsquigarrow D} \qquad \mathsf{R}^{-}_{\rightsquigarrow}\ \frac{G \rightsquigarrow D}{G : D \sqsubseteq D}$$

$$\mathsf{R}^{-}_{\exists}\ \frac{C \sqsubseteq \exists R.D}{D \sqsubseteq D} \qquad \mathsf{R}^{+}_{\top}\ \frac{G \rightsquigarrow C}{g : C \sqsubseteq \top} : \top \text{ occurs in } \mathcal{O}$$

$$\mathsf{R}_{\{\}}\ \frac{G : C \sqsubseteq \{o\} \quad G : D \sqsubseteq \{o\} \quad G \rightsquigarrow C \quad G \rightsquigarrow D}{G : C \sqsubseteq D} : \top \text{ occurs in } \mathcal{O}$$

Figure A.7: Inference rules for $\mathcal{ELHO}$ [Kazakov et al. 2011]

Observe that $R^{+}_{\rightsquigarrow}$ is used for propagating the reachability relation, and $R^{-}_{\rightsquigarrow}$ is a new version of rule $R^{-}_{\exists}$. Indeed, the combined action of $R^{-}_{\rightsquigarrow}$ and $R^{+}_{\rightsquigarrow}$ reproduces exactly $R^{-}_{\exists}$. Finally, the key addition is the rule $\mathsf{R}_{\{\}}$, which deals with nominal interactions. Reachability relations are crucial for applying this rule; furthermore, the use of conditional subsumptions as premises prevents the problem discussed in Appendix A.1.

The calculus maintains some of the good properties that characterised the $\mathcal{ELH}$ procedure: it is worst-case optimal, it is capable of a weak form of one-pass classification (see Section 7.3 for more detail), and,it becomes analogous to the $\mathcal{ELH}$ procedure on an $\mathcal{ELH}$ ontology.

## A.4 The Calculus for Horn-$\mathcal{SROIQ}$

The method described in this section is used in Horn-$\mathcal{SROIQ}$ ontologies [105]. Compared with the Horn-$\mathcal{SHIQ}$ method discussed inAppendix A.2, this calculus addi-

tionally supports nominals ($\mathcal{O}$), role boxes ($\mathcal{R}$), self-reflexivity, and disjointness axioms. However, as it is the case with transitivity, role boxes can be encoded away [55]. Thus, the main contribution of the paper consists in extending the consequence-based method to $\mathcal{ALCHOIQ}^+$ ontologies. Another novelty is that this method uses a Datalog-inspired [41] formalism to represent the ontology, the inference rules, and the consequences. In Figure A.8 we present the set of rules. For the sake of simplicity, we have excluded self-reflexivity and disjointness axioms, in order to focus only on Horn-$\mathcal{SHOIQ}$. We have that $T_1, T_2$ are sets of concept names, $E$ are sets of role names, and, as usual, $A, B$, $R$, and $o$ are concept names, a role, and an individual name, respectively.

Even though the notation is substantially different from the previous cases, it is still possible to see some similarities. This procedure resembles a combination of the $\mathcal{ELHO}$ and Horn-$\mathcal{SHIQ}$ procedures: it uses a normalisation which is closer to the former, but it includes clauses to deal with universal quantification and functional restrictions which are analogous to those used in the latter. Rule M1 and rule M2 do not correspond to any particular rule in previous calculi, since they are simply axiom encodings; however, observe that the choice of *ontology* normalisation corresponds to that used in the paper discussed Appendices A.1 and A.2, rather than that used in Figure A.6, even though the language for conclusions is most similar to the latter. Rule M3 corresponds to rule $\mathsf{R}^+_\top$, even though it does not have the corresponding optimisation. Rule M4 does not have a direct analgous in previous calculi, even though strictly speaking, those calculi should have similar rules. However, inclusions of the form $\bot \sqsubseteq A$ are trivially entailed by any ontology, so they are simply omitted from those calculi and the corresponding completeness theorems. Rule M5 can be seen as analogous to the rule $\mathsf{R}^-_\sqcap$, with $D_1 \sqcap D_2$ playing the role of $T_1$ and $D_1$ playing the role of $T_2$; this rule would then derive $D_1 \sqcap D_2 \sqsubseteq D_1$, and this, in combination with $C \sqsubseteq D_1 \sqcap D_2$, would lead to recovering $\mathsf{R}^-_\sqcap$. Rule M6 corresponds to the special case in rule $R_\sqsubseteq$ where every participating concept is a conjunction of concept names. Rule M7 is most similar to $\mathsf{R}^+_\sqcap$, even though it is more general: it corresponds to an application of rule $\mathsf{R}^-_\sqcap$ to $T_1 \sqcap A$ to generate $T_1$, then $\mathsf{R}_\sqsubseteq$ to generate $T_1 \sqcap A \sqsubseteq T_2$, and only then $\mathsf{R}^-_\sqcap$ with $A \sqsubseteq A$ to reach $T_1 \sqcap A \sqsubseteq T_2 \sqsubseteq A$. Rule M8 does not have a direct analogous in Figure A.6, since relations between conjunctions are stored simply as derivations $M \sqsubseteq \exists R.N$, and we can see it just as an axiom encoding. Observe, however, that there is a stronger analogy betwen this rule and CR3, since the calculus in Figure A.1 also represents relations using a special set; note, though, that the

| | | | |
|---|---|---|---|
| M1 | | $\rightarrow$ | $\mathsf{imp}(\{A\},\{B\})$<br>for each axiom $A \sqsubseteq B$ |
| M2 | | $\rightarrow$ | $\mathsf{imp}(\{A,B\},\{C\})$<br>for each axiom $A \sqcap B \sqsubseteq C$ |
| M3 | | $\rightarrow$ | $\mathsf{imp}(\{\},\{\top\})$ |
| M4 | | $\rightarrow$ | $\mathsf{imp}(\{\bot\},\{\mathcal{BC}_\mathcal{O}\})$ |
| M5 | | $\rightarrow$ | $\mathsf{imp}(T_2,T_1)$<br>for each $T_1, T_2$ s.t. $T_1 \subseteq T_2$ |
| M6 | $\mathsf{imp}(T_1,T_2)$ $\mathsf{imp}(T_2,T_3)$ | $\rightarrow$ | $\mathsf{imp}(T_1,T_3)$ |
| M7 | $\mathsf{imp}(T_1,T_2)$ | $\rightarrow$ | $\mathsf{imp}(T_1 \cup \{A\}, T_2 \cup \{A\})$<br>for each $A \in \mathcal{BC}_\mathcal{O}$ |
| M8 | | $\rightarrow$ | $\mathsf{enf}(\{A\},\{R\},\{B\})$<br>for each axiom $A \sqsubseteq \exists R.B$ |
| M9 | $\mathsf{enf}(T_1,E,T_2)$,<br>$\mathsf{imp}(T_1',T_1)\mathsf{imp}(T_2,T_2')$ | $\rightarrow$ | $\mathsf{enf}(T_1',E,T_2')$ |
| M10 | $\mathsf{enf}(T_1,E,T_2), \in (R,E)$ | $\rightarrow$ | $\mathsf{enf}(T_1, E \cup \{S\}, T_2)$<br>for each axiom $R \sqsubseteq S$ |
| M11 | $\mathsf{enf}(T_1,E,T_2), \in (\bar{R},E)$ | $\rightarrow$ | $\mathsf{enf}(T_1, E \cup \{\bar{S}\}, T_2)$<br>for each axiom $R \sqsubseteq S$ |
| M12 | $\mathsf{enf}(T_1,E,T_2)$,<br>$\in (A,T_1), \in (R,E)$ | $\rightarrow$ | $\mathsf{enf}(T_1, E \cup T_2 \cup \{B\})$<br>for each axiom $A \sqsubseteq \forall R.B$ |
| M13 | $\mathsf{enf}(T_1,E,T_2)$,<br>$\in (\bar{R},E), \in (A,T_2)$ | $\rightarrow$ | $\mathsf{imp}(T_1,\{B\})$<br>for each axiom $A \sqsubseteq \forall R.B$ |
| M14 | | $\rightarrow$ | $\mathsf{inv}(\{R\},\{\bar{R}\})$<br>for each $R \in \mathsf{R}_\mathcal{O}$ |
| M15 | $\mathsf{inv}(E_1,E_2)$ | $\rightarrow$ | $\mathsf{inv}(E_1 \cup \{R\}, E_2 \cup \{\bar{R}\})$<br>for each $R \in \mathsf{R}_\mathcal{O}$ |
| M16 | $\mathsf{enf}(T,E_1,T_1), \mathsf{enf}(T,E_2,T_2)$,<br>$\in (A,T), \in (R,E_1)$,<br>$\in (R,E_2), \in (B,T_1), \in (B,T_2)$ | $\rightarrow$ | $\mathsf{enf}(T_1, E_1 \cup E_2, T_1 \cup T_2)$,<br><br>for each axiom $A \sqsubseteq \leq 1R.B$ |
| M17 | $\mathsf{enf}(T_1,E_1,T_2), \mathsf{enf}(T_2,E_2,T_3)$,<br>$\in (A,T_2), \in (R,E_2), \in (\bar{R},E_1)$,<br>$\in (B,T_1), \in (B,T_3), \mathsf{inf}(E_2,\bar{E}_2)$ | $\rightarrow$ | $\mathsf{imp}(T_1,T_3)$,<br>$\mathsf{enf}(T_1, E_1 \cup E_2', T_2)$<br>for each axiom $A \sqsubseteq \leq 1R.B$ |
| M18 | | $\rightarrow$ | $\mathsf{rel}(\{\{a\}\})$<br>for each $\{o\} \in \mathsf{B}$ |
| M19 | $\mathsf{rel}(T_1), \mathsf{enf}(T_1,E,T_2)$ | $\rightarrow$ | $\mathsf{rel}(T_2)$ |
| M20 | $\mathsf{rel}(T_1), \mathsf{imp}(T_1,T_2)$ | $\rightarrow$ | $\mathsf{rel}(T_2)$ |
| M21 | $\mathsf{rel}(\{\bot\})$ | $\rightarrow$ | |
| M22 | $\mathsf{rel}(T_1), \mathsf{rel}(T_2)$,<br>$\in (\{o\},T_1), \in (\{o\},T_2)$ | $\rightarrow$ | $\mathsf{same}(T_1,T_2)$ |
| M23 | $\mathsf{same}(T_1,T_2)$ | $\rightarrow$ | $\mathsf{real}(T_1 \cup T_2)$ |
| M24 | $\mathsf{rel}(T_1), \mathsf{enf}(T_1,E,T_3)$,<br>$\mathsf{rel}(T_2), \mathsf{enf}(T_2,E_2,T_4)$,<br>$\in (B,T_1), \in (B,T_2)$,<br>$\in (\bar{R},E_1), \in (\bar{R},E_2)$,<br>$\in (A,T_3), \mathsf{same}(T_3,T_4)$ | $\rightarrow$ | $\mathsf{same}(T_1,T_2)$<br><br><br><br><br>for each $A \sqsubseteq \leq 1R.B$ |

Figure A.8: Inference rules for Horn-$\mathcal{SROIQ}$, Ortiz et al. (2010)

structures for representing relations in Figure A.1 are associated to single role names, while this calculus uses "types" of relations, which can include many role names.

Rule M9 is a more general version of rule $\mathsf{R}^+_\exists$ which includes first an application of $\mathsf{R}_\sqsubseteq$ to generate the premise $C \sqsubseteq \exists R.D$ from $C' \sqsubseteq \exists R.D$ and $C \sqsubseteq C'$. Rule M10 has no analogous in Figure A.6, from where we have excluded ways of dealing with axiom role inclusions, but it is equivalent with CR10, which uses a similar representation for role relations between concepts. Rule M11 generalises the previous case to inverse roles. Observe that these two rules are "spread" in Horn-$\mathcal{SHIQ}$: instead of explicitly computing all consequences replacing roles by those that subsume them, whenever such consequence would participate in an inference of a different kind, rules R3-R6 already compute that consequence as a part of their execution, and if it would participate in an inference of the same kind, this is taken into account because the procedure calculates the closure of role inclusions. Rule M12 has no equivalent in Figure A.6 since universal restrictions are not supported there, so we compare it instead with Figure A.5 and see that it corresponds directly to R3. Rule M13 is similarly analogous to R4. Rules M14 and M15 are specific to this calculus and deal with the encoding of inverse roles; inverse roles are directly used in the rules of Figure A.5. Rules M16 and M17 deal exactly with the same situations than R5 and R6.

The remaining rules deal with nominals, and use a relation rel which is similar to the reachability relations $\rightsquigarrow_R$ in Figure A.1 and $\rightsquigarrow$ in Figure A.7. In Appendix A.1 we saw that using the former relation to deal with nominals leads to incompleteness, so we draw the connections with the latter instead. First, M18 draws this relation for every nominal: every nominal is reachable. Rule M19 is analogous to $\mathsf{R}^+_\rightsquigarrow$, and rule M20 is unnecessary since it can be recovered with rule M9 and M19. Rule M21 is included in $\mathsf{R}_{\{\}}$ in case $D = \bot$, since it can be used to derive $\{o\} \sqsubseteq \bot$, which implies that the ontology is unsatisfiable. Rule M22 and M23 correspond to $\mathsf{R}_{\{\}}$, but more general in order to include "equalities" that may also be generated in M24. The rule M24 has no analogous in any of the other calculi, since it deals with a combination of the expressive features of nominals, inverse roles, and functional restrictions, which does not appear in any of the other languages discussed. This has been discussed in Section 7.3, and we note that the rule M23 ensures that predicate same has the intended meaning.

Once more, it is proved that the calculus is worst-case optimal. However, it is not clear that it is pay-as-you-go, due to the massive initialisation required, which may lead to an exponential number of rules. Moreover, this calculus does not allow

$$\mathsf{R}^{+}_{\mathsf{A}}\ \frac{}{H \sqsubseteq A} : A \in H \qquad \mathsf{R}^{-}_{\mathsf{A}}\ \frac{H \sqsubseteq N \sqcup A}{H \sqsubseteq N} : \neg A \in H$$

$$\mathsf{R}^{n}_{\sqcap}\ \frac{H \sqsubseteq N_i \sqcup A_i}{H \sqsubseteq \bigsqcup_{i=1}^{n} N_i \sqcup M} : \bigsqcap_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$$

$$\mathsf{R}^{+}_{\exists}\ \frac{H \sqsubseteq N \sqcup A}{H \sqsubseteq \exists R.B} : A \sqsubseteq \exists R.B \in \mathcal{O}$$

$$\mathsf{R}^{-}_{\exists}\ \frac{H \sqsubseteq M \sqcup \exists R.K \quad K \sqsubseteq N \sqcup A}{H \sqsubseteq M \sqcup B \sqcup \exists R.(K \sqcap \neg A)} : \exists S.A \sqsubseteq B \in \mathcal{O},\ \ R \sqsubseteq_{\mathcal{O}} S$$

$$\mathsf{R}^{\perp}_{\exists}\ \frac{H \sqsubseteq M \sqcup \exists R.K \quad K \sqsubseteq \perp}{H \sqsubseteq M}$$

$$\mathsf{R}_{\forall}\ \frac{H \sqsubseteq M \sqcup \exists R.K \quad H \sqsubseteq N \sqcup A}{H \sqsubseteq M \sqcup N \sqcup \exists R.(K \sqcap B)} : A \sqsubseteq \forall S.B \in \mathcal{O},\ \ R \sqsubseteq_{\mathcal{O}} S$$

Figure A.9: Inference rules for $\mathcal{ALCH}$

for one-pass classification, as it is devised only as a satisfiability procedure. Finally, this calculus has not been implemented, so it is unclear what are the strengths and weaknesses of this method in practice.

## A.5 The Calculus for $\mathcal{ALCH}$

The calculus for $\mathcal{ALCH}$ by Simančík et al. [124] can be best explained as a direct analogy to that discussed in Appendix A.2. In fact, it is a natural extension of the restriction of Horn-$\mathcal{SHIQ}$ calculus to Horn-$\mathcal{ALCH}$, which corresponds exactly to rules IR1,IR2,R1-R4 in Figure Figure A.5. Therefore, this calculus compares with that in Appendix A.1 in the way discussed in Appendix A.2. We have already covered the basic challenges and standard form of consequences in this calculus, in Section 3.1.4. Thus, in this section, we simply focus on the difference between this $\mathcal{ALCH}$ calculus, and the calculus in Appendix A.2, restricted to Horn-$\mathcal{ALCH}$.

The rules of the calculus are given in Figure A.9. Note that in the inference rules, symbol $A$ *can be a negated atomic concept*, in addition to an atomic concept. The comparison with the calculus in Figure A.5 is very direct: rule $\mathsf{R}^{+}_{\mathsf{A}}$ is identical to I1; rule $\mathsf{R}^{n}_{\sqcap}$ is the generalisation of R1 (with a conjunction in the right-hand side of the ontology axiom) to inferences with disjunction on the right-hand side, and the same relation holds between $\mathsf{R}^{+}_{\exists}$ and R1 (with an existential restriction in the right-hand

| | | |
|---|---|---|
| Hyper | If | $\prod_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$, |
| | | $K_i \sqsubseteq M_i \sqcup A_i \in \mathcal{S}(v)$ for $1 \leq i \leq n$ |
| | then | add $\prod_{i=1}^{n} K_i \sqsubseteq M \sqcup \bigsqcup_{j=1}^{n} M_i$ |
| Pred | If | $\langle v, u, \exists R.A \rangle \in \mathcal{E}$, |
| | | $A \sqcap \prod_{i=1}^{n} B_i \sqsubseteq \bigsqcup_{j=1}^{m} \forall(R^-).C_j \in \mathcal{S}(u)$, |
| | | $K_0 \sqsubseteq M_0 \sqcup \exists R.A \in \mathcal{S}(v)$, |
| | | and $K_i \sqsubseteq M_i \sqcup \forall R.B_i \in \mathcal{S}(v)$ for $1 \leq i \leq n$. |
| | then | add $\prod_{i=0}^{n} K_i \sqsubseteq \bigsqcup_{i=0}^{n} M_i \sqsubseteq \bigsqcup_{j=1}^{m} C_j$ to $\mathcal{S}(v)$. |
| Succ | If | $K \sqsubseteq M \sqcup \exists R.A \in \mathcal{S}(v)$ |
| | | for $\mathcal{B}_p = \{B \mid K' \sqsubseteq M' \sqcup \forall R.B \in \mathcal{S}(v)$ |
| | then | let $\langle u, \mathsf{core}' \rangle = \mathsf{strategy}(\exists R.A, \mathcal{B}_k, \mathcal{D})$ where |
| | | $\mathcal{B}_k = \{B \mid \top \sqsubseteq \forall R.B \in \mathcal{S}(v)$ |
| | | if $u$ is new, let $\mathsf{core}(u) = \mathsf{core}'$ |
| | | and $\mathcal{S}(u) = \{\top \sqsubseteq L \mid L \in \mathsf{core}(u)\}$ |
| | | add $\langle v, u, \exists R.A \rangle$ to $\mathcal{E}$ and |
| | | for each $L \in \{\{A\} \cup \mathcal{B}_p\} \backslash \mathsf{core}(v)$ add $L \sqsubseteq L$ to $\mathcal{S}(u)$ |

Figure A.10: Rules of inference for $\mathcal{ALCH}$

side of the ontology axiom). Rule $\mathsf{R}_{\exists}^{-}$ plays the same role with regard to R4, and it suffers major changes because the second premise now can have disjunctions on the right-hand side (this is why negation is introduced). There is a similar analogy between rule $\mathsf{R}_{\forall}$ and R3. Finally, we then have that rule $\mathsf{R}_{\exists}^{\bot}$ corresponds exactly to R2, and rule $\mathsf{R}_{\mathsf{A}}^{-}$ is not required in Figure A.5, because there are no negative literals.

The calculus is again worst-case optimal, and it reduces to the Horn-$\mathcal{ALCH}$ approach for Horn-$\mathcal{ALCH}$ ontologies, as well as to the $\mathcal{ELH}$ approach in $\mathcal{ELH}$ ontologies (because the method for Horn-$\mathcal{ALCH}$ reduces to that of $\mathcal{ELH}$). Moreover, one-pass classification is retained.

## A.6 The Calculus for $\mathcal{ALCHI}$

The article examined in this task [125] introduces an algorithm for reasoning in $\mathcal{ALCHI}$ which makes use of a new, general framework for consequence-based reasoning. This framework has been introduced in Section 3.2, so in this section we highlight the aspects that are different in the $\mathcal{ALCHI}$ calculus. One of the key differences is that this calculus uses DL notation for representing ontologies, context clauses, edge labels (which have now labels of the form $\exists R.A$), and context cores (which are conjunctions of concept names). The inference rules of the calculus are given in Figure A.10.

The authors of this paper claim that previous methods for consequence-based reasoning can be seen as analogous to their own method with a particular expansion strategy. Here we explore this claim and attempt to re-write each of the CB calculi discussed above in terms of this new framework, and in particular, we point out which expansion strategy would correspond to each method. This is of special interest to us since our calculus reduces to the $\mathcal{ALCHIQ}^+$ calculus in absence of nominals, which in turn reduces to this $\mathcal{ALCHI}$ calculus in the absence of number restrictions [20]. Thus, this comparison constitutes a detailed exploration of the relation between our calculus and previous existing CB calculi.

## Original $\mathcal{EL}$ calculus

| | |
|---|---|
| CR'1 | If $\top \sqsubseteq B \in \mathcal{S}(v_A)$ and $B \sqsubseteq C \in \mathcal{O}$, then add $\top \sqsubseteq C$ to $\mathcal{S}(v_A)$. |
| CR'2 | If $\top \sqsubseteq B, \top \sqsubseteq C \in \mathcal{S}(v_A), B \sqcap C \sqsubseteq D \in \mathcal{O}$ then add $\top \sqsubseteq D$ to $\mathcal{S}(v_A)$. |
| CR'3 | If $\top \in B \in \mathcal{S}(v_A), B \sqsubseteq \exists R.C \in \mathcal{O}$, then add an edge labelled $\exists R.C$ between context $v_A$ and context $v_C$. |
| CR'4 | If there is an edge $\exists R.B$ between contexts $v_A$ and $v_B$, and $\top \in C \in \mathcal{S}(B)$, and $\exists R.C \sqsubseteq D \in \mathcal{O}$, then add $\top \sqsubseteq D$ to $\mathcal{S}(v_A)$. |

Figure A.11: Version of the first $\mathcal{EL}$ calculus adapted to the context-based framework.

The original calculi for the $\mathcal{EL}$ family, discussed in Appendix A.1, can be seen as implementing the framework of this paper in the following way: for each concept name $A$, a context $u_A$ with core $A$ is introduced. Each concept $C$ in $\mathcal{S}(A)$ can be seen as a clause $\top \sqsubseteq C$ in the set of clauses $\mathcal{S}(u_A)$. Each element $(A, B) \in \mathcal{E}(R)$ can be seen as an edge labeled with $\exists R.B$ between contexts $u_A$ and $u_B$. If we rewrite the rules of Figure A.1 with this formalisation, we see that they are special cases of those in Figure A.11. For the sake of brevity we have only written the rules that correspond to $\mathcal{EL}$. Indeed, CR'1 and CR'2 correspond exactly to applications of Hyper; CR'3 is a Hyper step (for adding $\exists R.C$ to the clause) plus a Succ step, and CR'4 is also a Hyper step plus a Pred step: observe that $\exists R.C \sqsubseteq D$ is represented as $C \sqsubseteq \forall R^-.D$, so first $C$ is resolved to generate $\top \sqsubseteq \forall R^-.D$ in $\mathcal{S}(v_B)$, then Pred leads to the required result. The strategy used corresponds clearly to the so-called *cautious* strategy, which introduces, precisely, a context for each concept name.

I'1' For each $v_M$ and each $A$ in $M$, add $\top \sqsubseteq A$ to $\mathcal{S}(v_M)$.

R'1 If $\top \sqsubseteq A_i \in \mathcal{S}(v_M)$ for each $i$ such that $\bigsqcap A_i \sqsubseteq C \in \mathcal{O}$, then add $\top \sqsubseteq C$ to $\mathcal{S}(v_M)$.

R'2 If $\top \sqsubseteq \exists R.N \in \mathcal{S}(v_M)$ and $\top \sqsubseteq \bot \in \mathcal{S}(v_N)$, then add $\top \sqsubseteq \bot$ to $\mathcal{S}(v_N)$.

R'3 If $\top \sqsubseteq \exists R_1.N \in \mathcal{S}(v_M)$, $\top \sqsubseteq \forall R_2.A \in \mathcal{S}(v_M)$ and $R_1 \sqsubseteq R_2 \in \mathcal{O}$, then add $\top \sqsubseteq R_1.(N \sqcap A)$ to $\mathcal{S}(v_N)$.

R'4 If $\top \sqsubseteq \exists R_1.N \in \mathcal{S}(v_M)$, $\top \sqsubseteq \exists \forall R_2.A \in \mathcal{S}(v_N)$, and $R_1 \sqsubseteq R_2 \in \mathcal{O}$, then add $\top \sqsubseteq A$ to $\mathcal{S}(v_M)$.

Figure A.12: Version of the Horn-$\mathcal{SHIQ}$ calculus adapted to the context-based framework

## Horn-$\mathcal{SHOIQ}$ calculus

The Horn-$\mathcal{SHIQ}$ method can be rewritten in a similar way. In this case, however, it is easier to see this calculus as implementing the so-called *eager* strategy, which introduces a new context for every conjunction of concept names, because inferences in the calculus contain conjunctions on the left-hand side. A difference with respect to the context-based framework, however, is that we also allow conjunctions in the right-hand side, inside existential restrictions.

We again rewrite the calculus using the context-based framework; the result is given in Figure A.12. Rule I'1 implements the final part of the Succ rule, where new contexts are initialised. Rule I'2 is hidden in the application of Hyper: in the original algorithm, for axioms of type $\top \sqsubseteq C$ in the ontology, we require $M \sqsubseteq \top$, but now we can write directly $\top \sqsubseteq C$ with Hyper in context $v_M$. Consequences like $M \sqsubseteq \top$ may not be derived, but the completeness theorem guarantees that they are entailed. Observe that rule R'1 corresponds clearly to Hyper. Rule R'2 corresponds to a special application of the Pred rule: rather than following an edge, we apply Pred from a context $v_N$ as long as there is another context with $\exists R.N$; we will next observe that these contexts play the role of edges (though edges improve performance, since thanks to them, we do not have to search across all contexts). Rule R'3 can be seen as the first part of the Succ rule, that is, the part of the Succ rule which creates the edge, but as we have discussed, rather than creating explicitly the edge, a new atom is created $\top \sqsubseteq \exists R.(N \sqcap A)$ which "points towards" the context $v_{N \sqcap A}$. Finally, rule R'4, corresponds clearly to the Pred rule. We have ignored rules R'5 and R'6, since they have no analogous in the method of Figure A.10, but it is easy to see that they could be rewritten using the context formalism in an way similar to the previous rules.

## $\mathcal{ELHO}$ calculus

The $\mathcal{ELHO}$ method is discussed here because, even though the rules in Figure A.10 do not have support for nominals, the discussion about how they can be added to this framework produces some conclusions that can be useful in further work. One key feature to deal with nominals is the use of preconditions of the form $G$ : in front of inferences. We could represent this directly in the inferences written in the contexts; however, we also need to represent reachability inferences of the form $C \rightsquigarrow D$, and while we could represent these inferences also *in* the contexts, there is a more natural way of representing them. Indeed, as we will see below, a context $u_D$ is reachable (through edges) from context $u_C$ if and only if $C \rightsquigarrow D$. Therefore, it is simpler to let reachability relations be represented implicitly as we draw edges between the contexts. But reachability relations depend on preconditions, so in order for this to make sense, there must be one context structure for each possible precondition. This brings us closer to the algorithm in Figure A.10, for then we can ignore the preconditions $G$: in exchange for assigning one of these preconditions to each graph (and assuming that the precondition applies to every inference of that context structure). For this reason, we write the rules in Figure A.13 without preconditions, as we assume they apply to each context structure. Observe now that the inclusions written as part of the initialisation correspond to *different contexts in different graphs*, so now there must be a rule that generates new contexts *within* a graph.

Rule $\mathsf{R'}_{\sqsubseteq}$ is equivalent to the Hyper rule, but also to part of the Pred rule. The only difference is that in this calculus only one premise of the form $\top \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$ is used, instead of $n$ premises of the form $\top \sqcap A_1, \ldots, \top \sqcap A_n$; this is simply because of the differences in the normal form used in each algorithm. Rule $\mathsf{R'}^{-}_{\sqcap}$ is required for completeness in the normal form chosen. Rule $\mathsf{R'}^{+}_{\rightsquigarrow}$ is analogous to the first part of the Succ which generates a new context. Rule $\mathsf{R'}^{-}_{\rightsquigarrow}$ complements the previous and corresponds to the second part of the Succ rule, which initialises the new context. Rule $\mathsf{R'}^{+}_{\top}$ adds a rule which is not required in Figure A.10, again because the completeness theorem guarantees that these inclusions are implicitly entailed. Rule $\mathsf{R'}^{+}_{\sqcap}$ can be seen again as part of the rule Hyper. Finally, rule $\mathsf{R'}_{\{\}}$ deals with the reasoning with nominals. Once again, the strategy used is the eager strategy.

## Horn-$\mathcal{SROIQ}$ calculus

The calculus discussed in Appendix A.2 seems to use an eager strategy, initialising all contexts. However, it does not use preconditions of the form $G$: and at first glance, it

$\mathsf{R'}_{\sqsubseteq}$ If $\top \sqsubseteq D \in \mathcal{S}(v_C)$ and $D\top \sqsubseteq E \in \mathcal{O}$ then add $\top \to E$ in $\mathcal{S}(v_C)$.

$\mathsf{R'}_{\sqcap}^{-}$ If $\top \sqsubseteq D_1 \sqcap D_2$, add $\top \sqsubseteq D_1$ and $\top \sqsubseteq D_2$ to $\mathcal{S}(v_C)$.

$\mathsf{R'}_{\rightsquigarrow}^{+}$ If $\top \sqsubseteq \exists R.D \in \mathcal{S}(v_C)$ then add a context $v_D$ in the same structure, and an edge $(v_C, v_D)$ labelled $\exists R.D$.

$\mathsf{R'}_{\rightsquigarrow}^{-}$ For each context $v_D$ in each context structure, add $\top \sqsubseteq D$ to $\mathcal{S}(v_D)$.

$\mathsf{R'}_{\top}^{+}$ For each context $v_C$ in each context structure, add $\top \sqsubseteq \top$ if $\top$ occurs in $\mathcal{O}$.

$\mathsf{R'}_{\sqcap}^{+}$ If $\top \sqsubseteq D_1 \in \mathcal{S}(C)$ and $\top \sqsubseteq D_2 \in \mathcal{S}(C)$, then add $\top \sqsubseteq D_1 \sqcap D_2$ if $D_1 \sqcap D_2$ occurs in $\mathcal{O}$.

$\mathsf{R'}_{\exists}^{+}$ If $\top \sqsubseteq \exists R.D \in \mathcal{S}(C)$ and $\top \sqsubseteq E \in \mathcal{S}(D)$, then add $\top \sqsubseteq \exists R.E \in \mathcal{S}(C)$ if $R.E$ occurs in $\mathcal{O}$

$\mathsf{R'}_{\{\}}$ If $\top \sqsubseteq \{o\} \in \mathcal{S}(C)$ and $\top \sqsubseteq \{o\} \in \mathcal{S}(D)$, then add $\top \sqsubseteq D \in \mathcal{S}(C)$.

Figure A.13: Version of the $\mathcal{ELHO}$ calculus adapted to the context-based framework

does not seem to use more than one context structure, even though it uses nominals. But this turns out to be misleading, as this algorithm cannot be used for one-pass classification: the reason for this is that in order to check an inclusion of type $A \sqsubseteq B$, one would need to add the axiom $\mathsf{rel}(A)$ claiming that $A$ is realised (i.e. not empty), but this cannot be used as a global fact in order to check whether $C \sqsubseteq D$ for two unrelated concepts $C, D$, since it may be the case that $\mathcal{I} \models C \sqsubseteq D$ in every model $\mathcal{I}$ where $A$ is realised, but $\mathcal{O} \not\models C \sqsubseteq D$, in general. Therefore, the appropriate way of comparing this calculus to that in Figure A.10 is to see it as a special case of the comparison given for the $\mathcal{ELHO}$ calculus, in which the root context is $\top$ i.e. we only assume that there is at least "some" element which is realised in the model. Observe that this is a tautology, so no additional assumptions are introduced. Other than this, we have already compared the rules in this algorithm with those in the $\mathcal{EL}$ version of the $\mathcal{ELHO}$ procedure, so the reduction to the context-based framework is equivalent. The "reachability" relation is once again captured by those atoms that end up connected to a nominal or the original, the "root" context $\top$, and we can simply assume that each time the predicate $\mathsf{same}$ is generated, this is immediately followed by the application of rule $\mathsf{M23}$ so that the all possible consequences from this atom with $\mathsf{same}$ are generated, and therefore the atom need not be stored.

## $\mathcal{ALCH}$ calculus

The method presented in Section 3.1.4 can be transformed into the context framework in a way which is very similar to that used for the Horn-$\mathcal{SHIQ}$ method above, given the similarities between both calculi. For the sake of simplicity, we will ignore role inclusions. Even though the formalism in Figure A.9 uses negative literals, a translation can still be devised if we assume that negative atoms are encoded away in the usual way, that is, replacing $\neg A$ with a new concept name $\bar{A}$ such that $A \sqcap \bar{A} \sqsubseteq \bot$ is added to the ontology. The reason why this is not done already in the calculus of Figure A.9 is that even if negative literals are eliminated from the input, they will be produced by some inference steps. In our translation, we assume that these replacement of negative literals by positive literals with a new "complement" concept are also part of the inference rules, so we avoid generating negative literals. The translation is given in Figure A.14.

We can observe, therefore, that a rule $\mathsf{R'}^{-}_{A}$ is no longer required after making this change. Rule $\mathsf{R'}^{+}_{\mathsf{A}}$ in Figure A.14 stands once again for the last part of Succ rule; rule $\mathsf{R'}^{n}_{\sqcap}$ is clearly the Hyper rule, and the same is true for rule $\mathsf{R'}^{+}_{\exists}$. Rule $\mathsf{R'}^{-}_{\exists}$ contains first a case of Hyper where $\top \sqsubseteq N \sqcup A$ uses axiom $A \sqsubseteq \forall R^{-}.B$ to derive $\top \sqsubseteq N \sqcup \forall R^{-}.B$. Then it works like a generalisation of rule Pred. Observe that in Figure A.10, rule Pred is never applied unless the head of the main premise consists only of elements of the form $\forall R^{-}.C$, while here we can apply it to inclusions with a right-hand side of the form $N \sqcup \forall R^{-}.B$. Thus, in order to continue exploring the consequences of the head $N$, we add $\exists R.(K \sqcap \bar{A})$, which is analogous to the part of the Succ rule that draws a link to the context $v_{K \sqcap \bar{A}}$, which by rule $\mathsf{R'}^{+}_{\mathsf{A}}$ will have $\top \sqsubseteq \bar{A}$, and also $\top \sqsubseteq K$, like $\mathcal{S}(v_K)$, so it will eventually generate $\top \sqsubseteq N \sqcup A$; then, Hyper applied to this and $\top \sqsubseteq \bar{A}$ with axiom $A \sqcap \bar{A} \sqsubseteq \top$ will generate $\top \sqsubseteq N$; and hence we can now explore the consequences of $N$. Rule $\mathsf{R'}^{\bot}_{\exists}$ is clearly another example of Pred, and rule $\mathsf{R'}_{\forall}$ is analogous to Succ. It is easy to see from Figure A.14 that the strategy used in this situation is, again, the eager strategy.

$\mathsf{R'}^{+}_{\mathsf{A}}$ For each $v_H$ and each $A$ in $H$, add $\top \sqsubseteq A$ to $\mathcal{S}(v_H)$.

$\mathsf{R'}^{n}_{\sqcap}$ If $\top \sqsubseteq N_i \sqcup A_i \in \mathcal{S}(v_H)$ for each $i$ such that $\bigsqcap_{i=1}^{n} A_i \sqsubseteq M \in \mathcal{O}$,
then add $\top \rightarrow \bigsqcup_{i=1}^{n} N_i \sqcup M$ to $\mathcal{S}(v_H)$

$\mathsf{R'}^{+}_{\exists}$ If $\top \sqsubseteq N \sqcup A \in \mathcal{S}(v_H)$ and $A \sqsubseteq \exists R.B \in \mathcal{O}$
then add $\top \sqsubseteq \exists R.B$ to $\mathcal{S}(v_H)$.

$\mathsf{R'}^{-}_{\exists}$ If $\top \sqsubseteq M \sqcup \exists R.K \in \mathcal{S}(v_H)$ and $\top \sqsubseteq N \sqcup A \in \mathcal{S}(v_K)$
and $\{\exists S.A \sqsubseteq B, R \sqsubseteq S\} \in \mathcal{O}$
then add $\top \sqsubseteq M \sqcup B \sqcup \exists R.(K \sqcap \bar{A})$

$\mathsf{R'}^{\perp}_{\exists}$ If $\top \sqsubseteq M \sqcup \exists R.K \in \mathcal{S}(v_H)$ and $\top \sqsubseteq \perp \in \mathcal{S}(v_K)$
then add $\top \sqsubseteq M$ to $\mathcal{S}(v_M)$

$\mathsf{R'}_{\forall}$ If $\top \sqsubseteq M \sqcup \exists R.K \in \mathcal{S}(v_H)$ and $\top \sqsubseteq N \sqcup A \in \mathcal{S}(v_H)$
and $\{\exists S.A \sqsubseteq B, R \sqsubseteq S\} \in \mathcal{O}$
then add $\top \sqsubseteq M \sqcup N \sqcup \exists R.(K \sqcap B)$ to $\mathcal{S}(v_H)$.

Figure A.14: Version of the $\mathcal{ALCH}$ calculus adapted to the context-based framework

# Bibliography


[1] Carlos Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, ILLC, University of Amsterdam, 2000. ILLC Dissertation Series 2000–5.

[2] Ana Armas Romero, Bernardo Cuenca Grau, and Ian Horrocks. MORe: Modular Combination of OWL Reasoners for Ontology Classification. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *Proc. of the 11th International Semantic Web Conference (ISWC'12)*, volume 7649 of *LNCS*, pages 1–16, Boston, November 11–15 2012. Springer.

[3] Ana Armas Romero, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. Module extraction in expressive ontology languages via datalog reasoning. *J. of Artificial Intelligence Research*, 55:499–564, 2016.

[4] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In L. Pack Kaelbling and A. Saffiotti, editors, *Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 364–369, Edinburgh, July 30–August 5 2005. Morgan Kaufmann Publishers.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.

[6] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. In B. Nebel, C. Rich, and W. Swartout, editors, *Proc. of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281. Morgan Kaufmann Publishers, San Francisco, October 1992.

[7] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[8] Franz Baader. Terminological Cycles in a Description Logic with Existential Restrictions. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 325–330, Acapulco, Mexico, 2003. Morgan Kaufmann Publishers.

[9] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.

[10] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, Cambridge, 2017.

[11] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.

[12] Franz Baader, Carsten Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. SV, 2006.

[13] Franz Baader and Ulrike Sattler. Expressive number restrictions in description logics. *J. of Logic and Computation*, 9(3):319–350, 1999.

[14] L. Bachmair and H. Ganzinger. Equational Reasoning in Saturation-Based Theorem Proving. In W. Bibel and P.H. Schmidt, editors, *Automated Deduction—A Basis for Applications*, pages 353–397. Kluwer, Dordrecht, 1998.

[15] L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier Science, Amsterdam, 2001.

[16] Samantha Bail, Bijan Parsia, and Ulrike Sattler. Extracting finite sets of entailments from OWL ontologies. In Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev, editors, *Proceedings of the 24th International Workshop on Description Logics (DL 2011)*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[17] James P. Balhoff. The phenoscape knowledgebase: Tools and apis for computing across phenotypes from evolutionary diversity and model organisms. In Pankaj Jaiswal, Robert Hoehndorf, Cecilia N. Arighi, and Austin Meier, editors, *Proceedings of the Joint International Conference on Biological Ontology and BioCreative*, volume 1747 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[18] Andrew Bate, Boris Motik, Bernardo Cuenca Grau, Franstisek Simancik, and Ian Horrocks. Extending consequence-based reasoning to SRIQ. In *Proc. of the 15th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'16)*. pp 187-96, AAAI, 2016.

[19] Andrew Bate, Boris Motik, Bernardo Cuenca Grau, František Simančík, and Ian Horrocks. Extending Consequence-Based Reasoning to $\mathcal{SRIQ}$. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Proc. of the 15th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'16)*, pages 187–196, Palo Alto, April 25–29 2016. AAAI Press.

[20] Andrew Bate, Boris Motik, Bernardo Cuenca Grau, David Tena Cucala, Frantisek Simančík, and Ian Horrocks. Consequence-based reasoning for description logics with disjunctions and number restrictions. *J. Artif. Intell. Res.*, 63:625–690, 2018.

[21] P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA '96)*, volume 1126 of *LNCS*, pages 1–17, Berlin, September 30–October 3 1996. Springer.

[22] P. Baumgartner, U. Furbach, and I. Niemelä. Hyper tableaux. Technical Report 8/96, Universität Koblenz-Landau, Institut für Informatik, 1996. to appear in proceedings of JELIA'96.

[23] Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[24] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI'84)*, pages 34–37, 1984.

[25] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.

[26] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[27] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI'04)*, pages 298–302, Amsterdam, 2004. IOS Press.

[28] Markus Brenner and Birte Glimm. Embracing change by abstraction materialization maintenance for large aboxes. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, (IJCAI 2018)*, pages 1767–1773, Palo Alto, 2018. AAAI Press.

[29] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. Dl-learner - A framework for inductive learning on the semantic web. *J. Web Semant.*, 39:15–24, 2016.

[30] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 10th Ital. Conf. on Database Systems (SEBD'02)*, pages 161–168, 2002.

[31] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi. Data integration through DL-Lite$_{\mathcal{A}}$ ontologies. In *Proc. of SDKB 2008*, volume 4925 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 2008.

[32] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tailoring OWL for data intensive ontologies. In *Proc. of the 1st Int. Workshop on OWL Experiences and Directions (OWLED'05)*, 2005.

[33] David Carral, Irina Dragoste, and Markus Krötzsch. The combined approach to query answering in horn-alchoiq. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference*, pages 339–348, Palo Alto, 2018. AAAI Press.

[34] David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Sequoia: A consequence based reasoner for SROIQ. In Mantas Simkus and Grant E. Weddell, editors, *Proceedings of the 32nd International Workshop on Description Logics 2019.*, volume 2373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[35] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res.*, 31:273–318, 2008.

[36] H. de Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-Based Methods for Modal Logics. *Logic Journal of the IGPL*, 8:265–292, 2000.

[37] S. Demri and H. de Nivelle. Deciding Regular Grammar Logics with Converse Through First-Order Logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005.

[38] S. Derriere, A. Richard, and A. Preite-Martinez. An ontology of astronomical object types for the virtual observatory. *Proc. of Special Session 3 of the 26th meeting of the IAU: Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities*, 2006.

[39] F. M. Donini, M. Lenzerini, D. Nardi, Hollunder B, W. Nutt, and A. M. Spacamella. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2–3):309–327, 1992.

[40] Francesco M. Donini and Fabio Massacci. EXPTIME tableaux for ALC. *Artif. Intell.*, 124(1):87–138, 2000.

[41] Thomas Eiter, Georg Gottlob, and Heikki Mannilla. Disjunctive Datalog. *ACM Trans. on Database Systems*, 22(3):364–418, 1997.

[42] Cristina Feier, David Carral, Giorgio Stefanoni, Bernardo Cuenca Grau, and Ian Horrocks. The combined approach to query answering beyond the OWL 2 profiles. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 2971–2977, Buenos Aires, Argentina, 2015. AAAI Press.

[43] H. Ganzinger and H. de Nivelle. A Superposition Decision Procedure for the Guarded Fragment with Equality. In *Proc. of the 14th IEEE Symposium on Logic in Computer Science (LICS '99)*, pages 295–305, Washington, July 2–5 1999. IEEE Computer Society.

[44] Tom Gardiner, Dmitry Tsarkov, and Ian Horrocks. Framework for an automated comparison of description logic reasoners. In *Proc. of the 5th International Semantic Web Conference (ISWC'06)*, volume 4273 of *LNCS*, pages 654–667, Heidelberg, 2006. Springer.

[45] L. Georgieva, U. Hustadt, and R. A. Schmidt. Hyperresolution for Guarded Formulae. *J. Symb. Comput.*, 36:163–192, 2003.

[46] Birte Glimm, Ian Horrocks, Boris Motik, Rob Shearer, and Giorgos Stoilos. A novel approach to ontology classification. *J. of Web Semant.*, 14:84–101, 2012.

[47] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.

[48] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. Conjunctive query answering for the description logic SHIQ. *J. Artif. Intell. Res.*, 31:157–204, 2008.

[49] C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Michie, editors, *Proc. of the 4th Annual Machine Intelligence Workshop*, pages 183–208, Edinburgh, 1969. Edinburgh University Press.

[50] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The RacerPro knowledge representation and reasoning system. *J. Web Semant.*, 3:267–277, 2012.

[51] Jan Hladik. A tableau system for the description logic $\mathcal{SHIO}$. In Ulrike Sattler, editor, *Contributions to the Doctoral Programme of the 2nd Int. Joint Conf. on Automated Reasoning (IJCAR'04)*, volume 106 of *CEUR Workshop Proceedings*, pages 21–25. CEUR-WS.org, 2004.

[52] Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of the German Workshop on Artificial Intelligence*, pages 38–47. Springer, 1990.

[53] Bernhard Hollunder and Franz Baader. Qualifying number restrictions in concept languages. Technical Report RR-91-03, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern (Germany), 1991. An abridged version appeared in Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91).

[54] I. Horrocks and P. F. Patel-Schneider. Evaluating optimised decision procedures for propositional modal $\mathbf{K}_{(\mathbf{m})}$ satisfiability. In I. Gent, H. van Maaren,

and T. Walsh, editors, *Highlights of Satisfiability Research in the Year 2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*. ISO Press, Amsterdam, 2000.

[55] Ian Horrocks, Oliver Kutz, and Uli Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67, Palo Alto, 2006. AAAI Press.

[56] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*, pages 199–204. Morgan Kaufmann, Los Altos, 2001.

[57] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. LTCS-Report 02-06, Lehrstuhl für Automatentheorie Institut für Theoretische Informatik, TU Dresden, 2002.

[58] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artif. Intell.*, 160:79–104, 2004.

[59] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 448–453, Palo Alto, 2005. AAAI Press.

[60] Ian Horrocks and Ulrike Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *J. Autom. Reason.*, 39:249–276, 2007.

[61] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[62] U. Hustadt and R. A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In R. Caferra and G. Salzer, editors, *Selected Papers from Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 191–205. Springer, Berlin, 1999.

[63] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog programs. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'04)*, pages 152–162, Palo Alto, 2004. AAAI Press.

[64] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 466–471, 2005.

[65] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. of Automated Reasoning*, 39(3):351–384, 2007.

[66] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Deciding Expressive Description Logics in the Framework of Resolution. *Inf. Comput.*, 206:579–601, 2008.

[67] Ullrich Hustadt and Renate A. Schmidt. Using Resolution for Testing Modal Satisfiability and Building Models. *J. Autom. Reason.*, 28:205–232, 2002.

[68] Nikoo Zolfaghar Karahroodi and Volker Haarslev. A consequence-based algebraic calculus for $\mathcal{SHOQ}$. In *Proceedings of the 30th International Workshop on Description Logics*, volume 1879 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[69] Y. Kazakov. Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In Craig Boutilier, editor, *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09)*, pages 2040–2045, Palo Alto, July 11–17 2009. AAAI Press.

[70] Yevgeny Kazakov. $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$ are Harder than $\mathcal{SHOIQ}$. In Gerhard Brewka and Jérôme Lang, editors, *Proc. of the 11th Int. Joint Conf. on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 274–284, Palo Alto, 2008. AAAI Press.

[71] Yevgeny Kazakov and Pavel Klinov. Bridging the Gap between Tableau and Consequence-Based Reasoning. In Meghyn Bienvenu, Magdalena Ortiz, Riccardo Rosati, and Mantas Simkus, editors, *Proc. of the 27th Int. Workshop on Description Logics (DL'14)*, volume 1193 of *CEUR Workshop Proceedings*, pages 579–590, Vienna, July 17–20 2014. CEUR-WS.org.

[72] Yevgeny Kazakov, Markus Kroetzsch, and Frantisek Simancik. Practical reasoning with nominals in the $\mathcal{EL}$ family of description logics. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, Palo Alto, 2012. AAAI Press.

[73] Yevgeny Kazakov, Markus Krötzsch, and Frantivsek Simanvcík. Practical Reasoning with Nominals in the EL Family of Description Logics. *KR*, 2012.

[74] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. The incredible ELK — from polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. *J. Autom. Reason.*, 53:1–61, 2014.

[75] Yevgeny Kazakov and Boris Motik. A resolution-based decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06)*, pages 662–677, Heidelberg, 2006. Springer.

[76] Yevgeny Kazakov and Boris Motik. A resolution-based decision procedure for $\mathcal{SHOIQ}$. *J. Autom. Reason.*, 40:89–116, 2008.

[77] Evgeny Kharlamov, Dag Hovland, Ernesto Jiménez-Ruiz, Davide Lanti, Christoph Pinkel, Martin Rezk, Martin G. Skjæveland, Evgenij Thorstensen, Guohui Xiao, Dmitriy Zheleznyakov, Eldar Bjørge, and Ian Horrocks. Ontology based access to exploration data at Statoil. In *Proc. of the 14th International Semantic Web Conference (ISWC'15)*, volume 9367 of *Lecture Notes in Computer Science*, pages 93–112. Springer, 2015.

[78] Evgeny Kharlamov, Ernesto Jiménez-Ruiz, Christoph Pinkel, Martin Rezk, Martin G. Skjæveland, Ahmet Soylu, Guohui Xiao, Dmitriy Zheleznyakov, Martin Giese, Ian Horrocks, and Arild Waaler. Optique: Ontology-based data access platform. In *International Semantic Web Conference (Posters & Demos)*, volume 1486 of *CEUR Workshop Proceedings*, 2015.

[79] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in dl-lite. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference*, Toronto, Ontario, 2010. AAAI Press.

[80] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for horn description logics. In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI-07)*, pages 452–457. AAAI Press/The MIT Press, 2007.

[81] Lee Lacy, Gabriel Aviles, Karen Fraser, William Gerber, Alice Mulvehill, and Robert Gaskill. Experiences using OWL in military applications. In *Proc. of the 1st Int. Workshop on OWL Experiences and Directions (OWLED'05)*, volume 188 of *CEUR Workshop Proceedings*. CEUR Workshop Proceedings, 2005.

[82] C Le Duc, M Lamolle, and O. Curé. A Worst-case Optimal Algorithm for SHOIQ. https://hal.inria.fr/inria-00570161v2.

[83] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.

[84] Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 181–200. Springer, 1999.

[85] Carsten Lutz. An improved NExpTime-hardness result for description logic $\mathcal{ALC}$ extended with inverse roles, nominals, and counting. LTCS-Report 05-05, Institute for Theoretical Computer Science, Dresden University of Technology, 2005.

[86] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. LTCS-Report 02-04, Institute for Theoretical Computer Science, Dresden University of Technology, 2002.

[87] Carsten Lutz, Ulrike Sattler, and Lidia Tendera. The complexity of finite model reasoning in description logics. *Inf. Comput.*, 199(1-2):132–171, 2005.

[88] Nicolas Matentzoglu, Bijan Parsia, and Uli Sattler. OWL reasoning: Subsumption test hardness and modularity. *J. Autom. Reason.*, 60(4):385–419, 2018.

[89] Alejandro Metke-Jimenez and Michael Lawley. Snorocket 2.0: Concrete Domains and Concurrent Classification. In Samantha Bail, Birte Glimm, Rafael S. Gonçalves, Ernesto Jiménez-Ruiz, Yevgeny Kazakov, Nicolas Matentzoglu, and Bijan Parsia, editors, *Proc. of the 2nd Int. Workshop on OWL Reasoner Evaluation (ORE-2013)*, volume 1015 of *CEUR Workshop Proceedings*, pages 32–38, Ulm, Germany, July 22 2013.

[90] Alejandro Metke-Jimenez and Michael Lawley. Snorocket 2.0: Concrete domains and concurrent classification. In *Informal Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE-2013)*, volume 1015 of *CEUR Workshop Proceedings*, pages 32–38, 2013.

[91] Marvin Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*. The MIT Press, 1981. A longer version appeared in *The Psychology of Computer Vision* (1975). Republished in [25].

[92] Michael Mortimer. On languages with two variables. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:135–140, 1975.

[93] Boris Motik. *Reasoning in description logics using resolution and deductive databases*. PhD thesis, Karlsruhe Institute of Technology, Germany, 2006.

[94] Boris Motik. KAON2 - scalable reasoning over ontologies with large data sets. *ERCIM News*, 2008, 2008.

[95] Boris Motik, Yavor Nenov, Robert Edgar Felix Piro, and Ian Horrocks. Incremental update of datalog materialisation: the backward/forward algorithm. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1560–1568, Palo Alto, 2015. AAAI Press.

[96] Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large description logic ABoxes. In *Proc. of the 13th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Computer Science*, pages 227–241, 2006.

[97] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. *J. Artif. Intell. Res.*, 36:165–228, 2009.

[98] Linh Anh Nguyen. A Tableau Method with Optimal Complexity for Deciding the Description Logic $\mathcal{SHIQ}$. In Ngoc Thanh Nguyen, Tien Van Do, and Hoai An Le Thi, editors, *Advanced Computational Methods for Knowledge Engineering*, volume 479 of *SCI*, pages 331–342. Springer, Heidelberg, 2013.

[99] Linh Anh Nguyen. ExpTime tableaux with global state caching for the description logic $\mathcal{SHIO}$. *Neurocomputing*, 146:249–263, 2014.

[100] Linh Anh Nguyen and Joanna Golińska-Pilarek. An ExpTime Tableau Method for Dealing with Nominals and Qualified Number Restrictions in Deciding the Description Logic $\mathcal{SHOQ}$. *Fundam. Inform.*, 135:433–449, 2014.

[101] R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *J. of Symb. Comput.*, 19:312–351, 1995.

[102] R. Nieuwenhuis and A. Rubio. Paramodulation-Based Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 371–443. Elsevier Science, Amsterdam, 2001.

[103] Nils J. Nilsson. *The Quest for Artificial Intelligence, a History of Ideas and Achievements*. Cambridge University Press, 2009.

[104] A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 335–367. Elsevier Science, Amsterdam, 2001.

[105] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'10)*, pages 269–279, Palo Alto, 2010. AAAI Press.

[106] OWL 2 Web Ontology Language Overview. W3C Recommendation. `http://www.w3.org/TR/owl2-overview`, 27 October 2009. Accessed 02 February 2019.

[107] OWL 2 Web Ontology Language Profiles. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-profiles/`.

[108] OWL 2 Web Ontology Language Structural Specification and Functional-style Syntax. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-syntax/`.

[109] I Pratt-Hartmann. Complexity of the Two-Variable Fragment with Counting Quantifiers. *Journal of Logic, Language, and Information*, 14(3):369–395, 2005.

[110] M. Ross Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. The MIT Press, 1968.

[111] Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier, 2001.

[112] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of the 12th International Semantic Web Conference (ISWC'13)*, volume 8218 of *Lecture Notes in Computer Science*, pages 558–573. Springer, 2013.

[113] Ulrike Sattler and Moshe Y. Vardi. The hybrid $\mu$-calculus. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the First International Joint Conference of Automated Reasoning, IJCAR 2001*, volume 2083 of *Lecture Notes in Computer Science*, pages 76–91, Berlin, 2001. Springer.

[114] Viachaslau Sazonau and Uli Sattler. Mining hypotheses from data in OWL: advanced evaluation and complete construction. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *Proceedings of the 16th International Semantic Web Conference (ISWC 2017)*, volume 10587 of *Lecture Notes in Computer Science*, pages 577–593. Springer, 2017.

[115] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[116] R. A. Schmidt. Resolution is a decision procedure for many propositional modal logics. In M. Kracht, M. de Rijke, H. Wansing, and M Zakharyaschev, editors, *Advances in Modal Logic*, volume 87, pages 189–208. CSLI Publications, 1998.

[117] Renate A. Schmidt and Ullrich Hustadt. First-Order Resolution Methods for Modal Logics. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics—Essays in Memory of Harald Ganzinger*, volume 7797 of *LNCS*, pages 345–391, Berlin, 2013. Springer.

[118] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[119] Stefan Schulz, Boontawee Suntisrivaraporn, and Franz Baader. SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In *MEDINFO 2007 - Proceedings of the 12th World Congress on Health (Medical) Informatics - Building Sustainable Health Systems, 20-24 August, 2007, Brisbane, Australia*, volume 129 of *Studies in Health Technology and Informatics*, pages 802–806. IOS Press, 2007.

[120] Stephan Schulz. Simple and efficient clause subsumption with feature vector indexing. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *Lecture Notes in Computer Science*, pages 45–67. Springer, Berlin, 2013.

[121] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A highly-efficient owl reasoner. In *Proc. of the 5th Int. Workshop on OWL Experiences and Directions (OWLED'08)*, volume 432 of *CEUR Workshop Proceedings*, pages 26–27. CEUR-WS.org, 2008.

[122] František Simančík. Elimination of complex RIAs without automata. In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proc. of the 25th Int. Workshop on Description Logics (DL'12)*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, June 7–10 2012.

[123] F. Simančík, Y. Kazakov, and I. Horrocks. Consequence-Based Reasoning beyond Horn Ontologies. In Toby Walsh, editor, *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, pages 1093–1098, Palo Alto, July 16–22 2011. AAAI Press.

[124] František Simančík, Yevgeny Kazakov, and Ian Horrocks. Consequence-based reasoning beyond Horn ontologies. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, pages 1093–1098, 2011.

[125] František Simančík, Boris Motik, and Ian Horrocks. Consequence-based and fixed-parameter tractable reasoning in description logics. *Artif. Intell.*, 209:29–77, 2014.

[126] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *J. Web Semant.*, 5:51–53, 2007.

[127] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, OBI Consortium, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H Scheuermann, Nigam Shah, Patricia L Whetzel, and Suzanna Lewis. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.*, 25:1251–1255, 2007.

[128] R. M. Smullyan. *First Order Logic*. Springer, Berlin (Germany), 1968.

[129] Dagobert Soergel, Boris Lauser, Anita Liang, Frehiwot Fisseha, Johannes Keizer, and Stephen Katz. Reengineering thesauri for new applications: The AGROVOC example. *J. of Digital Information*, 4(4), 2004.

[130] W.D. Solomon, A. Roberts, J. E. Rogers, C. J. Wroe C.J., and A. L. Rector. Having our cake and eating it too: How the GALEN Intermediate Representation reconciles internal complexity with users' requirements for appropriateness and simplicity. In *Proc. of the Annual Fall Symposium of American Medical Informatics Association*, pages 819–823, Los Angeles, CA, USA, November 4–8 2000.

[131] Weihong Song, Bruce Spencer, and Weichang Du. Complete classification of complex ALCHO ontologies using a hybrid reasoning approach. In Thomas Eiter, Birte Glimm, Yevgeny Kazakov, and Markus Krötzsch, editors, *Informal Proceedings of the 26th International Workshop on Description Logics (DL 2013)*, volume 1014 of *CEUR Workshop Proceedings*, pages 942–961. CEUR-WS.org, 2013.

[132] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System description. *J. Web Semant.*, 27:78–85, 2014.

[133] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System description. *J. Web Semant.*, 27, 2014.

[134] David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Consequence-based reasoning for description logics with disjunction, inverse roles and nominals. In Alessandro Artale, Birte Glimm, and Roman Kontchakov, editors, *Proceedings of the 30th International Workshop on Description Logics (DL'17)*, volume 1879 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[135] David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. Consequence-based reasoning for description logics with disjunction, inverse roles, number restrictions, and nominals. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 1970–1976, Palo Alto, 2018. AAAI Press.

[136] David Tena Cucala, Bernardo Cuenca Grau, and Ian Horrocks. 15 years of consequence-based reasoning. In Carsten Lutz, Uli Sattler, Cesare Tinelli, Anni-Yasmin Turhan, and Frank Wolter, editors, *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, volume 11560 of *Lecture Notes in Computer Science*, pages 573–587, Cham, 2019. Springer.

[137] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.

[138] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen University, 2001.

[139] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06)*, volume 4130 of *LNAI*, pages 292–297, Heidelberg, August 17–20 2006. Springer.

[140] Dmitry Tsarkov and Ignazio Palmisano. Chainsaw: a metareasoner for large ontologies. In Ian Horrocks, Mikalai Yatskevich, and Ernesto Jiménez-Ruiz, editors, *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[141] Tania Tudorache, Csongor Nyulas, Natalya Fridman Noy, and Mark A. Musen. Using semantic web in ICD-11: three years down the road. In *Proc. of the 12th International Semantic Web Conference (ISWC'13)*, volume 8219 of *Lecture Notes in Computer Science*, pages 195–211. Springer, 2013.

[142] Jelena Vlasenko, Maryam Daryalal, Volker Haarslev, and Brigitte Jaumard. A Saturation-based Algebraic Reasoner for $\mathcal{ELQ}$. In Pascal Fontaine, Stephan Schulz, and Josef Urban, editors, *Proc. of the 5th Workshop on Practical Aspects of Automated (PAAR'16)*, volume 1635 of *CEUR Workshop Proceedings*, pages 110–124. CEUR-WS.org, 2016.

[143] Johanna Völker and Mathias Niepert. Statistical schema induction. In Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan, editors, *Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011)*, volume 6643 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2011.

[144] L Wos and G Robinson. Paramodulation and theorem-proving in first-order theories with equality. In D Michie and B Meltzer, editors, *Proceedings of the 4th Annual Machine Intelligence Workshop*. Edinburgh University Press, 1969.

[145] Haoruo Zhao, Uli Sattler, and Bijan Parsia. Avoiding subsumption tests during classification using the atomic decomposition. In Mantas Simkus and Grant E. Weddell, editors, *Proceedings of the 32nd International Workshop on Description Logics*, volume 2373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.