Pay-As-You-Go Consequence-Based Reasoning for the Description Logic $\mathcal{SROIQ}$

David Tena Cucala, Bernardo Cuenca Grau, Ian Horrocks

Department of Computer Science, University of Oxford, UK

Abstract

Consequence-based (CB) reasoners combine ideas from resolution and (hyper)tableau calculi for solving key reasoning problems in Description Logics (DLs), such as ontology classification. Existing CB reasoners, however, are only capable of handling DLs without nominals (such as $\mathcal{ALCHIQ}$), or DLs without disjunction (such as Horn-$\mathcal{ALCHOIQ}$). In this paper, we present a consequence-based calculus for concept subsumption and classification in the DL $\mathcal{ALCHOIQ}^+$, which 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: our calculus is worst-case optimal for all the well-known proper fragments of $\mathcal{ALCHOIQ}^+$. Furthermore, our calculus can be applied to DL reasoning problems other than subsumption and ontology classification, such as instance retrieval and realisation. We have implemented our calculus as an extension of Sequoia, a CB reasoner which previously supported ontology classification in $\mathcal{SRIQ}$. We have performed an empirical evaluation of our implementation, which shows that Sequoia offers competitive performance. Although there still remains plenty of room for further optimisation, the calculus presented in this paper and its implementation provide an important addition to the repertoire of reasoning techniques and practical systems for expressive DLs.

Keywords: Knowledge Representation and Reasoning, Ontologies, Automated Reasoning, Description Logics

1. Introduction

Description Logics (DLs) [3] are a prominent family of knowledge representation formalisms with well-defined semantics that have found applications in a wide range of domains. A DL ontology describes an application domain in terms of concepts (unary predicates), roles (binary predicates), and individuals (constants). The highly expressive DL $\mathcal{SROIQ}$ [24] provides the formal underpinning for the Web Ontology Language OWL 2 [52], which has been standardised by the World Wide Web Consortium (W3C). A key reasoning service for DLs is concept subsumption: given ontology $\mathcal{O}$ and concepts C and D, determine whether each instance of C is also an instance of D in every model of $\mathcal{O}$. In turn, ontology classification is the problem of computing all subsumption relationships between pairs of concepts named in an ontology.

Concept subsumption in $\mathcal{SROIQ}$ is 2NExpTime-complete; however, despite such high complexity, practical calculi for this logic and other related DLs have been developed and successfully implemented. Tableau and hypertableau calculi [12] underpin many of the currently available ontology reasoners, including FaCT++ [67], RacerPro [22], Pellet [60], HermiT [56], and Konclude [62]. To check whether a subsumption holds, 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. Tableau-based algorithms face two main sources of complexity [44]: non-determinism caused by disjunctive statements (*or-branching*), which can lead to frequent backtracking, and generation of very large models due to existential restrictions (*and-branching*). These problems are exacerbated in tasks such as classification, where a large number of subsumption checks must be performed. Tableau-based algorithms are typically not worst-case optimal for subsumption and consistency checking, but they have been heavily optimised to reduce the impact of or- and and-branching. In addition, these calculi use techniques to reduce the number of subsumption checks during classification, such as the enhanced traversal algorithm [4, 20]. The technique of global caching has been used to ensure worst-case optimal behaviour in DLs that are ExpTime-complete for concept subsumption [45, 46, 47]; 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. Furthermore, despite optimisations, the aforementioned sources of complexity can still compromise the performance of tableaux-based reasoners on complex ontologies.

A second category of DL reasoning calculi are based on first-order resolution [8]. These algorithms transform an ontology into a set of clauses, and then attempt to derive the empty clause using a resolution-based inference system [29, 16, 32, 54, 28, 17, 38,

Email addresses: david.tena.cucala@cs.ox.ac.uk (David Tena Cucala), bernardo.cuenca.grau@cs.ox.ac.uk (Bernardo Cuenca Grau), ian.horrocks@cs.ox.ac.uk (Ian Horrocks)

27, 43, 42, 30]. An advantage of this approach is that reasoning can be performed without grounding the clauses, thus avoiding the construction of an explicit representation of a model. We can identify two different resolution-based approaches to DL reasoning [29]. The first approach ensures termination and worst-case optimal behaviour while preserving completeness by constraining the application of resolution inferences by means of carefully defined orderings and selection functions [16, 32, 54, 28, 17, 38, 27]; the KAON2 reasoner [43] implements this approach for the Description Logic $\mathcal{SHIQ}$ [42]. The second approach uses resolution to simulate model-building (hyper)tableau techniques [29, 19], including blocking methods which ensure termination [30, 13]. It has been observed in practice that resolution and tableau algorithms complement each other nicely, because each method performs best on different types of input [63, 13]. On the one hand, tableau methods terminate as soon as they find a model of the ontology and hence they are better suited to problems where the input is satisfiable; on the other hand, resolution methods terminate as soon as the empty clause is derived, and therefore they perform better when the input is unsatisfiable.

Consequence-based (CB) calculi emerged as a promising approach to DL reasoning combining ideas from tableau and resolution. On the one hand, similarly to resolution, they derive formulae entailed by the ontology (thus avoiding the explicit construction of large models), and they are typically worst-case optimal for concept subsumption. On the other hand, clauses are organised into contexts arranged as a graph structure reminiscent of that used for model construction in (hyper)tableau; this prevents CB calculi from drawing many unnecessary inferences and yields a nice goal-oriented behaviour. Furthermore, in contrast to existing resolution and (hyper)tableau calculi, CB methods can verify a large number of subsumptions in a single execution, allowing for one-pass classification. Finally, CB calculi are very effective in practice, and systems based on them, such as ELK [37] and Snorocket [41], have shown outstanding performance. The first CB calculi were developed for the $\mathcal{EL}$ family of lightweight DLs [14, 2, 36], and were later extended to the more expressive logics Horn-$\mathcal{SHIQ}$ [34], Horn-$\mathcal{SROIQ}$ [51], $\mathcal{ALCH}$ [58], $\mathcal{SRIQ}$ [11], and $\mathcal{SHOI}$ [64].

CB calculi have also been used in combination with other reasoning methods. The CB calculus by Vlasenko et al. [68] for the DL $\mathcal{ELQ}$ relies on an Integer Linear Programming solver to efficiently deal with number restrictions; this approach has been extended to the DL $\mathcal{SHOQ}$ in [33] and implemented in the reasoner Avalanche [68], with encouraging results. The $\mathcal{SROIQ}$ reasoner Konclude [62] 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. [35] introduces a non-deterministic rule to handle disjunction, which is analogous to the disjunction rule used in tableaux. The reasoner MORe [1] exploits module extraction techniques [15] 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. As we discuss in Section 3, the interaction of these features causes a complexity jump from ExpTime [40, 23, 66] to NExpTime [39] for subsumption checking, and poses major practical challenges for reasoning algorithms [26, 38]. Furthermore, the presence of nominals in the ontology poses additional challenges specific to CB approaches; in particular, nominals can be used to express global constraints on models of the ontology, which are difficult to represent within the CB framework [36].

In Section 4 we present a consequence-based calculus for the DL $\mathcal{ALCHOIQ}^+$, which supports all Boolean connectives, role hierarchies, inverse roles, number restrictions, and nominals. By encoding role inclusion axioms using well-known methods [24, 57], our calculus extends to $\mathcal{SROIQ}$, which covers all of OWL 2 DL except for datatypes. Following Bate et al. [10], we encode derived consequences as clauses of first-order equational logic, similar to those used in resolution-based methods for reasoning in DLs [31], and we handle equality reasoning using a variant of ordered paramodulation. This allows us to exploit techniques from paramodulation-based theorem proving to show completeness and establish complexity bounds [8, 48, 49, 7]. To handle nominals, we allow for ground atoms in derived clauses, add new inference rules, and introduce a *root context* where all inferences on ground atoms are performed.

In Section 5 we introduce a subsumption algorithm that uses our calculus, and we show how this algorithm can be applied to solve other DL reasoning problems. We also illustrate how variants of the calculus and the algorithm based on it can achieve worst-case optimal performance for all proper fragments of $\mathcal{ALCHOIQ}^+$. In particular, we show how to perform classification in deterministic exponential time for all of $\mathcal{ALCHOI}$, $\mathcal{ALCHOQ}$, $\mathcal{ALCHIQ}^+$ and Horn-$\mathcal{ALCHOIQ}^+$, and in polynomial time for the lightweight DL $\mathcal{ELHO}$. Our algorithm is, however, not worst-case optimal for $\mathcal{ALCHOIQ}$, and it exhibits a worst-case running time in line with other existing tableau- and resolution-based approaches [44, 38]. The source of additional complexity is, however, very localised and only manifests itself in non-Horn ontologies when nominals, number restrictions and inverse roles interact simultaneously (a rare situation in practice).

In Section 6 we compare our approach to other CB calculi for DLs with nominals, and we also compare it to reasoning algorithms based on resolution and tableau for DLs with disjunction, inverse roles, number restrictions, and nominals. We point out the similarities and differences between these calculi, and we analyse our comparative strengths and weaknesses.

We have implemented our calculus as an extension of the $\mathcal{SRIQ}$ reasoner Sequoia [11]. In Section 7 we discuss in detail all modifications and novel features in our new version of Sequoia, which supports nominals and can perform additional reasoning tasks other than classification.

In Section 8, we describe the results of an experimental evaluation of Sequoia's classification performance on a large corpus of real-world ontologies. In our evaluation, we compared Sequoia with the state-of-the-art reasoners Pellet, FaCT++, HermiT, and Konclude. Our experiments show that Sequoia offers competitive performance, which is overall comparable on our corpus to that of HermiT. It is, however, worth noticing that the sets of ontologies that Sequoia and HermiT fail to classify have a small overlap, which reflects the significant differences between the underpinning calculi. Therefore, our results show that Sequoia provides an important addition to the repertoire of practical implementation techniques for DL reasoning which nicely complements previously existing approaches. Finally, to test the pay-as-you-go behaviour of Sequoia, we compared its performance with that of ELK and Snorocket on OWL 2 EL ontologies, and we could verify that the performance of Sequoia is comparable to that of the aforementioned specialised reasoners on the ontologies that they support.

This article is an extension of our earlier conference publication [65]. New material includes the implementation, optimisation, and evaluation of the calculus, as well as detailed proofs of all our results.

2. Preliminaries

In this section we recapitulate basic notions and introduce the notation used in the rest of our paper. Section 2.1 introduces many-sorted first-order equational logic. Following [10, 8, 48, 49, 7], we consider formulas where equality is the only predicate; as discussed in Section 2.1, this is without loss of generality since first-order atoms with arbitrary predicates can be converted into equality atoms while preserving satisfiability and entailment [48, 49]. Section 2.2 provides basic definitions related to term orderings. Finally, Section 2.3 provides an overview of the basics of DLs and defines a clausal form for DL ontologies.

2.1. Many-Sorted Clausal Equational Logic

A *many-sorted* signature Σ is a pair (Σ^S, Σ^F), where Σ^S is a non-empty set of *sorts*, and Σ^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 s, and suppose that sets of variables of different sorts are disjoint. If $\mathsf{s} \in \Sigma^S$, the set of *terms* of sort 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 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 s is often called an s-*term*. A *position* p of a term is a finite sequence of positive integers; we represent the empty position as ϵ 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 then $t|_p$ 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 mentions no variables. If s_1 is an s-subterm of term t at position p, and s_2 is an 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 simultaneously 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 so is each of its terms. 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 $1 \leq i \leq n$; furthermore, we assume that true is a p-constant. We refer to function symbols of sort 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 $A(t_1, \ldots, t_n)$. This correspondence allows us to transform formulas with arbitrary predicates into formulas with equality as the only predicate, while preserving entailment and satisfiability [49]. 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 a sentence of the form $\forall x_1, \ldots, x_n. (\Gamma \to \Delta)$ with $n \in \mathbb{N}_0$, where the *body* of the clause Γ is a finite conjunction of equalities, the *head* of the clause Δ 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 [7]. The empty conjunction is represented as $\top$, and the empty disjunction is represented as $\bot$. A formula (and in particular 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* 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 $\Gamma \to \Delta$ up to redundancy, we write $\Gamma \to \Delta \mathrel{\hat{\in}} N$. This notion of redundancy is the same as that in [11].

A *substitution* is a mapping σ from variables to terms such that variables $x \in \mathcal{X}_\mathsf{s}$ are mapped to terms of sort s, and all but finitely many variables are mapped to themselves by σ. The image of variable x by σ is represented as $x\sigma$. Substitution σ is often written 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 σ consists of all variables not mapped to themselves by σ. If α is either a term, a conjunction, a disjunction, or a clause, then $\alpha\sigma$ is the result of replacing every occurrence of x in α 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 s for each $\mathsf{s} \in \Sigma^S$. We assume that for each $\mathsf{s} \in \Sigma^S$ there is at least one ground 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 α be a ground conjunction, a ground disjunction, a (not necessarily ground) clause, or any set thereof; we say that $\mathcal{I}$ *satisfies* α if it does so according to the usual criteria, but assuming that each universally quantified variable of sort $\mathsf{s} \in \Sigma^S$ ranges over s-terms in HU. We say that $\mathcal{I}$ is a *model* of α, written $\mathcal{I} \models \alpha$, if $\mathcal{I}$ satisfies α. A set of clauses α *entails* a clause $\Gamma \to \Delta$, written $\alpha \models \Gamma \to \Delta$, if each model of α 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 $\succ$ 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 $\succ$ on X is a strict order on X such that, given any two distinct elements $a, b \in X$, either $a \succ b$ or $b \succ a$. 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 μ 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 μ_1 and μ_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 $t \in$ 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 σ. 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 $s \in$ 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 order on a set of function symbols. The *lexicographic path order* $\succ_{lpo}$ induced by $\succ$ is defined as follows: given terms $s = f(s_1, \ldots, s_m)$ and $t = g(t_1, \ldots, t_n)$, 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 can be shown that if $\succ$ is well-founded, then $\succ_{lpo}$ is a simplification order, and if $\succ$ is total, then $\succ_{lpo}$ is total [6].

2.3. $\mathcal{ALCHOIQ}^+$ Ontologies

Let $\mathbf{B}$, $\mathbf{S}$, and $\mathbf{I}$ be disjoint sets of *concept names*, *role names*, and *individual names*, respectively. The set of *roles* extends $\mathbf{S}$ with all expressions of the form S^- where $S \in \mathbf{S}$. The set of *concepts* is the minimal set containing: (i) each concept name, (ii) symbols $\top$ and $\bot$, (iii) $\neg C$ for every concept C, (iv) $C_1 \sqcap C_2$ and $C_1 \sqcup C_2$ for each pair of concepts C_1, C_2, (v) $\exists R.C$ and $\forall R.C$ for every role R and concept C, (vi) $\geqslant n\, R.C$ and $\leqslant n\, R.C$ for each $n \in \mathbb{N}_0$, each role R, and each concept C, (vii) $\{o\}$ for each individual name $o \in \mathbf{I}$, and (viii) $\exists\, \mathsf{Self}.R$ for each role R.

An $\mathcal{ALCHOIQ}^+$ *TBox* is a finite set of *axioms* of the form $C_1 \sqsubseteq C_2$, $R_1 \sqsubseteq R_2$, or $\mathsf{Disj}(R_1, R_2)$, where C_1 and C_2 are concepts, and R_1 and R_2 are roles. Note that $\mathcal{ALCHOIQ}^+$ is usually defined with some additional types of axioms, such as role reflexivity or asymmetry [5], which can be easily rewritten into axioms allowed by our definition in a way that preserves satisfiability and entailment. An $\mathcal{ALCHOIQ}^+$ *ABox* is a finite set of *assertions* of the form $C(o_1)$, $R(o_1, o_2)$, or $\neg R(o_1, o_2)$, with C a concept, R a role, and o_1, o_2 individual names. A *knowledge base* is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ a *TBox* and $\mathcal{A}$ an *ABox* in $\mathcal{ALCHOIQ}^+$. The first-order interpretation of $\mathcal{ALCHOIQ}^+$ knowledge bases is standard [5, Chapter 8]. Every $\mathcal{ALCHOIQ}^+$ knowledge base can be transformed in polynomial time into a *normal form* consisting of axioms of the forms DL1 to DL11 on the left-hand side of Table 1 [31]; the normalisation procedure is a variant of the structural transformation [50] used in first-order theorem proving.

The DL $\mathcal{SROIQ}$ [24] extends $\mathcal{ALCHOIQ}^+$ by allowing in the TBox *complex role inclusion axioms* (RIAs) of the form $R_1 \circ \cdots \circ R_n \sqsubseteq R$ for some $n \in \mathbb{N}$ and each R_i a role. Extending $\mathcal{ALCHOIQ}^+$ with RIAs leads to undecidability [25]; however, RIAs can be eliminated (with an exponential blowup) in knowledge bases satisfying certain global restrictions while preserving satisfiability and entailment [57]. In particular, RIA elimination can be combined with any worst-case optimal decision procedure for $\mathcal{ALCHOIQ}^+$ to obtain a worst-case optimal procedure for $\mathcal{SROIQ}$.

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

Table 1: Normalised $\mathcal{ALCHOIQ}^+$ axioms and transformation to DL-clauses. Each $B_{(i)}$ is a concept name in Σ_A; each $S_{(i)}$ is a role name in Σ_S; each f_i is a fresh successor function symbol in Σ_f; symbol o is a constant in Σ_c. Roles S_{B_2} in DL4 are fresh, contained in Σ_S, and determined by S and B_2.

A *DL signature* is a two-sorted signature $\Sigma = \{\Sigma^S, \Sigma^F\}$ where Σ^S contains a sort a representing standard FOL terms, and a predicate sort p satisfying the properties described in Section 2.1. The set of function symbols Σ^F is the disjoint union of the following countable sets: (i) a set Σ_f of *successor function* symbols of type $\mathsf{a} \to \mathsf{a}$, (ii) a set Σ_A of *unary predicate* symbols of type $\mathsf{a} \to \mathsf{p}$, (iii) a set Σ_S of *binary predicate* symbols of type $\mathsf{a} \times \mathsf{a} \to \mathsf{p}$, and (iv) a set Σ_c of *constant* symbols of type $\to \mathsf{a}$.

In the remainder of this section, we assume that $\Sigma = \big\{\{\mathsf{a}, \mathsf{p}\}, \{\Sigma_f \uplus \Sigma_A \uplus \Sigma_S \uplus \Sigma_c\}\big\}$ is a fixed DL signature; any reference to notions defined in Section 2.1 is implicitly over Σ. For each $f \in \Sigma_f$, the *f-successor* of a term t is the term $f(t)$; we also say that t is the *predecessor* of $f(t)$. We assume there exists a *central variable* $x \in \mathcal{X}_\mathsf{a}$, and a set of *neighbour variables* $\{z_i \mid i \in \mathbb{N}\} \subset \mathcal{X}_\mathsf{a}$. As we show next, the central variable x will represent an arbitrary element of the domain, and the neighbour variables will represent elements of the domain connected to x via binary predicates.

A *DL-a-term* is a term of the form z_i for $i \in \mathbb{N}$, x, $f(x)$ for some $f \in \Sigma_f$, or $o \in \Sigma_c$. A *DL-p-term* is a term of the form $B(z_i)$, $B(x)$, $B(f(x))$, $B(o)$, $S(z_i, x)$, $S(x, z_i)$, $S(x, x)$, $S(x, f(x))$, $S(f(x), x)$; 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 $A \approx \mathsf{true}$, written A for short, where A is a DL-p-term. A *DL-literal* is a DL-atom or an equality or inequality between DL-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, normalised $\mathcal{ALCHOIQ}^+$ axioms correspond to DL-clauses as specified in Table 1, where we assume $\mathbf{B} \subseteq \Sigma_A$, $\mathbf{S} \subseteq \Sigma_S$, and $\mathbf{I} \subseteq \Sigma_c$. Note that axioms of the form DL4 would normally be translated into first-order logic 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 atoms $B_2(z_i)$. We do not allow atoms of this form in DL-clauses because ensuring completeness in presence of these atoms would require making the calculus significantly more involved. Instead, our translation introduces fresh predicates S_{B_2} for each pair S, 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 (1) with literal $S_{B_2}(x, z_i)$.

We define an *ontology* as a set O of DL-clauses. An ontology O is $\mathcal{ALCHOIQ}^+$ if each DL-clause in O is of one of the forms given in Table 1. It is $\mathcal{ALCHIQ}^+$ if it does not contain clauses of the form DL10 or DL11. It is $\mathcal{ALCHOQ}$ if it does not contain clauses DL5, DL6, DL8, or DL9. It is $\mathcal{ALCHOI}$ if it does not contain clauses DL4 to DL6, or DL9, and if for every clause of the form DL2 we have $n = 1$. It is *Horn* if it does not contain clauses with more than one literal in the head. Finally, it is $\mathcal{ELHO}$ if it is Horn and contains only clauses of the form DL1, DL2 with $n = 1$, DL3, DL7, DL10, and DL11.

A *query clause* is a DL-clause $\Gamma_Q \to \Delta_Q$ where all atoms are of the form $B(x)$ for $B \in \Sigma_A$. Query clauses in the CB calculus for $\mathcal{SRIQ}$ [11] also allow atoms of the form $S(x, x)$ in Δ_Q; however, these can be replaced with a fresh atom $B_S(x)$ if O is extended with a DL-clause $S(x, x) \to B_S(x)$, which is of the form DL6.

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

In this section we describe the key challenges in extending the CB calculus for $\mathcal{ALCHIQ}^+$ from [11] to $\mathcal{ALCHOIQ}^+$ and we discuss informally how our calculus approaches them. In Section 3.1 we provide an overview of the calculus in [11]. In Section 3.2 we examine the problem of combining the local nature of CB reasoning with the global constraints on ontology models imposed by the use of nominals, and we sketch the intuitions underpinning our solution. Finally, in Section 3.3 we consider the difficulties caused by the simultaneous interaction between inverse roles, number restrictions, and nominals.

3.1. The Consequence-Based Calculus for $\mathcal{ALCHIQ}^+$

The calculus for $\mathcal{ALCHIQ}^+$ proposed in [11] decides whether $\mathcal{O} \models Q$ for $\mathcal{O}$ an ontology and $Q = \Gamma_Q \rightarrow \Delta_Q$ a query clause. We illustrate how the calculus works using a running example that checks whether $\mathcal{O}_1 \models Q_1$, with $\mathcal{O}_1$ the ontology from Figure 1 and Q_1 the query clause $A(x) \rightarrow D(x)$.

Following the framework introduced in [59], the rules of the calculus are applied to a *context structure*—a directed graph where each vertex v (called a *context*) can be thought of as representing a collection of domain elements in a model of the input ontology $\mathcal{O}$. A context structure assigns to each context v the following information.

- A conjunction core_v of atoms, called the *core* of v, involving only two variables x and y. The core of v determines the collection of domain elements represented by v. Concretely, for each model $\mathcal{I}$ of $\mathcal{O}$, context v represents each element c such that $\mathcal{I} \models \exists y.\mathsf{core}_v\{x \mapsto c\}$, and we refer to such elements mapped to variable x as the *instances* of core_v in $\mathcal{I}$.
- A set of constant-free *context clauses* $\mathcal{S}_v$ where the only terms allowed are x, y, and functional terms of the form $f(x)$, for $f \in \Sigma_f$. These variables have a special meaning: for each model $\mathcal{I}$ of $\mathcal{O}$, x represents instances of core_v in $\mathcal{I}$, and y represents the predecessor of x. Each context clause $\Gamma \rightarrow \Delta$ represents a logical consequence of $\mathcal{O}$ and is such that $\mathsf{core}_v \subseteq \Gamma$; for succinctness, it is written as $\Gamma' \rightarrow \Delta$, where $\Gamma = \mathsf{core}_v \wedge \Gamma'$.
- Similarly to ordered resolution calculi [8, 31], each context v is parametrised by an ordering $\succ_v$ that determines which literals and context clauses can be selected by inference rules.

Edges in a context structure are labelled with successor function symbols. If there is an edge from context v to context w, then v is a *predecessor context* of w, and w is a *successor context* of v. An f-labelled edge from a context v to a context w indicates that, in each model $\mathcal{I}$ of $\mathcal{O}$, every element represented by context v may have an f-successor represented by context w, and this successor is guaranteed to exist if $\mathsf{core}_w \neq \{\}$. A context structure may have more than one edge from v to w, each labelled with a different function symbol. A context structure may also contain self-loops.

To check whether $\mathcal{O} \models Q$, the calculus needs a context structure with a context representing (at least) all instances of Γ_Q in a model of $\mathcal{O}$. In our example for $\mathcal{O}_1$ and Q_1, we introduce a context structure with a single context v_A with $\mathsf{core}_{v_A} = \{A(x)\}$, representing all instances of $A(x)$ in a model of $\mathcal{O}_1$. Furthermore, we choose an ordering $\succ_A$ for the literals in v_A which satisfies $g(x) \succ_A f(x)$ and makes $D(x)$ smaller than other atoms, and also smaller than equalities between functional terms. The example is illustrated in Figure 2.

The calculus applies its inference rules to derive logical consequences of $\mathcal{O}$ as context clauses in the context structure, until no rule can be further applied. Given an arbitrary context v, each of the following inference rules can be applied to v.

- Rule Core adds a context clause $\top \rightarrow A$ to v for each atom $A \in \mathsf{core}_v$. For example, in Figure 2, this rule is triggered after the creation of the context structure, and it adds context clause (15) to v_A, which has core $\{A(x)\}$.
- Rule Hyper applies an ordered hyperresolution step [8] using context clauses in v as side premises, and a clause in $\mathcal{O}$ as the main premise. For instance, in Figure 2, the rule is triggered for context clause (15) and ontology clause (2) to produce context clause (16). Analogous applications of this rule to clause (15) produce context clauses (17) and (18), using ontology clauses (3) and (6), respectively. Rule Hyper may only pick literals in heads of context clauses that are maximal with respect to other literals in the head, according to the order $\succ_v$ chosen for context v. For instance, in clause (18), the Hyper rule can pick $G(x)$ but not $D(x)$, since we have $G(x) \succ_A D(x)$. Two inferences by this rule on clause (18) produce clauses (19) and (20) from ontology clauses (7) and (8). Finally, the rule can be applied to ontology clause (9) and clauses (17) and (20) to produce clause (21). Inferences by Hyper can only unify variable x in $\mathcal{O}$ with variable x in context clauses; this ensures that derived consequences preserve the syntactic form of context clauses.
- Rule Eq finds a clause in v with an equality $t \approx s$ in the head, where $t \succ_v s$, and combines it with another clause in v that has a literal A mentioning t, replacing t by s in A. For instance, in Figure 2, we can apply the Eq rule to combine clause (21) with clause (19), and the result is clause (22). A similar inference produces clause (23) from clause (21) and clause (20). This rule, together with the next two, implement a form of ordered paramodulation reasoning [7].
- Rule Ineq eliminates literals of the form $t \not\approx t$ from heads of context clauses in v. For instance, if a context clause $\top \rightarrow f(x) \not\approx f(x)$ appears in v, the rule will write $\top \rightarrow \bot$ in this context.

$$
\begin{array}{lllll}
A \sqsubseteq \exists R.B \rightsquigarrow & A(x) & \rightarrow & B(f(x)) & (2)\\
 & A(x) & \rightarrow & R(x,f(x)) & (3)\\
A \sqsubseteq \exists S.B \rightsquigarrow & A(x) & \rightarrow & B(h(x)) & (4)\\
 & A(x) & \rightarrow & S(x,h(x)) & (5)\\
A \sqsubseteq D \sqcup G \rightsquigarrow & A(x) & \rightarrow & D(x) \vee G(x) & (6)\\
G \sqsubseteq \exists R.C \rightsquigarrow & G(x) & \rightarrow & C(g(x)) & (7)\\
 & G(x) & \rightarrow & R(x,g(x)) & (8)
\end{array}
\qquad
\begin{array}{rrcll}
\top \sqsubseteq \leqslant 1 R.\top \rightsquigarrow & R(x,z_1) \wedge R(x,z_2) & \rightarrow & z_2 \approx z_1 & (9)\\
B \sqcap C \sqsubseteq F_1 \rightsquigarrow & B(x) \wedge C(x) & \rightarrow & F_1(x) & (10)\\
\exists R.F_1 \sqsubseteq D_1 \rightsquigarrow & R(z_1,x) \wedge F_1(x) & \rightarrow & D_1(z_1) & (11)\\
B \sqsubseteq F_2 \rightsquigarrow & B(x) & \rightarrow & F_2(x) & (12)\\
\exists S.F_2 \sqsubseteq D_2 \rightsquigarrow & S(z_1,x) \wedge F_2(x) & \rightarrow & D_2(z_1) & (13)\\
D_1 \sqcap D_2 \sqsubseteq D \rightsquigarrow & D_1(x) \wedge D_2(x) & \rightarrow & D(x) & (14)
\end{array}
$$

Figure 1: Ontology clauses for O_1 are represented together with their corresponding equivalent $\mathcal{ALCHIQ}$ axioms.

$$
\begin{array}{ll}
\top \rightarrow A(x) & (15)\\
\top \rightarrow B(f(x)) & (16)\\
\top \rightarrow R(x,f(x)) & (17)\\
\top \rightarrow D(x) \vee G(x) & (18)\\
\top \rightarrow D(x) \vee C(g(x)) & (19)\\
\top \rightarrow D(x) \vee R(x,g(x)) & (20)\\
\top \rightarrow D(x) \vee g(x) \approx f(x) & (21)\\
\top \rightarrow D(x) \vee C(f(x)) & (22)\\
\top \rightarrow D(x) \vee R(x,f(x)) & (23)\\
\top \rightarrow D(x) \vee D_1(x) & (24)\\
\top \rightarrow B(h(x)) & (25)\\
\top \rightarrow S(x,h(x)) & (26)\\
\top \rightarrow D_2(x) & (27)\\
\top \rightarrow D(x) & (28)
\end{array}
$$

$$\begin{tikzcd}
 & v_{R,B} \\
v_A \arrow[ur, "f"] \arrow[dr, "h"'] & \\
 & v_{S,B}
\end{tikzcd}$$

$$
\begin{array}{rll}
\top \rightarrow & B(x) & (29)\\
\top \rightarrow & R(y,x) & (30)\\
C(x) \rightarrow & C(x) & (31)\\
C(x) \rightarrow & F_1(x) & (32)\\
C(x) \rightarrow & D_1(y) & (33)
\end{array}
$$

$$
\begin{array}{ll}
\top \rightarrow B(x) & (34)\\
\top \rightarrow S(y,x) & (35)\\
\top \rightarrow F_2(x) & (36)\\
\top \rightarrow D_2(y) & (37)
\end{array}
$$

Figure 2: Context structure generated for O_1 and Q_1 using the eager expansion strategy. The core of v_A is $\{A(x)\}$, the core of $v_{R,B}$ is $\{B(x), R(y,x)\}$, and the core of $v_{S,B}$ is $\{B(x), S(y,x)\}$.

- Rule Factor "factors out" a term that appears on the left-hand side of two equalities in the head of a context clause in v; for example, $\top \rightarrow f(x) \approx y \vee f(x) \approx g(x)$. The rule replaces one of the equalities by the inequality between the terms on the right-hand side of the equalities. In the previous example, the rule will add $\top \rightarrow g(x) \not\approx y \vee f(x) \approx g(x)$ in v. The new clause is equivalent to the previous, but smaller with respect to the literal ordering. When the context structure becomes saturated, this rule ensures that all context clauses are reduced with respect to $>_v$, which is a standard requirement for completeness in paramodulation-based techniques.

- Rule Elim is used for redundancy elimination within a context. For instance, it will remove a clause $G(x) \rightarrow H(x)$ from v if a clause $\top \rightarrow H(x)$ is added to v. It also eliminates clauses with a literal $s \approx s$ or a disjunction $s \approx t \vee s \not\approx t$ in the head.

The calculus also introduces inference rules involving two neighbouring contexts in the context structure.

- Rule Succ propagates information from v to successor contexts. The rule searches for *successor triggers* in v, which are atoms in heads of context clauses that unify with atoms in the body of ontology clauses via a substitution mapping $f(x)$ to x. For these atoms, the Hyper rule cannot perform the corresponding inference, because it requires that variable x in the ontology clause must unify with variable x in context clauses, as we discussed above. For instance, in Figure 2 we have that atom $B(f(x))$ in the head of clause (16) unifies with the atom in the body of ontology clause (12), but the Hyper rule cannot be applied to derive $F(f(x))$. Instead, the Succ rule propagates successor triggers to a successor context w, where $f(x)$ is replaced by x, and so the Hyper rule can be applied to these atoms in w.

  A single application of the Succ rule to a context v selects a successor function symbol f mentioned in v and a set of successor triggers in different clauses of v that mention $f(x)$. This set is transformed via substitution $\{f(x) \mapsto x, x \mapsto y\}$ to a set K_2 of the same form used for context cores. The Succ rule must then find a *target context* w representing (at least) all instances of K_2 in a model of O, and then add an f-labelled edge from v to w. For instance, in Figure 2, the application of rule Succ identifies the set $K_2 = \{B(x), R(y,x), C(x)\}$ for term $f(x)$ in v_A from clauses (16), (17) and (22).

  The Succ rule has a parameter called the *expansion strategy* which determines how to choose the target context w. The core of w must satisfy $\mathsf{core}_w \subseteq K_2$, for otherwise w may not represent all instances of K_2 in a model of O. Furthermore, recall that if there exists an f-labelled edge from v to w and core_w is not empty, then every element represented by v *must* have an f-successor represented by w in a model of O. Hence, the core of w must also satisfy $\mathsf{core}_w \subseteq K_1$, where K_1 is the set of all atoms in K_2 that appear in Horn clauses of the form $\top \rightarrow A$ in v. In the application of Succ described above for Figure 2, we have $K_1 = \{B(x), R(y,x)\}$, from clauses (16) and (17).

$$\begin{aligned}
&\top \to A(x) && (15)\\
&\top \to B(f(x)) && (16)\\
&\top \to R(x, f(x)) && (17)\\
&\top \to D(x) \lor G(x) && (18)\\
&\top \to D(x) \lor C(g(x)) && (19)\\
&\top \to D(x) \lor R(x, g(x)) && (20)\\
&\top \to D(x) \lor g(x) \approx f(x) && (21)\\
&\top \to D(x) \lor C(f(x)) && (22)\\
&\top \to D(x) \lor R(x, f(x)) && (23)\\
&\top \to D(x) \lor D_1(x) && (24)\\
&\top \to B(h(x)) && (25)\\
&\top \to S(x, h(x)) && (26)\\
&\top \to D_2(x) && (27)\\
&\top \to D(x) && (28)
\end{aligned}$$

$$\begin{tikzcd}
v_A \arrow[r, bend left, "f"] \arrow[r, bend right, "h"'] & v_B
\end{tikzcd}$$

$$\begin{aligned}
\top \to B(x) &&& (38)\\
R(y, x) \to R(y, x) &&& (39)\\
C(x) \to C(x) &&& (40)\\
C(x) \to F_1(x) &&& (41)\\
R(y, x) \land C(x) \to D_1(y) &&& (42)\\
S(y, x) \to S(y, x) &&& (43)\\
\top \to F_2(x) &&& (44)\\
S(y, x) \to D_2(y) &&& (45)
\end{aligned}$$

Figure 3: Context structure generated for O_1 and Q_1 using the cautious expansion strategy. The core of v_A is $\{A(x)\}$, and the core of v_B is $\{B(x)\}$.

The expansion strategy takes as input the conjunction K_1 and the function symbol f, and it outputs core_w and a decision as to whether w should be created fresh or selected from those already in the context structure. Bate et al. [11] describe three examples of expansion strategies:

- The *trivial* expansion strategy always chooses a fixed context $v_\top$ with empty core as the target context. If the Succ rule is triggered and $v_\top$ is not already in the context structure, then it is created fresh. Since $\mathsf{core}_{v_\top} = \{\}$, we trivially have $\mathsf{core}_{v_\top} \subseteq K_1$ for any K_1.
- The *eager* expansion strategy chooses, for a given K_1, a fixed context v_{K_1} with $\mathsf{core}_{v_{K_1}} = K_1$. Similarly to the previous case, if the Succ rule is triggered for some K_1 and v_{K_1} is not already in the context structure, then it is created fresh. The application of Succ on v_A and f illustrated in Figure 2 uses the eager expansion strategy. Context $v_{R,B}$ with $\mathsf{core}_{v_{R,B}} = \{B(x), R(y, x)\}$ is created, since it did not exist previously.
- The *cautious* expansion strategy first checks, for a given K_1 and f, whether f occurs in a single atom of the form $B(f(x))$ in O such that $B(x) \subseteq K_1$. If this is the case, it selects a fixed context v_B with $\mathsf{core}_{v_B} = \{B(x)\}$, creating it if it does not exist already. Otherwise, it selects the context $v_\top$ defined as in the trivial strategy. This strategy is designed to mimic the $\mathcal{EL}$ calculus of [2]. Figure 3 shows an alternate version of our running example for O_1 and Q_1 where Succ is applied to f and v_A with the cautious strategy.

The choice of expansion strategy can be guided by practical considerations. Expansion strategies that generate a larger number of contexts are more goal-oriented, since they produce contexts with smaller numbers of clauses, and they make context clauses simpler; these factors help in the application of Hyper and paramodulation rules. However, strategies that generate a smaller number of contexts produce fewer repeated consequences. We refer the reader to [59] and [11] for further discussion about expansion strategies.

Once the target context w has been identified, and the corresponding f-labelled edge has been added, the Succ rule adds a clause $A \to A$ to w for each $A \in K_2 \backslash \mathsf{core}_w$. This is necessary to compute all relevant consequences for instances of K_2 in models of O. In our example from Figure 2, $K_2 \backslash \mathsf{core}_{v_{R,B}} = \{C(x)\}$, so the rule adds clause (31). Indeed, $v_{R,B}$ represents all instances of $\{B(x), R(y, x)\}$ in models of O_1, but we need to explore the consequences for those instances which also satisfy $C(x)$. By adding clause (31) to v_B, its head atom $C(x)$ becomes available for further inferences, which will produce clauses with $C(x)$ in the body capturing consequences for instances of K_2 in models of O_1. Similarly, in our example from Figure 3, we have $K_2 \backslash \mathsf{core}_{v_B} = \{R(y, x), C(x)\}$, so now the Succ rule adds clause (40) as well as clause (39) to v_B.

A subsequent application of the Succ rule to v_A can use function symbol h instead. This can happen after clauses (25) and (26) have been derived via Hyper from clause (15) and ontology clauses (4) and (5), respectively. In this case, $K_2 = K_1 = \{S(y, x), B(x)\}$. In Figure 2, the eager expansion strategy creates a new context $v_{S,B}$. In contrast, in Figure 3 the cautious expansion strategy re-uses the context v_B with core $B(x)$, and clause (43) is added because $S(y, x) \in K_2 \backslash \mathsf{core}_{v_B}$, together with an h-labelled edge from v_A to v_B.

- Rule Pred complements Succ by propagating back to v information from its successors. For this, assume that we can identify a successor function symbol f and a corresponding set K_2 in a context v as in the Succ rule. Furthermore, let w be a successor of v connected via an f-labelled edge. The Pred rule is triggered when there is a clause in w of the form $K_2 \to \Delta$, where each literal in Δ is a *predecessor trigger*, namely a literal that holds relevant information for v. Predecessor triggers can be of three kinds. The first kind consists simply of the equation $x \approx y$, showing that entities represented by w

and v are identical in every model. The second kind comprises atoms that unify with atoms in the body of ontology clauses via a substitution mapping y to x; these atoms must be propagated to v, where y becomes x, and hence the Hyper rule can be applied to them in v. The third kind consists of atoms of the form $B(y)$, for any $B \in \Sigma_A$; since the corresponding atom $B(x)$ might appear in the head of a query clause, they need to be propagated to v to ensure completeness of the calculus.

To propagate the atoms in Δ back to v, the Pred rule applies a hyperresolution step using as side premises the clauses of v that produce the atoms of K_2, and the clause $K_2 \to \Delta$ as the main premise. We illustrate an example of this in Figure 2. To reach the state where Pred can be applied, first clauses (29) and (30) are derived by the Core rule and then the Hyper rule produces clause (32) from clauses (29) and (31) and ontology clause (10), as well as clause (33) from clauses (30) and (32) and ontology clause (11). Now, the only atom in the head of clause (33) is a unary atom mentioning y, so the Pred rule is triggered using this clause as the main premise, and clause (22) as side premise, and it produces clause (24) in v_A. Notice that the elided core atoms $B(x)$ and $R(y, x)$ in the body of clause (33) are implicitly resolved with those in clauses (16) and (17), respectively. Similarly, in context $v_{S,R}$, first clauses (34) and (35) are derived by the Core rule, and then the Hyper rule produces clause (36) from clause (34) and ontology clause (12), followed by clause (37) from clauses (35) and (36) and ontology clause (13). Since the only atom in the head of clause (37) mentions y and the body of this clause is empty, the Pred rule can be applied to derive clause (27) in v_A. From here, a simple application of Hyper to clauses (24) and (27) and ontology clause (14) produces clause (28), which we will be enough to prove the query target.

Compare the execution of the calculus described in the previous paragraph with Figure 3. The derivations via Hyper of clause (41) and clause (42) are analogous to the previous case. When the Pred rule is triggered by clause (42), however, we need two side premises in v_A, namely clause (22) and clause (23), and this produces again clause clause (24) in v_A. Observe also that the inferences performed in context $v_{S,R}$ of Figure 2 are now performed in the same context v_B of Figure 3. Indeed, the Hyper rule produces clause (44) from clause (43) and ontology clause (12), followed by clause (45) from clauses (43) and (44) and ontology clause (13). Finally, the Pred rule is triggered on clause (45) with clause (26) as a side premise to derive clause (27) in v_A, and then clause (28) is obtained as before.

The context structure will become saturated after a finite number of steps if the expansion strategy introduces a finite number of contexts for a given (finite) signature, because no inference rule is applied twice. The trivial, eager, and cautious strategies all introduce a finite number of contexts for a given signature, so the calculus is terminating if either of them is used. There exist, however, expansion strategies that can produce an infinite number of contexts. For instance, consider a variant of the eager strategy which creates a new context w with $\mathsf{core}_w = K_1$ every time Succ is triggered for some K_1. Then, if we extend O_1 with clauses expressing the DL-axiom $A \sqsubseteq \exists R.A$, namely $A(x) \to A(h'(x))$ and $A(x) \to R(x, h'(x))$, the strategy will generate an infinite number of contexts in Figure 2, each with core $\{A(x), R(y, x)\}$.

Once the context structure is saturated, the *soundness* property of the calculus ensures that derived context clauses are indeed logical consequences of the ontology: if $\Gamma \to \Delta$ is a clause in v, then $O \models \mathsf{core}_v \wedge \Gamma \to \Delta$. In our example from Figure 2, we deduce $O_1 \models Q_1$ from clause (28). Conversely, the *completeness* property ensures that if $O \models Q$, then each properly initialised context v will contain clause Q up to redundancy. In particular, the initialisation requirements for completeness are: *(i)* having clauses of the form $A \to A$ up to redundancy in v for each $A \in \Gamma_Q \backslash \mathsf{core}_v$; and *(ii)* ensuring that the order $\succ_v$ does not block relevant inferences in v, by making the atoms in Δ_Q as small as possible, analogously to the *answer literal* technique in [21]. Therefore, a typical method for CB reasoning will introduce a context structure v_Q with $\mathsf{core}_{v_Q} = \Gamma_Q$ and an order $\succ_{v_Q}$ making smallest the atoms in Δ_Q. Then, $\top \to \Delta_Q$ will be contained up to redundancy in v_Q if and only if $O \models Q$.

This calculus is worst-case optimal for $\mathcal{ALCHIQ}^+$ and runs in polynomial time on $\mathcal{ELH}$ ontologies. 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 O. *Goal-oriented* behaviour results from initialising the context structure with contexts representing elements of a model of O that disprove Q, adding (or reusing) contexts only on demand, and applying resolution rules only to clauses in the same context. This behaviour imitates forward chaining as used in (hyper-)tableau-based calculi, which typically introduce a fresh individual c, define a model fragment which satisfies atom $A(c)$ and literal $\neg D(c)$, and then resolve $A(c)$ with ontology clauses in order to generate new ground facts.

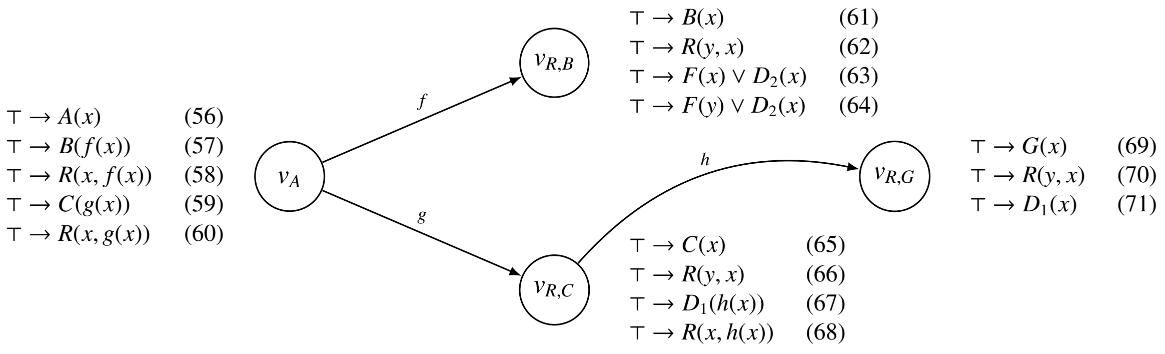
3.2. Reconciling Nominals with Local Reasoning

Reasoning in CB calculi is essentially *local*. On the one hand, contexts exchange information only with their neighbours; on the other hand, context clauses mention only terms x, $f(x)$, and y, where the latter variable appears only in clauses with body atoms of the form $S(y, x)$ or $S(x, y)$. Breaking up inferences into local clauses has important benefits.

1. Similar resolution inferences can be combined into a single CB inference. For instance, in our example from Section 3.1, individual resolution steps can be applied to clause (12) and each of clauses (2) and (4) to yield clauses $A(x) \to F_2(f(x))$ and $A(x) \to F_2(h(x))$, respectively. In contrast, in Figure 3, these two inferences are carried out simultaneously in context v_B, when rule Hyper is applied to clause (38) and ontology clause (12) to derive clause (44).

$$\begin{array}{lclr@{\qquad}lclr}
A \sqsubseteq \exists R.B & \rightsquigarrow & A(x) \rightarrow B(f(x)) & (46) & C \sqsubseteq \exists R.G & \rightsquigarrow & C(x) \rightarrow G(h(x)) & (51)\\
 & & A(x) \rightarrow R(x, f(x)) & (47) & & & C(x) \rightarrow R(x, h(x)) & (52)\\
B \sqsubseteq F \sqcup D_2 & \rightsquigarrow & B(x) \rightarrow F(x) \vee D_2(x) & (48) & G \sqsubseteq D_1 & \rightsquigarrow & G(x) \rightarrow D_1(x) & (53)\\
A \sqsubseteq \exists R.C & \rightsquigarrow & A(x) \rightarrow C(g(x)) & (49) & D_1 \sqcap D_2 \sqsubseteq \bot & \rightsquigarrow & D_1(x) \wedge D_2(x) \rightarrow \bot & (54)\\
 & & A(x) \rightarrow R(x, g(x)) & (50) & \exists R.F \sqsubseteq F & \rightsquigarrow & R(z_1, x) \wedge F(x) \rightarrow F(z_1) & (55)
\end{array}$$

Figure 4: Ontology clauses for O_2 are represented together with their corresponding equivalent $\mathcal{ALCHIQ}$ axioms.



$\top \rightarrow A(x)$ (56)
$\top \rightarrow B(f(x))$ (57)
$\top \rightarrow R(x, f(x))$ (58)
$\top \rightarrow C(g(x))$ (59)
$\top \rightarrow R(x, g(x))$ (60)
v_A
f
g
$v_{R,B}$
$\top \rightarrow B(x)$ (61)
$\top \rightarrow R(y, x)$ (62)
$\top \rightarrow F(x) \vee D_2(x)$ (63)
$\top \rightarrow F(y) \vee D_2(x)$ (64)
h
$v_{R,G}$
$\top \rightarrow G(x)$ (69)
$\top \rightarrow R(y, x)$ (70)
$\top \rightarrow D_1(x)$ (71)
$v_{R,C}$
$\top \rightarrow C(x)$ (65)
$\top \rightarrow R(y, x)$ (66)
$\top \rightarrow D_1(h(x))$ (67)
$\top \rightarrow R(x, h(x))$ (68)

Figure 5: Context structure generated for O_2 and query $A(x) \rightarrow F(x)$ using the eager expansion strategy. The core of v_A is $\{A(x)\}$, the core of $v_{R,B}$ is $\{B(x), R(y, x)\}$, the core of $v_{R,C}$ is $\{C(x), R(y, x)\}$, and the core of $v_{R,G}$ is $\{G(x), R(y, x)\}$.

2. Context structures used for checking entailment of a query clause Q can be reused for checking entailment of other query clauses. The context structures in Figures 2 and 3 were created while checking $O_1 \models A(x) \rightarrow F(x)$, but they can be reused, for instance, to check whether $O_1 \models B(x) \rightarrow F_2(x)$.

3. It ensures that the number of clauses that can be derived within each context is bounded, which simplifies both termination and complexity arguments.

The presence of nominals in the ontology, however, requires some form of non-local reasoning. To illustrate this, we use a new example where we check whether $O_2 \models A(x) \rightarrow F(x)$, for O_2 the ontology in Figure 4, using the $\mathcal{ALCHIQ}^+$ calculus outlined in Section 3.1. We first introduce a single context v_A with $\mathsf{core}_{v_A} = \{A(x)\}$, and then we apply the inference rules of the calculus with the eager expansion strategy, as in the example in Figure 2. This results in the context structure shown in Figure 5.

Let O'_2 be the ontology extending O_2 with the following clauses (which we also provide in the form of $\mathcal{ALCHOIQ}^+$ axioms):

$$D_1 \sqsubseteq \{o\} \rightsquigarrow D_1(x) \rightarrow x \approx o \quad (72) \qquad D_2 \sqsubseteq \{o\} \rightsquigarrow D_2(x) \rightarrow x \approx o \quad (73)$$

It is easy to verify that $O'_2 \models A(x) \rightarrow F(x)$. Indeed, if $\mathcal{I}$ is a model of O'_2 and c is an arbitrary instance of $A(x)$ in $\mathcal{I}$, then $f(c)$ is an instance of the core $\{R(y, x), B(x)\}$ and $h(g(c))$ is an instance of $\{R(y, x), G(x)\}$ in $\mathcal{I}$. By soundness of the calculus and the fact that $O_2 \subseteq O'_2$, clause (63) guarantees that every instance of $\{R(y, x), B(x)\}$ in $\mathcal{I}$ is an instance of either $F(x)$ or $D_2(x)$. Similarly, clause (71) ensures that every instance of $\{R(y, x), G(x)\}$ in $\mathcal{I}$ is an instance of $D_1(x)$. However, ontology clauses (54), (72) and (73) together imply that $D_1(x)$ and $D_2(x)$ cannot be both realised in $\mathcal{I}$. Therefore, $f(c)$ must be an instance of $F(x)$ in $\mathcal{I}$. By clause (55), this implies $\mathcal{I} \models F(c)$, and therefore we conclude $\mathcal{I} \models A(x) \rightarrow F(x)$.

This example suggests that a complete CB procedure for $\mathcal{ALCHOIQ}^+$ should perform inferences that involve clauses (63) and (71), as well as ontology clauses (54), (72) and (73). The problem lies in the fact that clauses (63) and (71) belong to contexts that are not adjacent, which seems to require some form of non-local reasoning.

Our solution addresses this issue in a way that preserves the aforementioned benefits of local reasoning in previous CB calculi. The key idea is to allow constants to appear in context clauses, and then make sure that relevant literals with constants, including ground atoms, can be freely propagated from context to context. We also introduce new inference rules that perform ground atom resolution between clauses in the same context. All these new inference rules still involve at most two contexts, and non-ground terms in context clauses preserve the same form as in the $\mathcal{ALCHIQ}^+$ calculus.

For example, our calculus can prove $O'_2 \models A(x) \rightarrow F(x)$ as follows. First, it generates the context structure in Figure 5 in the same way as the $\mathcal{ALCHIQ}^+$ calculus. Then, it uses the Hyper rule to derive $\top \rightarrow x \approx o$ in context $v_{R,G}$ from clause (71) and ontology clause (73). This clause is propagated to $v_{R,C}$ as $\top \rightarrow h(x) \approx o$, where we can perform a paramodulation with clause (67) to derive a clause $\top \rightarrow D_1(o)$ in $v_{R,C}$. This clause shows that every model of O'_2 that realises $\{R(y, x), C(x)\}$ must satisfy $D_1(o)$. Then, the inference rules of our calculus allow us to propagate this clause to v_A. Clause $\top \rightarrow D_1(o)$ in v_A shows that every model of O'_2 that realises $\{A(x)\}$ must satisfy $D_1(o)$. Using a similar sequence of inferences, the calculus derives a clause $\top \rightarrow F(x) \vee D_2(o)$ in v_A, representing that if some instance of $A(x)$ in a model of O'_2 is not also an instance of $F(x)$, then the model satisfies $D_2(o)$. Finally, the calculus implements a ground resolution step with ontology clause (54) and clauses $\top \rightarrow D_1(o)$ and $\top \rightarrow F(x) \vee D_2(o)$ in v_A, which produces $\top \rightarrow F(x)$ in v_A. This suffices to prove the target query.

Our calculus also introduces a *root context* to carry out inferences involving ground atoms and ontology clauses, which can then be propagated to other contexts if necessary. For instance, if we add a clause $\top \rightarrow A(o)$ to $\mathcal{O}_2'$, the calculus will initialise the root context and copy this clause in it. Next, using a procedure analogous to our previous example, the calculus will derive a clause $\top \rightarrow F(o)$ in the root context, which then can be propagated to other contexts.

3.3. Interaction of $\mathcal{I}$, $\mathcal{Q}$, and $\mathcal{O}$.

Extending $\mathcal{ALCHIQ}$ with nominals presents well-known additional difficulties for reasoning; these are caused by the interaction between nominals, inverse roles, and number restrictions [44, 26]. The following example shows how this interaction compromises the forest model property of DLs. Consider the ontology $\mathcal{O}_2''$ extending $\mathcal{O}_2$ with the following clauses, which use inverse roles (clauses (79) and (80)), number restrictions (clause (81)), and nominals (clause (78)):

$$\begin{array}{lll} D_1 \sqsubseteq \exists S.D \rightsquigarrow & D_1(x) \rightarrow D(f_1(x)) & (74) \\ & D_1(x) \rightarrow S(x, f_1(x)) & (75) \\ D_2 \sqsubseteq \exists S.D \rightsquigarrow & D_2(x) \rightarrow D(f_2(x)) & (76) \\ & D_2(x) \rightarrow S(x, f_2(x)) & (77) \end{array} \qquad \begin{array}{llr} D \sqsubseteq \{o\} & \rightsquigarrow & D(x) \rightarrow x \approx o \quad (78) \\ S \sqsubseteq U^- & \rightsquigarrow & S(z,x) \rightarrow U(x,z) \quad (79) \\ U \sqsubseteq S^- & \rightsquigarrow & U(z,x) \rightarrow S(x,z) \quad (80) \\ \top \sqsubseteq \leqslant 1\, U.\top & \rightsquigarrow & U(x,z_1) \wedge U(x,z_2) \rightarrow z_1 \approx z_2 \quad (81) \end{array}$$

In proving $\mathcal{O}_2'' \models A(x) \rightarrow F(x)$, clauses (74) through (81) interact in a non-trivial way. Let $\mathcal{I}$ be an arbitrary model of $\mathcal{O}_2''$, and let c be an arbitrary instance of $D_1(x)$. By clause (74), $f_1(c)$ is an instance of $D(x)$ in $\mathcal{I}$. However, by clause (78), $f_1(c)$ must be equal to o. Furthermore, by clause (75), we have $\mathcal{I} \models S(c, o)$, and therefore by clause (79), $\mathcal{I} \models U(o, c)$. This implies, together with clause (81), that $D_1(x)$ has at most one instance in $\mathcal{I}$. A symmetric argument applies to $D_2(x)$. Moreover, if c_1 is an instance of $D_1(x)$, and c_2 is an instance of $D_2(x)$, then c_1 and c_2 must be identical. Hence, the behaviour of clauses (74) through (81) reproduces that of clauses (72) and (73) in our previous example.

Atoms $D_1(x)$ and $D_2(x)$ function as nominals in $\mathcal{O}_2''$, except that they need not be realised in each model of the ontology. As a result, the simple form of forest model property enjoyed by ExpTime DLs lacking either inverse roles, number restrictions or nominals no longer holds. In such DLs, satisfiable ontologies always have a model consisting of a "cloud" of arbitrarily interconnected elements representing constants, a collection of tree-shaped fragments rooted in elements of this cloud, and role links between these two parts of the model. The proofs of correctness and completeness for tableau, resolution, and CB algorithms rely on this property. In our example, however, an instance c of $D_1(x)$ or $D_2(x)$ may be connected to any other element in the model, including those that do not represent constants and lie in different tree-shaped fragments.

The tableau algorithm in [26] and the resolution calculus in [38] address this issue in a similar way: they introduce rules that name anonymous domain elements connected to a nominal o via a role R whenever the inverse of 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, whereas the resolution calculus introduces n constants, and it may later derive equalities between them. Fresh nominals may themselves trigger the introduction of new nominals; both calculi must ensure that the generation of nominals is bounded to guarantee termination.

Our CB calculus follows a similar approach to the resolution calculus in [38]. It introduces a rule Nom that generates new constants to represent anonymous elements behaving like nominals. This rule first searches for two atoms of the form $B(o)$ and $S(o, x)$ in heads of two distinct clauses in the same context such that there is an ontology clause of the form DL4 in Table 1 with $B = B_1$ and $S = S_{B_2}$. The rule then combines these two clauses and replaces atoms $B(o)$ and $S(o, x)$ with an equality of the form $x \approx o_{S^1} \vee \ldots, \vee x \approx o_{S^n}$, where each o_{S^i} is a constant not mentioned in the ontology, and $n + 1$ is the number of neighbour variables in the corresponding ontology clause. Intuitively, o_{S^i} is interpreted as a domain element t satisfying $S(o, t)$ in models where such t exists, and it is interpreted arbitrarily in all other models. The string S^i annotating the constant, which we call a *nominal label*, indicates that o_{S^i} is a successor of o via S generated by the Nom rule. The rule introduces n constants of this form, distinguished by the indices i, since no more than n successors of o via S may exist in a model. The precise interpretation of each constant o_{S^i} is not fixed, and in a model of the ontology some of these constants may be equal. Rule Nom can also be applied to atoms mentioning a constant o_ρ introduced in a previous inference, with ρ a nominal label. In this case, the nominal labels of the newly introduced constants will be of the form $\rho \cdot S^i$, where the suffix S^i is appended to ρ to indicate the relevant information associated to the latest application of the rule. To ensure termination, our calculus imposes a *depth limit* on the length of nominal labels; furthermore, we establish lower bounds on such depth limit ensuring completeness.

To illustrate how the rule Nom works, we provide a rough sketch of how our calculus can prove $\mathcal{O}_2'' \models A(x) \rightarrow F(x)$ using the eager strategy. First, we generate again the context structure in Figure 5. Then, following a chain of inferences similar to those used in Section 3.2 to derive $\top \rightarrow D_1(o)$ in $v_{R,C}$, our calculus derives a clause of the form $\top \rightarrow U(o, x)$ in context $v_{R,G}$. This clause represents that, in every model of $\mathcal{O}_2''$ that realises $\{R(y, x), G(x)\}$, the interpretation of o is connected by role U to each instance of $\{R(y,x), G(x)\}$. In such cases, by clause (81), there can be only one instance of $\{R(y,x), G(x)\}$. To capture this nominal-like behaviour, our calculus uses the Nom rule to introduce a clause $\top \to x \approx o_{U^1}$ in $v_{R,G}$. Constant o_{U^1} represents the only successor of o by U, so the context clause claims that all instances of $\{R(y,x), G(x)\}$ in a model of $\mathcal{O}_2''$ must be identical to o_{U^1}. The calculus then uses this clause in the same way as clause $\top \to x \approx o$ in $v_{R,G}$ in the example from Section 3.2 to derive $\top \to D_1(o_{U^1})$ in context v_A. A similar chain of inferences yields clause $\top \to F(x) \lor D_2(o_{U^1})$ in v_A, and then the rules implementing ground resolution derive $\top \to F(x)$, as in Section 3.2.

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

In this section we formalise our calculus for $\mathcal{ALCHOIQ}^+$, which we have informally described in Section 3. Our calculus is designed such that, in the absence of nominals, it behaves identically to the calculus for $\mathcal{ALCHIQ}^+$ from [11].

In Section 4.1 we introduce the notion of context structure and define the inference rules of the calculus. In Section 4.2 we establish its soundness and completeness. In Section 4.3 we define a variant of the calculus designed for Horn ontologies with nominals, and finally in Section 4.4 we illustrate the calculus on two fully worked out examples. We fix in the remainder of this section an arbitrary $\mathcal{ALCHOIQ}^+$ ontology $\mathcal{O}$ over a DL signature Σ.

4.1. Context Structures and Inference Rules

We start the description of our calculus by defining the notion of a *nominal label* attached to a constant.

Definition 1 (Nominal Labels)**.** *The set of* nominal labels *consists of the empty string ϵ and each string ρ of the form $S_1^{i_1} \cdot S_2^{i_2} \cdot \ldots \cdot S_n^{i_n}$, with $\{n, i_1, \ldots, i_n\} \subseteq \mathbb{N}$, and $S_k \in \Sigma_S$ for each $1 \leq k \leq n$.*[1] *The length of a nominal label ρ is represented as $|\rho|$. We define Σ_u as the subset of all constants of the form o_ρ, where $o \in \Sigma_c$ and ρ is a nominal label.*

Observe that each constant o in $\mathcal{O}$ is in Σ_u since it can be written as o_ϵ. We distinguish between *original constants* of the form o_ϵ and *auxiliary constants* of the form o_ρ with $\rho \neq \epsilon$. The depth of an auxiliary constant o_ρ is defined as $|\rho|$.

We next define the notions of *context term*, *context atom*, and *context clause*, which extend those in [59, 11] by allowing for the occurrence of constants.

Definition 2 (Context terms, atoms and clauses)**.** *A* context 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 p-term *is a term of one of the following forms:*

- *$B(t)$, $S(x,t)$, or $S(t,x)$, with $B \in \Sigma_A$, $S \in \Sigma_S$, and t a context a-term of the form x, y, $f(x)$, or u, with $f \in \Sigma_f$ and $u \in \Sigma_u$; or*
- *$B(t)$, $S(u,t)$, 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 $f \in \Sigma_f$.*

A context atom *is an equality of the form $A \approx \mathsf{true}$, where A is a context p-term. A* context literal *is either a context atom, or an expression of the form $l \bowtie r$, where l and r are a-terms and $\bowtie \in \{\approx, \not\approx\}$.*

Finally, 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 $u_1, u_2 \in \Sigma_u$.*

Let $\Sigma_f^{\mathcal{O}}$, $\Sigma_A^{\mathcal{O}}$, $\Sigma_S^{\mathcal{O}}$ be the subsets of Σ_f, Σ_A, and Σ_S containing all elements mentioned in $\mathcal{O}$ for each respective set. Furthermore, let $\Sigma_u^{\mathcal{O}}$ be the subset of Σ_u containing exactly every constant o_ρ such that o is in $\mathcal{O}$ and all binary predicates in ρ are in $\mathcal{O}$. We are now ready to introduce context structures. Similarly to [11], context structures define a *context order* for each context, which is used to compare terms and literals. Unlike previous CB calculi, however, our calculus introduces a distinguished *root context*, and allows for equalities between constants to occur in the body of context clauses.

Definition 3 (Context Structure)**.** *A* context structure *for $\mathcal{O}$ is a tuple $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \theta \rangle$, where:*

- *$\mathcal{V}$ is a finite set of* contexts *containing a distinguished* root context *v_r;*
- *$\mathcal{E}$ is a subset of $\mathcal{V} \times \mathcal{V} \times (\Sigma_f^{\mathcal{O}} \cup \Sigma_u^{\mathcal{O}})$;*
- *core is a function that maps the root context v_r to the empty set and each other context in $\mathcal{V}$ to a (possibly empty) set of context atoms of the form $B(x)$, $S(x,y)$, $S(y,x)$, or $S(x,x)$, for $B \in \Sigma_A^{\mathcal{O}}$ and $S \in \Sigma_S^{\mathcal{O}}$;*
- *$\mathcal{S}$ is a function that maps v_r to a set $\mathcal{S}_{v_r}$ of context clauses in the signature of $\mathcal{O}$ that do not contain variable x, and each $v \neq v_r$ to a set of clauses in the signature of $\mathcal{O}$ that do not contain terms of the form $f(u)$ for $f \in \Sigma_f^{\mathcal{O}}$ and $u \in \Sigma_u^{\mathcal{O}}$; and*
- *θ is a function that maps each context $v \in \mathcal{V}$ to a strict order $>_v$ on terms, which we call a* context order.

[1]Please note that the numbers $i_1, \ldots, i_n$ are simply labelling indices, so they do not represent exponentiation of the symbols $S_1, \ldots, S_n$, respectively.

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

Table 2: Single-context inference rules from our calculus for $\mathcal{O}$ and an arbitrary context $v \in \mathcal{V}$. To avoid clutter, we have omitted the precondition $\Gamma \rightarrow \Delta \hat{\notin} \mathcal{S}_v$ from each inference rule except Elim, where $\Gamma \rightarrow \Delta$ is the conclusion in the inference rule.

We next introduce the inference rules of our calculus, starting with the rules that can be independently applied to each context. The *single-context* rules of our calculus are given in Table 2.

Let v be an arbitrary, fixed context of $\mathcal{V}$. Rules Core, Hyper, Eq, Ineq, Factor, and Elim are analogous to the corresponding rules in the $\mathcal{ALCHIQ}^{+}$ calculus [11]. 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 $\Sigma_u^{\mathcal{O}}$. This stands in contrast to 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 as in [11]; however, in our calculus the Eq rule can be used to replace x with a constant in $\Sigma_u^{\mathcal{O}}$ or vice-versa. Rule Join implements a simple ground resolution step between two clauses in the same context. 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 3.3. The application of the Nom rule is constrained by the *depth limit* Λ—a parameter of the calculus which is used to ensure that no auxiliary constants of depth greater than Λ are introduced in the context structure. The existence of such limit on the depth of auxiliary constants is crucial to ensure termination.

Next, we will define *multi-context inference rules*, which exchange information between contexts. As in [11], we first 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 $\Sigma_f^{\mathcal{O}}$. 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 [11] to include ground atoms, equalities between elements of $\Sigma_u^{\mathcal{O}}$, and equalities of the form $x \approx u$ and $y \approx u$ for any $u \in \Sigma_u^{\mathcal{O}}$. 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 $\Sigma_u^{\mathcal{O}}$. Conversely, the set of *root predecessor triggers* contains literals that trigger propagation of information from v_r to its non-root predecessors.

Definition 4 (Triggers)**.** *The set of* successor triggers $\mathsf{Su}(\mathcal{O})$ *for $\mathcal{O}$ consists of:*

1. *atoms $B(x)$ and $\{B(u) \mid u \in \Sigma_u^{\mathcal{O}}\}$ 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 $S(x, z_i)$ 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 $\{u_1, u_2\} \subseteq \Sigma_u^O$ and predicate S mentioned in the body of a clause in O; and*

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

The set of predecessor triggers $\mathsf{Pr}(O)$ *for O consists of:*

1. *atoms $B(y)$ and $\{B(u) \mid u \in \Sigma_u^O\}$ for each unary predicate $B \in \Sigma_A^O$;*

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 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 O;*

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

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

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 [59, 11].

Definition 5. *An expansion strategy* strat *for O is a function which takes as input a triple $(f, K_1, \mathcal{D})$, where $f \in \Sigma_f^O$, $K_1 \subseteq \mathsf{Su}(O)$, and $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \theta \rangle$ is a context structure for O, and returns a triple $(v, \mathsf{core}, \succ)$ where $\mathsf{core} \subseteq K_1$, $\succ$ is a context order, 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* [59] does not introduce new contexts, and always reuses the *trivial context* $v_\top$ with an empty core; it is defined as $\mathsf{triv}(f, K_1, \mathcal{D}) = \langle v_\top, \emptyset, \succ_\top \rangle$, where $\succ_\top$ is an arbitrary context order. The *eager strategy* [59] never reuses contexts, and introduces instead a fresh context for every K_1; it is defined as $\mathsf{eager}(f, K_1, \mathcal{D}) = \langle v_{K_1}, K_1, \succ_{K_1} \rangle$, where $\succ_{K_1}$ is an arbitrary context order, and both v_{K_1} and $\succ_{K_1}$ are uniquely determined for each K_1. We refer the reader to Section 3 and to [59] and [11] for additional examples of expansion strategies and a discussion of their comparative strengths and weaknesses.

Table 3 shows the multi-context inference rules of our calculus. Rules Succ and Pred are analogous to the corresponding rules in [11]. As in the case of Hyper, these rules have been adapted to ensure that the correct substitution is used whenever they propagate information from the root context to a non-root successor, or from a non-root successor to the root context (acting as predecessor). 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 added to the body of the corresponding clause in the predecessor. This allows us to propagate ground atoms between contexts, which we need in order to simulate non-local reasoning, as discussed in Section 3.2. 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 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 general contexts, and it works analogously to Pred.

A (possibly infinite) sequence of rule applications to a context structure $\mathcal{D}_0$ for O defines a *derivation* $(\mathcal{D}_0, \mathcal{D}_1, \dots)$ with respect to O where, for each $i \geq 0$, $\mathcal{D}_{i+1}$ is a context structure for O obtained by applying a rule from Tables 2 and 3 to $\mathcal{D}_i$. A context structure $\mathcal{D}$ for O *is derivable from* $\mathcal{D}_0$ if there exists a finite derivation $(\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D})$.

4.2. Soundness and Completeness

In this section we state the soundness and completeness claims for our calculus. Proofs of these claims are given in Appendix B and Appendix D, respectively.

As in previous CB calculi, we define a notion of soundness for a context structure ensuring that derived clauses correspond to logical consequences of O.

Definition 6 (Sound Context Structure)**.** *Given a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \theta \rangle$ for O, let $C_\mathcal{D}$ be the set containing the following clause for each ontology clause of the form DL4 from Table 1 in O and each $o_\rho \in \Sigma_u$ occurring in $\mathcal{D}$:*

$$B_1(o_\rho) \wedge S_{B_2}(o_\rho, x) \to \bigvee_{i=1}^{n} x \approx o_{\rho \cdot S^i_{B_2}}. \tag{82}$$

We say that $\mathcal{D}$ is sound *if the following conditions hold:*

Succ	If	$\Gamma \rightarrow \Delta \vee A \in \mathcal{S}_v$ where $\Delta \not\succ_v A$,
		if $v \neq v_r$ then A contains $f(x)$ and otherwise A contains $f(u)$ for $u \in \Sigma_u^{\mathcal{O}}$,
		there is no $\langle v, w, f \rangle \in \mathcal{E}$ such that $A' \rightarrow A' \mathrel{\hat{\in}} \mathcal{S}_w$ 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 $A' \rightarrow A'$ 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 $\sigma = \{y \mapsto u, x \mapsto f(u)\}$,
		$K_1 = \{A' \in \mathsf{Su}(\mathcal{O}) \mid A'$ mentions $x\sigma$ and $\top \rightarrow A'\sigma \in \mathcal{S}_v\}$,
		$K_2 = \{A' \in \mathsf{Su}(\mathcal{O}) \mid \Gamma' \rightarrow \Delta' \vee A'\sigma \in \mathcal{S}_v\}$ and $\Delta' \not\succ_v A'\sigma$.
Pred	If	$v \neq v_r$, and there is $\langle w, v, f \rangle \in \mathcal{E}$,
		$\bigwedge_{i=1}^m A_i \wedge \bigwedge_{i=1}^n C_i \rightarrow \bigvee_{i=1}^k L_i \in \mathcal{S}_v$, where if $w \neq v_r$ then $L_i \in \mathsf{Pr}(\mathcal{O})$ for each
		non-ground L_i, and if $w = v_r$ then for each non-ground L_i
		either $L_i \in \mathsf{Pr}(\mathcal{O})$ or it is of the form $S(x, u)$ or $S(u, x)$,
		A_i is ground for $1 \leq i \leq m$,
		$\Gamma_i \rightarrow \Delta_i \vee C_i\sigma \in \mathcal{S}_w$ with $\Delta_i \not\succ_w C_i\sigma$ for $1 \leq i \leq n'$,
	then	add $\bigwedge_{i=1}^{n'} \Gamma_i \wedge \bigwedge_{i=1}^m A_i \rightarrow \bigvee_{i=1}^{n'} \Delta_i \vee \bigvee_{i=1}^k L_i\sigma$ 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 $u \in \Sigma_u^{\mathcal{O}}$, and $\{C_{n+1}, \ldots, C_{n'}\} = \mathsf{core}_v$.
r-Succ	If	$v \neq v_r$,
		$\Gamma \rightarrow \Delta \vee A\sigma \in \mathcal{S}_v$ where $\Delta \not\succ_v A\sigma$ with $\sigma = \{y \mapsto x\}$, $A \in \mathsf{Su}^r(\mathcal{O})$ and contains $u \in \Sigma_u^{\mathcal{O}}$,
		there is no $\langle v, v_r, u \rangle \in \mathcal{E}$ such that $A \rightarrow A \mathrel{\hat{\in}} \mathcal{S}_{v_r}$,
	then	add $\langle v, v_r, u \rangle$ to $\mathcal{E}$,
		add $A \rightarrow A$ to $\mathcal{S}_{v_r}$.
r-Pred	If	$\bigwedge_{i=1}^n C_i \rightarrow \bigvee_{i=1}^k L_i \in \mathcal{S}_{v_r}$,
		$L_i \in \mathsf{Pr}^r(\mathcal{O})$ for each nonground L_i,
		$C_i \in \mathsf{Su}^r(\mathcal{O})$, 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
		$\Gamma_i \rightarrow \Delta_i \vee C_i\sigma \in \mathcal{S}_v$ where $\Delta_i \not\succ_v C_i\sigma$ and $\sigma(y) = x$,
	then	add $\bigwedge_{i=1}^n \Gamma_i \wedge \bigwedge_{i=1}^m A_i \rightarrow \bigvee_{i=1}^n \Delta_i \vee \bigvee_{i=1}^k L_i\sigma$ to $\mathcal{S}_v$.

Table 3: Multi-context inference rules from our calculus for $\mathcal{O}$. To avoid clutter, we have omitted the precondition $\Gamma \rightarrow \Delta \mathrel{\hat{\notin}} \mathcal{S}_v$ from Pred, and r-Pred, where $\Gamma \rightarrow \Delta$ is the conclusion in each rule.

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

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

Clauses of the form (82) ensure that auxiliary constants have the intended meaning described in Section 3.3. Furthermore, whenever core_w is empty, the expression $\mathsf{core}_v \rightarrow \mathsf{core}_w\{x \mapsto f(x), y \mapsto x\}$ is equal to $\mathsf{core}_v \rightarrow \top$, 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 define successors only for specific constants mentioned in v_r, and not necessarily for *every* individual represented by v_r.

Theorem 1 (Soundness). *Given a context structure $\mathcal{D}$ for $\mathcal{O}$ which is sound and an expansion strategy for $\mathcal{O}$, the application of a rule from Table 2 or Table 3 to $\mathcal{D}$ yields a context structure for $\mathcal{O}$ that is also sound.*

To ensure completeness (i.e., that all relevant inferences are derived), we need to impose certain *admissibility* conditions on context orders, which extend those presented in [11]. For instance, all context orders in a context structure must agree on how constants are ordered, and constants with longer nominal labels should be greater 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 constants; thus, we require that the order must not compare variables x and y with any $u \in \Sigma_u$. Finally, as in [11], Condition 5 in the following definition ensures that all literals in the head of a context clause in v which could be propagated to a predecessor context should not block other literals in the same clause that may be relevant to inferences in v.

Definition 7 (Admissible Context Order). *Let $\gtrdot$ be a total order defined on $\Sigma_f \cup \Sigma_u$. We say that $\gtrdot$ is* a-admissible *if $o_\rho \gtrdot a_{\rho'}$ whenever $|\rho| > |\rho'|$, or both $|\rho| = |\rho'|$ and $o \gtrdot a$, for any $o_\rho, a_{\rho'} \in \Sigma_u$, and $f \gtrdot u$ for each $f \in \Sigma_f$ and $u \in \Sigma_u$. A context order $\succ$ is* admissible *with respect to $\gtrdot$ if it satisfies the following properties.*

1. *For each context term $s \neq$* true*, we have $s \succ$* true.

2. *For each context* p-term *A with $A \neq$* 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 $\gg$ on context* a*-terms; furthermore, it contains $x \succ y$, $f(x) \succ g(x)$ for each $f, g \in \Sigma_f$ with $f \gg g$, and $f(x) \succ a$ for every $f \in \Sigma_f$ and $u \in \Sigma_u$. 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.*

5. *If $A \succ s$ for some function-free context* p*-term A mentioning variable y or a constant in Σ_u then, either $s \in \{\mathsf{true}, x, y\} \cup \Sigma_u$, or s is obtained by replacing context* a*-terms in A with smaller* a*-terms relative to $\gg$.*

The context order $\succ$ implemented in our system Sequoia is obtained as follows. Given an arbitrary a-admissible total order $\gg$ on $\Sigma_f \cup \Sigma_u$. order $\succ$ is constructed as the the minimal context order containing the lexicographic path order induced by $\gg$ on context a-terms as well as all of the following comparisons:

1. $x \succ y$.
2. $f(x) \succ g(x)$ for each pair $f, g \in \Sigma_f$ with $f \gg g$.
3. $f(x) \succ s$ for each $f \in \Sigma_f$ and $s \in \Sigma_u \cup \{x, y\}$.
4. $s \succ \mathsf{true}$ for every context term $s \neq \mathsf{true}$.
5. $A \succ x$ and $A \succ y$ for every context p-term A.
6. $A \succ s$ for each context p-term A, a-term s, and proper position p such that $A|_p \succ s$.
7. $A_1 \succ A_2$ for each pair of distinct p-terms A_1 and A_2 such that A_2 is obtained from A_1 by replacing context a-terms with smaller a-terms relative to $\succ$.

In Appendix A we prove that this context order is admissible with respect to $\gg$.

We are ready to formulate the completeness claim. Intuitively, our calculus ensures that, if the context structure is suitably initialised, and the parameter Λ controlling the depth of auxiliary constants generated by the Nom rule is large enough, then any saturated context structure derivable from the initial one, where all context orders are admissible with respect to a single a-admissible order $\gg$ on function symbols and constants, will contain a well-defined set of query clauses entailed by the ontology.

Theorem 2 (Completeness)**.** *Let $\mathcal{D}$ be a context structure for $\mathcal{O}$ satisfying the following properties:*

- *it is derivable from a sound context structure for $\mathcal{O}$ that mentions no auxiliary constants;*
- *there exists an* a*-admissible total order $\gg$ on $\Sigma_f \cup \Sigma_u$ such that every context is assigned a context order admissible with respect to $\gg$; and*
- *no rule in Table 2 or Table 3 can be applied.*

Assume that the parameter Λ used in the Nom *rule satisfies $\Lambda \geq 2^{\tau_{\mathsf{Su}}} \cdot 2^{\tau_{\mathsf{Pr}}} \cdot \omega$, with τ_{Su} the number of atoms in $\mathsf{Su}(\mathcal{O})$ of the form $B(x)$, $S(y, x)$ or $S(x, y)$, τ_{Pr} the number of atoms in $\mathsf{Pr}(\mathcal{O})$ of the form $B(y)$, $S(y, x)$ or $S(x, y)$, and ω the number of contexts in $\mathcal{D}$.*

Then, $\Gamma_Q \to \Delta_Q \mathrel{\hat{\in}} \mathcal{S}_q$ holds for each query clause $\Gamma_Q \to \Delta_Q$ with $\mathcal{O} \models \Gamma_Q \to \Delta_Q$ and each context $q \in \mathcal{V}$ satisfying both of the following properties, where $\succ_q$ and $\mathcal{S}_q$ respectively denote the context order and set of clauses assigned to q in $\mathcal{D}$:

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

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 $s \approx \mathsf{true} \in \Delta_Q$.*

The context order for Sequoia described above satisfies Condition C2 of Theorem 2 for any query Q. Given a specific query Q, however, the efficiency of the calculus can be improved by introducing a context q in the context structure with a context order $\succ_q$ especially tailored for Q. For example, we can strengthen the context order described above by comparing all binary atoms as well as unary atoms not mentioned in Q. Such order is still admissible with respect to $\gg$ and satisfies Condition C2 of Theorem 2 for Q. We use this technique to improve the performance of concept subsumption and classification in Sequoia.

Hyper	If	$\bigwedge_{i=1}^{n} A_i \to \Delta \in O$,
		σ is a substitution such that $\sigma(x) = x$ or
		$\sigma(x) \in \Sigma_u$ if either $v = v_r$ or $A_i\sigma$ contains x for some $1 \leq i \leq n$,
		$\Gamma_i \to \Delta_i \vee A_i\sigma \in \mathcal{S}_v$ 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 $\mathcal{S}_v$.
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 is ground, is contained in $\mathsf{Su}^r(O)$, and contains $u \in \Sigma_u^O$,
		there is no $\langle v, v_r, u\rangle \in \mathcal{E}$ such that $\top \to A \ \hat{\in}\ \mathcal{S}_{v_r}$,
	then	add $\langle v, v_r, u\rangle$ to $\mathcal{E}$ and $\top \to A$ to $\mathcal{S}_{v_r}$.

Table 4: Modified versions of the Hyper and r-Succ rules for O.

4.3. Variant of the Calculus for Horn Ontologies

We now define a variant of our calculus that is applicable to Horn ontologies and enjoys better computational properties when checking entailment of a single query clause with respect to Horn ontologies containing nominals (as we discuss in Section 5).

The modified calculus replaces the Hyper rule from Table 2 and the r-Succ rule from Table 3 with the rules shown in Table 4. The version of the r-Succ rule in Table 4 differs from that in Table 3 as it only derives clauses with an empty body; this ensures that context clauses derived by this rule satisfy condition Z1 of Definition 8. Furthermore, the r-Succ rule in Table 4 is not triggered for atoms of the form $S(u, x)$ and $S(x, u)$, 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 $u \in \Sigma_u^O$ 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.

The notion of soundness for a context structure in this variant of the calculus differs from Definition 6 in that it is relative to a conjunction K of atoms of the form $B(x)$. In particular, the context clauses derived in $\mathcal{D}$ are only guaranteed to be satisfied by models of O for which the interpretation of K is not empty.

Definition 8. *Let $K = B_1(x) \wedge \cdots \wedge B_n(x)$ be a conjunction of atoms where $B_i \in \Sigma_A^O$ for $1 \leq i \leq n$. Let $\mathcal{D}$ be a context structure $\langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \theta\rangle$ for O and let $C_{\mathcal{D}}$ be the clause set defined in Definition 6. 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 Σ_f^O;

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

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

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

Theorem 3 (Soundness)**.** *Assume that O is Horn and K is a conjunction of atoms of the form $B(x)$ for $B \in \Sigma_A^O$. Given a context structure $\mathcal{D}$ for O which is sound for K, the application to $\mathcal{D}$ of a rule from Table 4, a rule from Table 2 other than* Hyper*, or a rule from Table 3 other than r-*Succ*, with the eager context strategy, yields a context structure for O which is sound for K.*

The proof of Theorem 3 is analogous to that of Theorem 1, and it is presented in Appendix E. The completeness statement is similar to Theorem 2, but is restricted to query clauses of the form $K \to \Delta$, where the saturated context structure is sound for K. As a result, context structures created to decide a query clause of the form $K \to \Delta$ cannot generally be reused for deciding entailment of query clauses with different bodies. The proof of Theorem 4 is given in Appendix E.

Theorem 4 (Completeness)**.** *Assume that O is Horn. Let $\mathcal{D}$ be a context structure for O satisfying the following properties:*

- *it is derivable from a context structure for O that is sound for K and mentions no auxiliary constants;*
- *there exists an* a*-admissible total order $>$ on $\Sigma_f \cup \Sigma_u$ such that every context is assigned a context order admissible with respect to $>$; and*
- *no rule from Table 4, or Table 2 except* Hyper*, or Table 3 except r-*Succ*, can be applied.*

$$
\begin{array}{rcll}
A \sqsubseteq \exists R.B & \rightsquigarrow & A(x) \to B(f(x)) & (83)\\
 & & A(x) \to R(x,f(x)) & (84)\\
B \sqsubseteq \exists S.C & \rightsquigarrow & B(x) \to C(g(x)) & (85)\\
 & & B(x) \to S(x,g(x)) & (86)\\
C \sqsubseteq F \sqcup D & \rightsquigarrow & C(x) \to F(x) \vee D(x) & (87)
\end{array}
\qquad
\begin{array}{rcll}
C \sqcap D \sqsubseteq E & \rightsquigarrow & C(x) \wedge D(x) \to E(x) & (88)\\
\exists R.F \sqsubseteq F & \rightsquigarrow & R(z_1,x) \wedge F(x) \to F(z_1) & (89)\\
\exists S.E \sqsubseteq F & \rightsquigarrow & S(z_1,x) \wedge E(x) \to F(z_1) & (90)\\
C \sqsubseteq \{o\} & \rightsquigarrow & C(x) \to x \approx o & (91)\\
D \sqsubseteq \{o\} & \rightsquigarrow & D(x) \to x \approx o & (92)
\end{array}
$$

Figure 6: Ontology O_3.

$$
\begin{array}{rl}
\top \to A(x) & (93)\\
\top \to F(x) \vee D(x) & (94)\\
\top \to F(x) \vee x \approx o & (95)\\
\top \to F(x) \vee D(o) & (96)\\
\top \to B(f(x)) & (97)\\
\top \to R(x,f(x)) & (98)\\
\top \to F(x) & (99)
\end{array}
$$

$$
\begin{array}{rl}
\top \to C(x) & (100)\\
S(y,x) \to S(y,x) & (101)\\
D(o) \to D(o) & (102)\\
\top \to x \approx o & (103)
\end{array}
$$

$$\begin{tikzcd}
v_A \arrow[d, "o"] \arrow[dr, "f"] & v_C \\
v_r & v_B \arrow[l, "o"] \arrow[u, "g"]
\end{tikzcd}$$

$$
\begin{array}{rl}
\top \to B(x) & (104)\\
R(y,x) \to R(y,x) & (105)\\
D(o) \to D(o) & (106)\\
\top \to C(g(x)) & (107)\\
\top \to S(x,g(x)) & (108)\\
\top \to g(x) \approx o & (109)\\
\top \to C(o) & (110)\\
\top \to S(x,o) & (111)\\
D(o) \to F(x) & (112)\\
D(o) \wedge R(y,x) \to F(y) & (113)
\end{array}
$$

$$
\begin{array}{rl}
D(o) \to D(o) & (114)\\
C(o) \to C(o) & (115)\\
S(y,o) \to S(y,o) & (116)\\
C(o) \wedge D(o) \to E(o) & (117)\\
C(o) \wedge D(o) \wedge S(y,o) \to F(y) & (118)
\end{array}
$$

Figure 7: Calculus execution for Example 1. This figure summarises inferences and clauses that are relevant for deriving the query clause. Clauses (93) through (99) belong to context v_A with core $\{A(x)\}$; clauses (104) through (113) belong to context v_B with core $\{B(x)\}$, clauses (100) through (103) belong to context v_C with core $\{C(x)\}$; all other clauses belong to v_r with empty core.

Assume that the parameter Λ used in the Nom *rule satisfies $\Lambda \geq \tau^2_{\mathsf{Su}}$, with τ_{Su} defined as in Theorem 2, and that no two contexts in $\mathcal{D}$ have the same core.*

Then, $K \to \Delta_Q \,\hat{\in}\, \mathcal{S}_q$ holds for each query clause $K \to \Delta_Q$ such that $O \models K \to \Delta_Q$ and each context $q \in \mathcal{V}$ satisfying the following properties, where $>_q$ and $\mathcal{S}_q$ respectively denote the context order and set of clauses assigned to q in $\mathcal{D}$:

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 $A >_q s$, we have $s \approx \mathsf{true} \in \Delta_Q$; and*

C3. every context distinct from v_r is reachable from q or v_r via edges in $\mathcal{E}$ labelled with symbols of Σ_f.

4.4. Examples

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

Example 1. *Consider ontology O_3 from Figure 6. We apply our calculus to prove $O_3 \models A(x) \to F(x)$. 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 $\{A(x)\}$. We use an expansion strategy that, on input K_1, always selects the context v_K for K the set of unary atoms in K_1; if such context does not yet exist in $\mathcal{D}$, the strategy creates it. We assume that all contexts use the context order defined in Section 4.2.*

Rule Core *adds clause (93), which ensures that v_A represents all instances of A in an arbitrary model. An application of* Hyper *on clause (93) with ontology clause (83) yields clause (94). Since $F(x)$ cannot be bigger than $D(x)$ by our choice of $>_{v_A}$, we can apply* Hyper *to $D(x)$ in clause (94) and ontology clause (92) to obtain clause (95). Once again, we have that $F(x)$ cannot be bigger than o, so we can apply* Eq *to $x \approx o$ in clause (95) and $D(x)$ in clause (94) to generate clause (96). This clause requires that, in any model where t is an instance 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 (114) to v_r and an edge $\langle v_A, v_r, o\rangle$. This clause illustrates that considering models where ground atom $D(o)$ is satisfied can be relevant for deciding our target query.*

Clauses (97) and (98), and then clauses (104) through (108) and clauses (100) through (103), are derived by applications of rules Core, Succ, *and* Hyper *that are analogous to those in the example in Figure 3 of Section 3.1. Next, notice that clause (103) can be back-propagated to v_B via* Pred, *and this leads to clause (109) in v_B. An application of* Eq *to such clause and clause (107)*

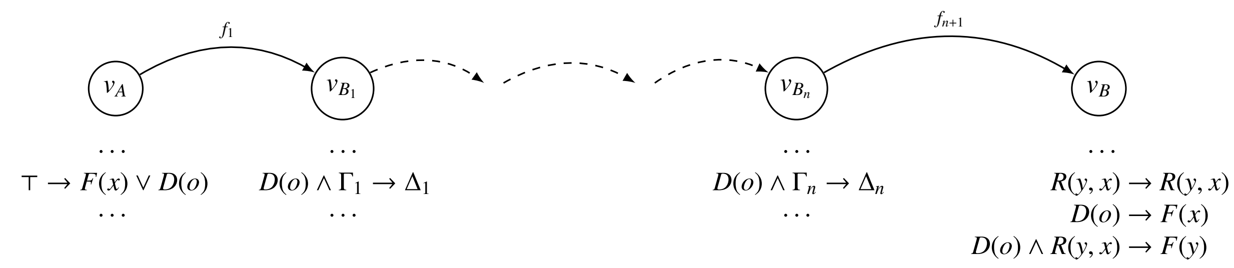

f_1
f_{n+1}
v_A
v_{B_1}
v_{B_n}
v_B
$\top \to F(x) \vee D(o)$
$D(o) \wedge \Gamma_1 \to \Delta_1$
$D(o) \wedge \Gamma_n \to \Delta_n$
$R(y,x) \to R(y,x)$
$D(o) \to F(x)$
$D(o) \wedge R(y,x) \to F(y)$

Figure 8: Sketch of a context structure representing an interaction analogous to Example 1, but where contexts v_A and v_B are arbitrarily distant

leads to clause (110). Please observe that this clause does not represent that $\mathcal{O} \models C(o)$; instead, this clause must be interpreted as saying that $C(o)$ is satisfied in those models where the interpretation of B is non-empty. Another application of Eq *to clause (109), with clause (108), yields clause (111). This clause, together with clause (110), can be propagated via r-*Succ *to the root context, where they are written as clause (116) and clause (115), respectively. Observe that the heads of clauses (114) and (115) unify with the body of clause (88), and since the ground context allows applications of* Hyper *which unify x with a constant, we obtain clause (117). In turn, this clause and clause (116) can participate in a new* Hyper *inference with clause (90) to produce clause (118). 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 (118) to v_B, using also clauses (106), (110) and (111), to produce clause (112).*

Deriving clause (112) would not have been possible without generating clause (114), which highlights the relevance of atom $D(o)$ in determining whether the target query holds. Now we have expanded upon this result 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 (105) and (112) can participate in a Hyper *inference with clause (89) to produce clause (113). The head of this clause triggers* Pred *with clause (98) and clause (96), and propagates as clause (99) to v_A, which proves $\mathcal{O} \models A(x) \to F(x)$.* □

Example 1 illustrates how ground literals in separate contexts can interact with each other via the root context. Although in this example the interaction involves contexts which are next to each other, it can also involve non-root contexts that are arbitrarily far apart from each other. To see this, consider Figure 8 and suppose the ontology is such that clause (113) can be back-propagated via Pred to a clause of the form $D(o) \wedge \Gamma_n \to \Delta_n$ in a predecessor v_{B_n} of v_B, with $\Gamma_n \to \Delta_n$ an arbitrary context clause. In turn, it could be the case that clause $D(o) \wedge \Gamma_n \to \Delta_n$ later participates in 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 n contexts $v_{B_1}, \dots, v_{B_n}$ 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 (96), 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. *Let $\mathcal{O}_4$ be the ontology from Figure 9 and let us consider again the query $A(x) \to F(x)$. Our initial context structure contains the root context v_r and a context v_A with core $\{A(x)\}$. We choose the eager strategy, which will produce shorter clauses and simplify the example, and assume that all contexts use the context order defined in Section 4.2. We skip the discussion of the derivation of clauses (132) through (136), clauses (143) through (146), and clauses (164) and (165), since these clauses can be obtained through inferences analogous to those in the example from Figure 2 in Section 3.1. Such clauses ensure that every instance of A in a model has a successor via R which is an instance of B_1, and every such element has a successor via S_1 which is an instance of C. Clause (128) represents that S_2 is the inverse of S_1, and so an inference by* Hyper *with clause (165) leads to clause (166). Now, we apply* Hyper *to clauses (127) and (164) to derive clause (167). The last two clauses in this context can be propagated via* Pred *to produce clauses (147) and (148). An inference with* Eq *using these two clauses yields clause (149).*

The inverse role S_2 is now interacting with nominal o and the number restriction in ontology clause (131). Indeed, clause (149) shows that (the interpretation of) o connects via S_2 to every instance of $\{B_1(x), R(y,x)\}$ in a model of $\mathcal{O}_4$. By clause (131), however, o can only have one successor via S_2. Thus, all instances of $\{B_1(x), R(y,x)\}$ in a model of $\mathcal{O}_4$ must be identical, and hence context v_{B_1} acts like a nominal. To capture this behaviour explicitly, rule Nom *derives clause (150) from clause (149) in v_{B_1}, with $o_\rho = o_{S_2^1}$ a fresh auxiliary constant. We can see clause (150) as capturing a consequence of clause (131) that is relevant for our derivation: every instance of $\{B_1(x), R(y,x)\}$ must be equal to the interpretation of individual o_ρ, where o is connected to o_ρ via S_2. Next, this clause can be propagated to v_A with* Pred *to yield clause (137), which we use in two inferences by* Eq *on v_A and clauses (133) and (134) to yield clauses (138) and (139), respectively.*

*The derivation illustrated so far can be straightforwardly "mirrored" starting from clauses (135) and (136), and using context v_{B_2}. This eventually results in the derivation of clauses (140) and (141) in v_A, analogous to clauses (137) and (138). Notice that the presence of o_ρ in this context ensures that r-*Succ *is triggered and clauses (159) through (161) 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*

$$
\begin{array}{rcll}
A \sqsubseteq \exists R.B_1 \rightsquigarrow & A(x) & \rightarrow & B_1(f(x)) & (119)\\
 & A(x) & \rightarrow & R(x,f(x)) & (120)\\
A \sqsubseteq \exists R.B_2 \rightsquigarrow & A(x) & \rightarrow & B_2(g(x)) & (121)\\
 & A(x) & \rightarrow & R(x,g(x)) & (122)\\
B_1 \sqsubseteq \exists S_1.C \rightsquigarrow & B_1(x) & \rightarrow & C(h(x)) & (123)\\
 & B_1(x) & \rightarrow & S_1(x,h(x)) & (124)\\
B_2 \sqsubseteq \exists S_1.C \rightsquigarrow & B_2(x) & \rightarrow & C(h'(x)) & (125)\\
 & B_2(x) & \rightarrow & S_1(x,h'(x)) & (126)
\end{array}
$$

$$
\begin{array}{rclcrcll}
C & \sqsubseteq & \{o\} & \rightsquigarrow & C(x) & \rightarrow & x \approx o & (127)\\
S_1 & \sqsubseteq & S_2^- & \rightsquigarrow & S_1(z_1,x) & \rightarrow & S_2(x,z_1) & (128)\\
B_1 \sqcap B_2 & \sqsubseteq & B & \rightsquigarrow & B_1(x) \wedge B_2(x) & \rightarrow & B(x) & (129)\\
\exists R.B & \sqsubseteq & F & \rightsquigarrow & R(z_1,x) \wedge B(x) & \rightarrow & F(z_1) & (130)\\
\top & \sqsubseteq & \leqslant 1\, S_2.\top & \rightsquigarrow & S_2(x,z_1) \wedge S_2(x,z_2) & \rightarrow & z_1 \approx z_2 & (131)
\end{array}
$$

Figure 9: Ontology O_4.

$$\begin{tikzcd}
 & v_{B_1} \arrow[dr, "h"] \arrow[ddrr, "o"] & & \\
v_A \arrow[ur, "f"] \arrow[dr, "g"'] \arrow[drrr, bend left=70, "o_\rho"] & & v_C & \\
 & v_{B_2} \arrow[ur, "h'"] \arrow[rr, "o"'] & & v_r
\end{tikzcd}$$

$\top \rightarrow A(x)$	(132)
$\top \rightarrow B_1(f(x))$	(133)
$\top \rightarrow R(x,f(x))$	(134)
$\top \rightarrow B_2(g(x))$	(135)
$\top \rightarrow R(x,g(x))$	(136)
$\top \rightarrow f(x) \approx o_\rho$	(137)
$\top \rightarrow B_1(o_\rho)$	(138)
$\top \rightarrow R(x,o_\rho)$	(139)
$\top \rightarrow g(x) \approx o_\rho$	(140)
$\top \rightarrow B_2(o_\rho)$	(141)
$\top \rightarrow F(x)$	(142)

$\top \rightarrow B_1(x)$	(143)
$\top \rightarrow R(y,x)$	(144)
$\top \rightarrow C(h(x))$	(145)
$\top \rightarrow S_1(x,h(x))$	(146)
$\top \rightarrow S_2(h(x),x)$	(147)
$\top \rightarrow h(x) \approx o$	(148)
$\top \rightarrow S_2(o,x)$	(149)
$\top \rightarrow x \approx o_\rho$	(150)

$\top \rightarrow B_2(x)$	(151)
$\top \rightarrow R(y,x)$	(152)
$\top \rightarrow C(h'(x))$	(153)
$\top \rightarrow S_1(x,h'(x))$	(154)
$\top \rightarrow S_2(h'(x),x)$	(155)
$\top \rightarrow h'(x) \approx o$	(156)
$\top \rightarrow S_2(o,x)$	(157)
$\top \rightarrow x \approx o_\rho$	(158)

$B_1(o_\rho) \rightarrow B_1(o_\rho)$	(159)
$R(y,o_\rho) \rightarrow R(y,o_\rho)$	(160)
$B_2(o_\rho) \rightarrow B_2(o_\rho)$	(161)
$B_1(o_\rho) \wedge B_2(o_\rho) \rightarrow B(o_\rho)$	(162)
$R(y,o_\rho) \wedge B_1(o_\rho) \wedge B_2(o_\rho) \rightarrow F(y)$	(163)

$\top \rightarrow C(x)$	(164)
$\top \rightarrow S_1(y,x)$	(165)
$\top \rightarrow S_2(x,y)$	(166)
$\top \rightarrow x \approx o$	(167)

Figure 10: Relevant clauses for Example 2. Clauses (132)–(142) belong to v_A with core $\{A(x)\}$; clauses (143)–(150) to v_{B_1} with core $\{B_1(x), R(y,x)\}$; clauses (151)–(158) to v_{B_2} with core $\{B_2(x), R(y,x)\}$; clauses (164)–(167) to v_C with core $\{C(x), S_1(y,x)\}$; all other clauses belong to v_r with empty core.

*o_ρ, which leads to the derivation of clauses (162) and (163) using ontology clauses (129) and (130), respectively. The head of the latter clause triggers r-*Pred*, and we can propagate it to v_A with the help of clauses (138), (139) and (141). This leads to our target query, represented by clause (142).*

5. Solving DL reasoning Problems Using the Calculus

This section describes how our calculus can be exploited to decide the following standard DL reasoning tasks:

- *Consistency checking:* decide whether there exists a model of a given ontology $\mathcal{O}$.
- *Subsumption:* given concepts C and D and ontology $\mathcal{O}$, decide whether each instance of C is also an instance of D in each model of $\mathcal{O}$.
- *Classification:* given ontology $\mathcal{O}$, decide whether B_1 is subsumed by B_2 for each pair of concept names in $\mathcal{O}$.
- *Instance retrieval:* given ontology $\mathcal{O}$, find all constants in $\mathcal{O}$ that are instances of a given concept C in all models of $\mathcal{O}$.
- *Realisation:* for each constant o in a given ontology $\mathcal{O}$, compute the set of concept names B such that o is an instance of B in each model of $\mathcal{O}$, and there is no other concept name B' subsumed by B such that o is also an instance of B'.

In Section 5.1 we propose a general algorithm which, given an $\mathcal{ALCHOIQ}^+$ ontology and a set of query clauses, returns those entailed by the ontology. We then establish soundness, completeness, and termination of the algorithm, we discuss its worst-case complexity, and we 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 5.2, we describe a variant of the algorithm based on the modified calculus described in Section 4.3, and we show that this variant of the algorithm is worst-case optimal for classification in Horn fragments of $\mathcal{ALCHOIQ}^+$ with nominals. Similarly to the previous section, we fix an arbitrary $\mathcal{ALCHOIQ}^+$ ontology $\mathcal{O}$ over a DL signature Σ for the remainder of this section.

5.1. The General Subsumption Checking Algorithm

Given a set of query clauses $\mathbf{Q} = \{Q_1, \dots, Q_n\}$ defined over Σ, 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 generalises the algorithm in [11] for checking query clause entailment in $\mathcal{ALCHIQ}^+$.

Step A1 initialises the context structure with the root context v_r containing no clauses; it also chooses an a-admissible order on successor function symbols and constants, and introduces a context order for v_r admissible with respect to $\succ$.

Step A2 creates contexts with the appropriate parameters and initial clauses to ensure that, once they are saturated, we can read whether each query clause is entailed or not by looking at the consequences derived in these contexts.

For the algorithm to terminate, we need to ensure that the expansion strategy does not introduce an infinite number of contexts. Hence, in Step A3 our algorithm selects an *admissible* expansion strategy (as defined next). This definition also requires that each application of the expansion strategy during the saturation phase can be performed in polynomial time.

Definition 9. *An expansion strategy* strat *for $\mathcal{O}$ is* admissible *if it is computable in polynomial time and there exists a $\omega \in \mathbb{N}$ such that for every context structure $\mathcal{D}$ for $\mathcal{O}$ and every $\Lambda \in \mathbb{N}$, the saturation of $\mathcal{D}$ by the rules from Table 2 and Table 3 using parameters* strat *and Λ adds at most ω contexts to $\mathcal{D}$.*

All expansion strategies discussed in Section 3.1 and Section 4.1 are admissible. In particular, each application of these strategies can be computed in polynomial time; furthermore, for m and m' the numbers of unary and binary predicates in $\mathcal{O}$, the eager strategy introduces at most $2^{m+m'+1}$ contexts, whereas the cautious strategy can introduce at most m contexts.

Having fixed the expansion strategy, in Step A4 the algorithm selects a depth limit Λ for the nominal labels which is large enough to satisfy the corresponding condition in Theorem 2.

Finally, Step A5 saturates the initial context structure, and Step A6 checks whether each target query has been derived in the corresponding context that was introduced during initialisation.

Soundness and completeness of Algorithm 1 are a consequence of Theorems 1 and 2.

Corollary 1 (Soundness and Completeness)**.** *Let $\mathbf{Q}$ be a set of query clauses over Σ, and let $\mathbf{Q}^+$ be the output of Algorithm 1 on input $\mathbf{Q}$. Then, 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 A5. Let $\mathcal{J}$ be the extension of $\mathcal{I}$ to $\Sigma_u^{\mathcal{O}}$ defined inductively as given next:

- $o_\epsilon^{\mathcal{J}} = o_\epsilon^{\mathcal{I}}$;

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 $\mathcal{D}$ for $\mathcal{O}$ with $\mathcal{V} = \{v_r\}$, $\mathcal{E} = \emptyset$, and $\mathcal{S}_{v_r} = \emptyset$. This step includes the selection of an a-admissible order $\succ$ on
    $\Sigma_f \cup \Sigma_u$ and a choice of context order $\succ_{v_r}$ admissible with respect to $\succ$.
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 a context order $\theta(v_i) = \succ_{v_i}$ admissible for $\succ$ and satisfying condition C2 of
        Theorem 2 w.r.t. $\Delta_{Q_i}$.
    then add $A \to A$ to $\mathcal{S}_{v_i}$ for each $A \in \Gamma_{Q_i} \setminus \mathsf{core}_{v_i}$.
A3. Select an admissible expansion strategy strat for $\mathcal{O}$ which can introduce at most $\omega$ contexts and such that all context orders it introduces
    are admissible with respect to $\succ$ .
A4. Select a depth limit $\Lambda$ for the nominal labels equal to $2^{\tau_{\mathsf{Su}}} \cdot 2^{\tau_{\mathsf{Pr}}} \cdot (\omega + \omega_{\mathcal{D}})$, where $\tau_{\mathsf{Su}}$ and $\tau_{\mathsf{Pr}}$ are defined as in Theorem 2, and $\omega_{\mathcal{D}}$ is the
    number of contexts in $\mathcal{D}$.
A5. Apply the inference rules from Table 2 and Table 3 using the parameters strat and $\Lambda$, until no new inferences are possible.
A6. Return $\mathbf{Q}_+ = \{Q_i \in \mathbf{Q} \mid \Gamma_i \to \Delta_i \hat{\in} \mathcal{S}_{v_i}\}$.
```

- $o_\rho^{\mathcal{J}}$ with $\rho = \rho' \cdot S_{B_2}^i$ is defined as follows: if there are exactly m domain elements $a_1, \ldots a_m$ in $\mathcal{I}$ such that $(o_{\rho'}^{\mathcal{I}}, a_i) \in S_{B_2}^{\mathcal{I}}$ 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, $\mathcal{J} \models \mathcal{O} \cup C_{\mathcal{D}}$, with $C_{\mathcal{D}}$ as in Definition 6. 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, let strat and Λ be the parameters chosen in Steps A3 and A4, and let $\mathcal{D}$ be the saturated context structure. Notice that $\mathcal{D}$ is derivable from the context structure initialised in Step A2, which is trivially sound and mentions no auxiliary constants. Furthermore, according to Step A3, each context in $\mathcal{D}$ is assigned a context order admissible with respect to $\succ$. Next, since strat is admissible, the number of contexts in $\mathcal{D}$ is smaller or equal to $\omega + \omega_{\mathcal{D}}$. Finally, our choice of parameters in Step A2 ensures that Conditions C1 and C2 are satisfied for Q and q. Therefore, by Theorem 2, we conclude $Q \hat{\in} \mathcal{S}_q$. Step A6 of Algorithm 1 then ensures $Q \in \mathbf{Q}^+$. □

As already mentioned, termination is guaranteed for admissible expansion strategies. Indeed, the signature of the context structure is finite, and no inference is performed twice, so there is a limit on the total number of context clauses that the calculus can derive. Furthermore, if the number of contexts generated by the expansion strategy can be bound exponentially in the size of $\mathcal{O}$ (as is the case for the trivial, eager, and cautious strategies), our choice of Λ ensures that our algorithm runs in triple exponential time—a bound that is in line with those obtained using the tableau and resolution-based procedures discussed in Section 6 [38, 24]. We refer the reader to Appendix F for a proof of Theorem 5.

Theorem 5 (Termination)**.** *Algorithm 1 is terminating. Furthermore, Algorithm 1 runs in 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 $\mathcal{O}$.*

Our algorithm is therefore not worst-case optimal for $\mathcal{ALCHOIQ}^+$, which is an NExpTime-complete logic [39]. 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 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 [11]. By carefully choosing the expansion strategy, the algorithm can also be worst-case polynomial (and therefore optimal) for classification in the DL $\mathcal{ELH}$.

Theorem 6 (Pay-as-you-go Behaviour)**.** *Let* strat *be an expansion strategy for $\mathcal{O}$ introducing at most exponentially many contexts. If Step A3 selects* strat*, then Algorithm 1 runs in exponential time in the size of $\mathcal{O}$ if this ontology is either $\mathcal{ALCHIQ}^+$, $\mathcal{ALCHOQ}$, or $\mathcal{ALCHOI}$; furthermore, if $\mathcal{O}$ is $\mathcal{ELH}$, the algorithm runs in polynomial time in the size of $\mathcal{O}$ with either the cautious or the eager strategy.*

We next show how, by adjusting the initialisation parameters, we can exploit Algorithm 1 to solve each of the standard DL reasoning tasks.

To solve consistency checking, it suffices to set $\mathbf{Q} = \{\top \to \bot\}$. Since query clause $\top \to \bot$ is inconsistent, $\mathcal{O} \models \top \to \bot$ 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 Σ_A. This leads to a polynomial increase in the size for $\mathcal{O}$. Finally, we run Algorithm 1 on $\mathcal{O}'$ and query $\{B_C(x) \to B_D(x)\}$ 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 $\mathbf{Q} = \{B_i(x) \to B_j(x) \mid B_i, B_j \text{ are unary predicates in } \mathcal{O}\}$. The result $\mathbf{Q}^+$ consists of all subsumptions of the form $B_i(x) \to B_j(x)$ entailed by $\mathcal{O}$, and it can be used to build a taxonomy. To improve performance, it is often helpful to introduce in Step A2 a single context v_i with core $\{B_i(x)\}$ for each unary predicate B_i and ensure that the corresponding context order $\succ_i$ makes all unary atoms incomparable with each other.

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 Σ_A. We then introduce a fresh unary B_o for each $o_\epsilon \in \Sigma_u^{\mathcal{O}}$, and extend $\mathcal{O}$ also with the set

$$C_u = \bigcup_{o_\epsilon \in \Sigma_u^{\mathcal{O}}} \{\, B_o(x) \to x \approx o, \ \top \to B_o(o) \,\}$$

to obtain $\mathcal{O}'$. We can then run Algorithm 1 on $\mathcal{O}'$ and query $\mathbf{Q} = \{B_o(x) \to B_C(x) \mid o_\epsilon \in \Sigma_u^{\mathcal{O}}\}$ since $\mathcal{O}' \models B_o(x) \to B_C(x)$ if and only if $\mathcal{O} \models C(o)$. To recover the instances of C, simply output constant o for each B_o mentioned in $\mathbf{Q}^+$.

Finally, to solve realisation, we extend $\mathcal{O}$ with the aforementioned set of clauses C_u to obtain $\mathcal{O}'$ and run Algorithm 1 on $\mathcal{O}'$ and $\mathbf{Q} = \{B_o(x) \to B(x) \mid o_\epsilon \in \Sigma_u^{\mathcal{O}}, B \text{ a unary predicate}\}$. It holds that $\mathcal{O} \cup C_u \models B_o(x) \to B(x)$ if and only if $\mathcal{O} \models B(o)$; hence, for each $o_\epsilon \in \Sigma_u^{\mathcal{O}}$, we know that $B_o(x) \to B(x)$ is in the output $\mathbf{Q}^+$ if and only if B realises o. We can use these results to construct the set of unary predicates instantiated by o_ϵ. To compute the subset of most specific such concepts, we extend $\mathbf{Q}$ with the set $\{B_i(x) \to B_j(x) \mid B_i, B_j \text{ unary predicates}\}$, so that we can use $\mathbf{Q}^+$ to compute the taxonomy of $\mathcal{O}$ as we did for classification.

It is important to note that a saturated context structure $\mathcal{D}$ used to solve any of these tasks can be reused to solve other tasks. 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, one can simply choose one of the contexts that already exist and weaken the context order to ensure that condition C2 of Theorem 2 is satisfied. Soundness of this method is also guaranteed because Theorem 1 does not depend on the initialisation of the context structure.

5.2. 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 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 4.3 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 eager strategy for $\mathcal{O}$ is fixed as the expansion strategy of Algorithm 2. This ensures that any context structure derived by the algorithm preserves soundness for K.

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 $B \in \Sigma_A^{\mathcal{O}}$, this algorithm decides whether $\mathcal{O} \models K \to \Delta_{Q_i}$ for each $Q_i \in \mathbf{Q}$, in the case where $\mathcal{O}$ is Horn.

```
A1. Create a context structure $\mathcal{D}$ for $\mathcal{O}$ with $\mathcal{V} = \{v_r\}$, $\mathcal{E} = \emptyset$, and $\mathcal{S}_{v_r} = \emptyset$. This step includes the selection of an a-admissible order $\succ$ and a choice of context order $\succ_{v_r}$ compatible with $\succ$.
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 depth limit $\Lambda = \tau_{\mathsf{Su}}^2$ for nominal labels, with $\tau_{\mathsf{Su}}$ defined as in Theorem 2.
A4. Apply the inference rules from Table 4, Table 2 minus Hyper, and Table 3 minus $r$-Succ, to $\mathcal{D}$ and $\mathcal{O}$, with the eager expansion strategy, until no new inferences are possible.
A5. Return $\mathbf{Q}_+ = \{Q_i \in \mathbf{Q} \mid \Gamma_i \to \Delta_i \,\hat{\in}\, \mathcal{S}_{v_i}\}$.
```

Corollary 2 (Soundness and Completeness)**.** *Suppose that $\mathcal{O}$ is Horn. Let $\mathbf{Q}$ be a set of query clauses over Σ 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 $B \in \Sigma_A^{\mathcal{O}}$, and let $\mathbf{Q}^+$ be the output of Algorithm 2 on input $\mathbf{Q}$. 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 1: 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 1, because the initial context structure defined in Steps A1 and A2 is clearly sound for K by Theorem 3, and therefore $\mathcal{D}$ is sound for K. Furthermore, Λ is large enough to satisfy the corresponding condition in Theorem 4, and the eager strategy never introduces two contexts with the same core. □

The termination and relevant complexity results for Horn DLs are stated in Theorem 7; the proof is given in Appendix F.

Theorem 7 (Termination & Pay-as-you-go Behaviour)**.** *Algorithm 2 is terminating and runs in exponential time in the size of $\mathcal{O}$. Furthermore, if $\mathcal{O}$ is $\mathcal{ELHO}$ and the* Hyper *rule is applied eagerly, then Algorithm 2 runs in polynomial time in the size of $\mathcal{O}$.*

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 unary predicate B using the query $\{B(x) \to B'(x) \mid B' \text{ is a unary predicate in } \mathcal{O}\}$. Similarly, for instance retrieval and realisation, the algorithm must be run once for each $o_\epsilon \in \Sigma_u^{\mathcal{O}}$, plus once for each unary predicate B, since 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 reuse of 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 may very well pay off to use Algorithm 1 for one-pass classification despite the fact that the algorithm is worst-case exponential. Evidence for this hypothesis is provided in Section 8, where we compare empirically implementations of both algorithms on a corpus of Horn ontologies.

6. Discussion

In this section we compare our algorithms with other CB, tableau, and resolution procedures for DLs with nominals.

6.1. 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 [36] and [51], respectively. These calculi address the problem of non-local clauses discussed in Section 3.2 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.

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 $G \rightsquigarrow D$, where $\rightsquigarrow$ represents 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, for some $u \in \Sigma_u^{\mathcal{O}}$. The derivation of non-local consequences, however, leads to the loss of properties 1-3 discussed in Section 3.2. The authors in [36] already point out the difficulties of reusing consequences across different queries and avoiding the derivation of similar clauses. As a result, their $\mathcal{ELHO}$ calculus cannot achieve one-pass classification. In contrast, our calculus can reuse derived consequences across multiple queries, and in particular Algorithm 1 can classify ontologies in a single pass. 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 4.3 is similar to the calculus in [36] 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 [36] targeted towards reuse of clauses across queries may also be applicable in our setting.

The discussion above also applies to the Horn-$\mathcal{SROIQ}$ calculus in [51], 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 a 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 [51] must also deal with the problem discussed in Section 3.3 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 Nom rule from our calculus also introduces a single nominal. The main difference lies in the label of the new nominal: the calculus in [51] uses the name of the concept that collapsed, while we use the names of the linked constant and the role establishing the link.

6.2. Tableau and Resolution Calculi for $\mathcal{SHOIQ}$

When equipped with the trivial strategy, our calculus bears a resemblance to the resolution calculus in [38]. 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 [38] addresses the simultaneous interactions between inverse roles, number restrictions, and nominals, by means of specialised inference rules which introduce new constants, similarly to our Nom rule; furthermore, the calculus in [38] identifies "blocking" clauses of a certain form, the derivation of which prevents the introduction of new nominals and ensures both termination and completeness. We use in Appendix D.3 a similar argument: by allowing for a large enough nominal depth, we can ensure that blocking clauses of the same form as in [38] will have been derived.

Our calculus also presents similarities with the tableau procedure for $\mathcal{SHOIQ}$ introduced in [26]. The completeness proof of our calculus is based on the ability to use context clauses to build a countermodel of every target query that is not entailed by the ontology. Models built this way have a similar structure to the canonical 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 canonical model of the ontology.

7. Implementation

We have implemented Algorithm 1 in Section 5 as an extension of Sequoia[2], a CB reasoner for $\mathcal{SRIQ}$ presented in [10]. Our implementation has been released as version v.0.7. It supports the reasoning tasks in Section 5.1 for $\mathcal{SROIQ}$ ontologies, and is available through the OWL API interface. We have also developed an alternative version that implements Algorithm 2.

In Section 7.1 we overview the architecture and main features of Sequoia v.0.6, the previous version for $\mathcal{SRIQ}$. In Section 7.2 we describe the implementation of the new features which enable Sequoia to reason with nominals. Finally, in Section 7.3, we present several novel optimisations that have been incorporated in the latest version of the reasoner.

7.1. Overview of Sequoia v.0.6

Figure 11 summarises the architecture of Sequoia v.0.6. 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 [24], and then it applies the *Clausification* sub-routine for computing a set of DL-clauses using a variant of the structural transformation [50]. Like many 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 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 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 Section 5.1, this condition is required for completeness. Following [11], the context order requires also that all 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.

[2]`http://www.cs.ox.ac.uk/isg/tools/Sequoia/`

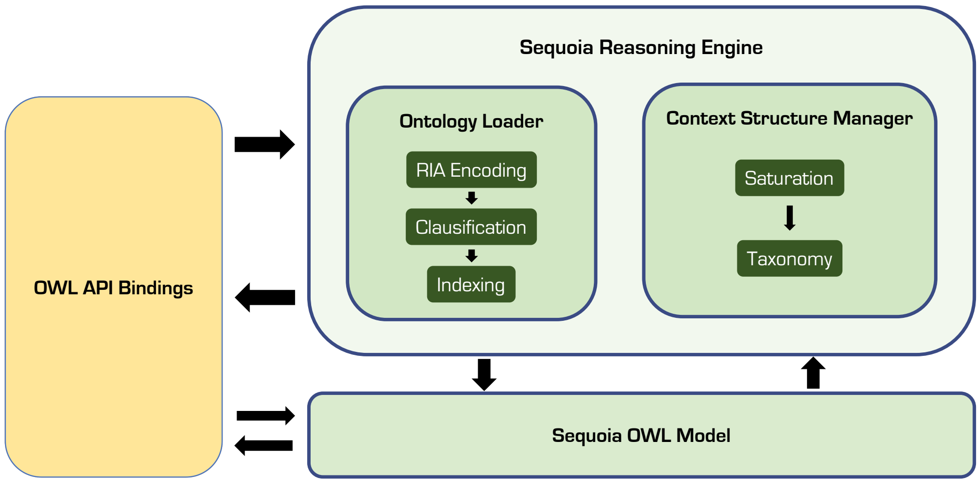


Figure 11: Architecture of Sequoia. Boxes represent key components, and arrows indicate flow of information.

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, i.e., concepts mentioned in the original ontology, where v_{K_1} and $\succ_{K_1}$ are as in the eager strategy; and otherwise defined as $\mathsf{safeCentral}(f, K_1, \mathcal{D}) = \langle v_{\beta(K_1)}, \beta(K_1), \succ_{\beta(K_1)}\rangle$, where β 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 $\mathcal{U}_v$. A set $\mathcal{P}_v$ is also created to hold clauses received from successors of v that may be relevant for Pred inferences in v. Sequoia then applies to v the steps in Algorithm 3. The Ineq rule is applied as a simplification after each new clause C is derived. 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 $C' \in \mathcal{S}_v$, and clauses in $\mathcal{S}_v$ subsumed by C are dropped when C is added to $\mathcal{S}_v$; these features implement the Elim rule. When the context structure has become saturated, Sequoia runs the *Taxonomy* sub-routine 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 Σ_f^O 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. The correspondence between long integers and literals is also very helpful during redundancy checks.

To implement forward and backward redundancy checking, Sequoia v.0.6 maintains a redundancy index in each context inspired by *feature vector indexing* from the theorem prover E [55]. 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 [11] 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 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, which 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

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 successor contexts 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.
```

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 for rules Pred and Eq; see [11] for details.

7.2. Implementing Support for Nominals

We next describe the modifications and extensions to the system described in Section 7.1 in order to implement support for nominals. Additionally, Sequoia v.0.7 supports all tasks discussed in Section 5.1, as well as novel features such as interruption, debugging mode, and extension of a loaded ontology with new axioms.

7.2.1. 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 are still 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 $u \in \Sigma_u^O$, we have that $y \succ_v u$ and $u \succ_v y$ are both forbidden by condition (4) of Definition 7. 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.

7.2.2. Fragmentation of the Root Context

Instead of using a single root context v_r, our implementation introduces a *nominal context* v_u with core $\{B_u(x)\}$ for each individual $u \in \Sigma_u^O$, with B_u a fresh unique predicate for u, and adds the clause $\top \to x \approx u$ to this context. Further contexts of this form are introduced whenever auxiliary constants are generated within the context structure. This method has several advantages.

- It makes the implementation of the calculus considerably easier. First, we can replace constant u with variable x in v_u using the Eq rule. Second, Hyper inferences that originally involved atoms of the form $B(u)$ in v_r can now be performed in the nominal context v_u using the unification pattern $B(x)$ and the indexing strategy defined in Section 7.1. Finally, Hyper inferences unifying x in an ontology clause with u in a context clause become unnecessary.

- It prevents the derivation of literals containing both constants and function symbols, which reduces the number of UIDs required to represent literals in v_u and thus allows for shorter representations of clauses in nominal contexts.
- It makes the algorithm more amenable to parallelisation—a feature that we expect to add in future versions of the system. Notice that the thread corresponding to the root context would require much more memory than any other thread (especially with large ABoxes), and it would need to communicate with most of the other threads. By splitting the root context, the computational workload and memory requirements can be spread across multiple threads.

To guarantee completeness, our implementation proceeds as follows. First, it ensures that each clause derived by an application of the r-Succ rule is added to the relevant nominal context. Second, if a clause having a constant in a maximal literal is derived in a nominal context, then the clause is propagated also to the nominal context representing that constant. Finally, if the maximal literal of a clause in a nominal context is a unary function-free ground atom, then this clause is propagated to all other nominal contexts. All of this ensures that our implementation does not miss any clause that would have been derived by the theoretical calculus in the single root context, or propagated to another context from the root context. This is so because, whenever our calculus performs an inference in the root context involving multiple premises, these premises always have one constant in common in their maximal literals, or involve maximal literals that are unary function-free ground atoms. Therefore, there exists always a nominal context in the implementation where this inference is carried out.

7.2.3. Implementation of Inference Rules

Rule Nom has been implemented in Step P3 of Algorithm 3, just after P3.(c). 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 $\Gamma \to \Delta \vee \Delta_{\approx} \in \mathcal{S}_v$ with $\Gamma \subseteq \Gamma_1 \cup \Gamma_2$, $\Delta \subseteq \Delta_1 \cup \Delta_2$, and such that each literal $L \in \Delta_{\approx}$ is of the form $x \approx y$, or $x \approx u$ or $y \approx u$ for some $u \in \Sigma_u^O$.

Rule Join is implemented just after Step P3.(a) 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 7.1. The propagation of clauses generated by r-Succ to other contexts occurs just after the propagation of clauses generated by Succ, in Step P4. In the case of nominal contexts, the propagation of clauses via r-Pred replaces the propagation of clauses via Pred.

Finally, the application of the Eq rule in our calculus may involve equalities of the form $x \approx u$ with $u \in \Sigma_u^O$, and result in the replacement of x with u (or vice-versa) in other clauses. The completeness proof in Appendix D, 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.

7.3. Optimisations

Next, we describe a suite of optimisation techniques that we have incorporated in Sequoia v.0.7.

7.3.1. Ordering of Query Atoms

As discussed in Section 5.1, classification requires that unary predicates in O (except auxiliary predicates introduced during normalisation) are incomparable in context orders assigned to *query contexts*, namely contexts initialised based on a query. This has a negative impact on performance: ontologies usually contain a large number of such predicates and many inferences occur in query contexts. Furthermore, the resolution of context clauses with unary predicates in O within a query context can lead to inefficient behaviour, as we illustrate next. Consider an ontology O_5 which contains, among others, the following clauses:

$$A(x) \to A_1(x) \vee \cdots \vee A_n(x) \quad (168) \qquad A_i(x) \to B(f_i(x)) \quad 1 \le i \le n \quad (169)$$
$$B(x) \to \bot \quad (170)$$

Let $A_i \in \Sigma_A^O$ for each $1 \le i \le n$, and suppose that no two atoms of in $\{A_i(x) \mid 1 \le i \le n\}$ can be compared in any query context. Consider a query context q containing clause $\top \to A(x)$. We have $O \models A(x) \to \bot$, so we expect to derive $\top \to \bot$ in q. Using clause $\top \to A(x)$, an inference by Hyper leads to:

$$\top \to A_1(x) \vee \cdots \vee A_n(x). \tag{171}$$

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 $\mathcal{U}_q$. 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 $\mathcal{P}_q$, derived via Pred from a successor context with a context clause $B(x) \to \bot$. In that case, for each $1 \le i \le n$, Sequoia will derive the clause

$$\top \to A_1(x) \vee \cdots \vee A_{i-1}(x) \vee A_{i+1}(x) \vee \cdots \vee A_n(x) \tag{172}$$

Although clause (172) makes clause (171) redundant, all clauses of the form (172) stay in $\mathcal{U}_q$ 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 $\top \rightarrow \bot$, 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 $\top \rightarrow \bot$ in linear time; this pattern appears in several ontologies in our evaluation, which Sequoia v.0.7 can now classify.

7.3.2. Controlled Propagation of the ABox

The application of rule Hyper ensures that ground atoms A in ontology clauses of the form $\top \rightarrow A$ (corresponding to ABox assertions) are propagated to every context. This, however, has proved problematic in practice as it results in very high memory usage. Furthermore, in contexts other than nominal contexts, these clauses are only relevant for redundancy elimination: they can block the derivation of clauses of the form $\Gamma \rightarrow \Delta \vee A$, or remove such clauses via the Elim rule. Therefore, clauses of the form $\top \rightarrow A$ would ideally be derived only in those contexts where they are relevant for redundancy checks.

Our approach to this issue has been to block the application of Hyper or r-Pred to ontology clauses of the form $\top \rightarrow A$ in all contexts except nominal contexts. This ensures that these clauses are not propagated to arbitrary contexts, thereby reducing memory usage and preventing the degradation of the performance of redundancy index checks. Later, if a clause of the form $\Gamma \rightarrow \Delta \vee A'$ is derived in a context v, where A' is a ground atom that mentions a constant $u \in \Sigma_u^{\mathcal{O}}$, Sequoia immediately applies Hyper to v for all clauses of the form $\top \rightarrow A$ in $\mathcal{O}$ such that A mentions u, to prevent the derivation in v of redundant clauses mentioning u. 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, which entail $\top \rightarrow D(o)$:

$$\top \rightarrow A(o) \qquad A(x) \rightarrow R(x, f(x)) \qquad A(x) \rightarrow B(f(x)) \qquad B(x) \rightarrow C(x) \qquad R(z_1, x) \wedge C(x) \rightarrow D(z_1).$$

Often, by the time $\top \rightarrow D(x)$ has been generated in the nominal context for o, redundant clauses containing $D(o)$ have already been derived in other contexts because of other ontology clauses. 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 when no constants 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, where o is replaced by x.

8. Evaluation

We have evaluated the performance of Sequoia on classification of real-world ontologies following the methodology by Steigmiller et al. [62]. The corpus selected for our experiment is the Oxford Ontology Repository, which consists of 799 ontologies with sizes ranging from ~1KB to ~1GB and expressiveness ranging from lightweight languages such as DL-Lite and $\mathcal{EL}$ to $\mathcal{SROIQ}(\mathcal{D})$.[3] Ontologies are sourced from a diverse range of repositories, such as the Gardiner Corpus [18], the OBO Foundry [61], and the Phenoscape project [9]. We processed each ontology by performing the following steps in the given order. As a result of this process, we obtained a corpus of 779 ontologies, out of which 104 contain nominals.

- *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.[4] 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.

[3]`http://www.cs.ox.ac.uk/isg/ontologies/`

[4]Ontologies with ID 158, 160, 291, and 785 required editing due to syntax errors.

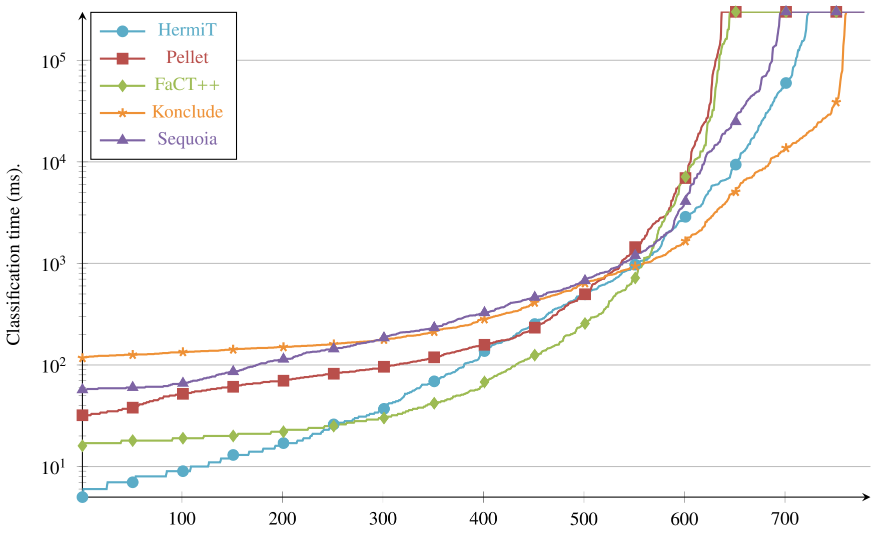


Figure 12: Classification Times for All Ontologies

- *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.

We used version v.0.7 of Sequoia, which has been described in Section 7. 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, 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 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 terminated 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 recorded the result as a failure. Taxonomies were hashed to check correctness. All systems and ontologies used in the experiment are available online.[5]

The (averaged) classification times for the full corpus are summarised in Figure 12. 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. Taxonomy hash values for Sequoia agreed with the values for at least one other reasoner in almost all cases; when this did not happen (often in ontologies classified only by one or two reasoners) we manually verified the equivalence of taxonomies. Please see the companion website for a full list of the test results and ontology hashes.

Figure 12 shows that the performance of Sequoia is competitive with that of well-established DL reasoners. Most timeouts (67 out of 82) occurred on non-Horn ontologies, where saturation produced many clauses with large numbers of disjuncts in the head—a well-known source of performance problems in CB reasoning [11]. These problems 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). It is interesting to observe, however, that Sequoia is the fastest amongst all the reasoners supporting OWL 2 DL to classify an ELK-supported version of SNOMED outside of the OWL 2 EL profile (ID 798),

[5]`http://krr-nas.cs.ox.ac.uk/2019/DL-sequoia/`

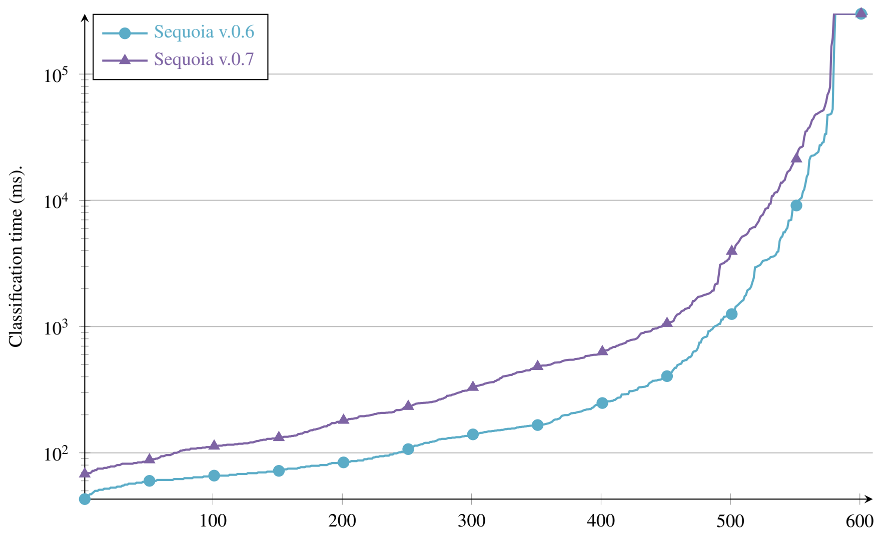


Figure 13: Classification Times for Sequoia v.0.7 and v.0.6. on Ontologies without Nominals or ABoxes

in approximately 160 seconds. In contrast, Konclude took almost twice as long, and the other reasoners timed out. For reference, the ELK reasoner required approximately 3 seconds to classify this ontology.

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 35 ontologies where Sequoia timed out; conversely, Sequoia successfully classified 13 ontologies in which HermiT timed out. This 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.

Overall, Sequoia could classify fewer ontologies with nominals than the other reasoners; for instance, it classified 22 fewer such ontologies than Pellet, and 40 fewer than Konclude. This shows that there is still plenty of room for optimisation when reasoning with nominals in Sequoia, especially in the case of ontologies with large ABoxes. The main challenge lies in the interactions between disjunction and nominals, which can lead to the derivation of a large number of clauses containing several disjuncts in the head where at least one of them is a ground atom. Furthermore, the presence of ABox axioms aggravates the known issues in CB reasoning related to number restrictions, since the generation of exponentially many clauses can occur in every nominal context. 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 interactions between nominals, inverse roles, and “at-most” number restrictions are rare.

To measure the effect of the optimisations described in Section 7.3, we disabled them one by one and ran the resulting modified versions of Sequoia on the entire corpus. The optimisation described in Section 7.3.1 improved the classification time of many ontologies in the corpus. Moreover, it allowed Sequoia to classify 9 additional ontologies. This result emphasises the importance of ordered resolution for restricting the number of inferences in our calculus. The optimisations described in Section 7.3.2 allowed Sequoia to classify 68 additional ontologies; furthermore, classification times for ontologies that did not require these optimisations were not significantly affected. The effect of these optimisations manifested almost entirely on ontologies with nominals and ABoxes (27 out of 68), or ontologies with ABoxes but no without nominals in the TBox (39 out of 68). Furthermore, in order to observe the effect of the implementation overhead described in Section 7.2.1, we ran Sequoia v.0.6 and v.0.7 on the subset of ontologies in the corpus that do not contain nominals. We also removed the ABoxes from these ontologies, since v.0.6 does not support them. Figure 13 summarises the average classification times of both reasoners on these ontologies. 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 7.2.1), 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.

We also evaluated the performance of the version of Sequoia implementing Algorithm 2 from Section 5, which is worst-case optimal for Horn ontologies but cannot achieve one-pass classification in ontologies with nominals. For this experiment, we ran both reasoners on the 612 Horn ontologies in our original corpus. The results show that for most Horn ontologies, the standard version of Sequoia performs better than the implementation of Algorithm 2, often improving the average classification time by a factor between 1 and 10. Although the implementation of Algorithm 2 has faster average times in a small number of ontologies, all ontologies that can be classified by it can also be classified by the standard version of Sequoia. We interpret the results of this experiment as evidence that one-pass classification is more important in practice than optimal worst-case complexity.

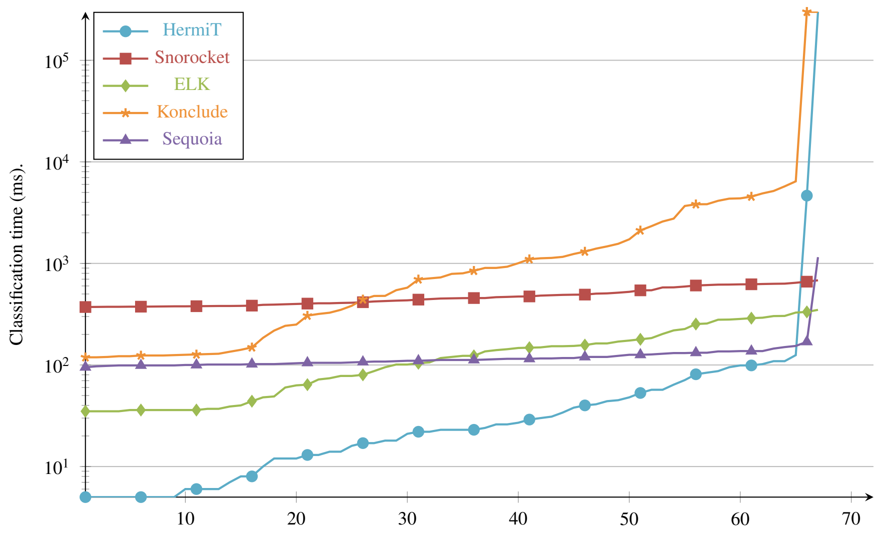


Figure 14: Classification Times for All OWL 2 EL ontologies supported by ELK and Snorocket.

To test empirically the pay-as-you-go behaviour of Sequoia, we compared its performance against specialised reasoners for the OWL 2 EL profile [53], namely ELK 0.4.0, and Snorocket 2.8.1. For reference, we also included in this test the reasoners HermiT and Konclude. The dataset for this experiment was obtained as follows: first, we removed from each ontology all axioms violating a restriction of the OWL 2 EL profile; then, we discarded all empty ontologies, and finally, we eliminated all ontologies not supported by either ELK or Snorocket, as neither of these reasoners fully supports the OWL 2 EL profile. The result is a corpus of 67 ontologies, out of which 52 contain nominals. The experiment follows the same methodology as our evaluation on the full corpus. Figure 14 summarises the (averaged) classification times for this corpus, sorted as in Figure 12. The performance of Sequoia is comparable with Snorocket and ELK, and it performed better than Konclude on this corpus. Sequoia is the only reasoner not designed specifically for OWL 2 EL which could classify all ontologies in the corpus. Both Konclude and HermiT time out on ontologies that Sequoia, as well as Snorocket and ELK, can easily classify. One of these ontologies (ID 104) is comparatively harder for Sequoia, but the classification time remains close to those of ELK and Snorocket. We interpret these results as evidence that the theoretical pay-as-you-go behaviour of our calculus manifests itself in practice.

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, and shown pay-as-you-go behaviour in practice. Reasoning with nominals, large ABoxes, 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.

9. Conclusion and Future Work

We have presented the first consequence-based reasoning algorithm for a DL featuring all Boolean operators, role hierarchies, inverse roles, nominals, and number restrictions. 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. Our implementation in the reasoner Sequoia currently covers OWL 2 DL, with the only exception of datatypes. Performance of our system is competitive with that of well-established DL reasoners, thanks to a number of novel optimisations; furthermore, our experiments show that Sequoia exhibits pay-as-you-go behaviour, and its strengths nicely complement those of (hyper-)tableau based reasoners.

We see many challenges for future work. First, in the aforementioned case where 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. Second, our algorithm should be extended with datatypes in order to cover all of OWL 2 DL. Third, we would like to explore the application of our calculus to other DL reasoning problems, such as role subsumption and classification. This, however, might require extending our framework so that we can represent arbitrary instances of particular roles. This is not possible in our current approach because contexts can only represent single elements of a model, not pairs. Finally, our evaluation shows that there is still plenty of room for optimisation; in particular, Sequoia still struggles with several non-Horn ontologies with number restrictions. ABoxes and nominals can also be problematic 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. Furthermore, we believe that it should be relatively straightforward to implement support for parallel and incremental reasoning, which seem particularly compatible with the underlying theoretical calculus.

Acknowledgements

This work was supported by the AIDA project (Alan Turing Institute), the SIRIUS Centre for Scalable Data Access (Research Council of Norway), and the EPSRC projects OASIS (EP/S032347/1), AnaLOG (EP/P025943/1), UK FIRES (EP/S019111/1), MaSI^3 (EP/K00607X/1), and ED^3 (EP/N014359/1).

References

[1] A. Armas Romero, B. Cuenca Grau, I. Horrocks, MORe: Modular Combination of OWL Reasoners for Ontology Classification, in: P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist (Eds.), Proc. of the 11th International Semantic Web Conference (ISWC'12), vol. 7649 of *LNCS*, Springer, Boston, 1–16, 2012.

[2] F. Baader, S. Brandt, C. Lutz, Pushing the $\mathcal{EL}$ Envelope, in: L. P. Kaelbling, A. Saffiotti (Eds.), Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI'05), Morgan Kaufmann Publishers, Edinburgh, 364–369, 2005.

[3] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

[4] F. Baader, E. Franconi, B. Hollunder, B. Nebel, 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, W. Swartout (Eds.), Proc. of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Morgan Kaufmann Publishers, San Francisco, 270–281, 1992.

[5] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, Cambridge, 2017.

[6] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

[7] L. Bachmair, H. Ganzinger, Equational Reasoning in Saturation-Based Theorem Proving, in: W. Bibel, P. Schmidt (Eds.), Automated Deduction—A Basis for Applications, Kluwer, Dordrecht, 353–397, 1998.

[8] L. Bachmair, H. Ganzinger, Resolution Theorem Proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier Science, Amsterdam, 19–99, 2001.

[9] J. P. Balhoff, The Phenoscape Knowledgebase: Tools and APIs for Computing Across Phenotypes from Evolutionary Diversity and Model Organisms, in: P. Jaiswal, R. Hoehndorf, C. N. Arighi, A. Meier (Eds.), Proceedings of the Joint International Conference on Biological Ontology and BioCreative, vol. 1747 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016.

[10] A. Bate, B. Motik, B. Cuenca Grau, F. Simančík, I. Horrocks, Extending Consequence-Based Reasoning to $\mathcal{SRIQ}$, in: C. Baral, J. P. Delgrande, F. Wolter (Eds.), Proc. of the 15th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'16), AAAI Press, Palo Alto, 187–196, 2016.

[11] A. Bate, B. Motik, B. Cuenca Grau, D. Tena Cucala, F. Simančík, I. Horrocks, Consequence-Based Reasoning for Description Logics with Disjunctions and Number Restrictions, J. Artif. Intell. Res. 63 (2018) 625–690.

[12] P. Baumgartner, U. Furbach, I. Niemelä, Hyper Tableaux, in: Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA '96), vol. 1126 of *LNCS*, Springer, Berlin, 1–17, 1996.

[13] P. Baumgartner, R. A. Schmidt, Blocking and Other Enhancements for Bottom-Up Model Generation Methods, J. Autom. Reason. 64 (2) (2020) 197–251.

[14] S. Brandt, Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and—What Else?, in: R. L. de Mántaras, L. Saitta (Eds.), Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI'04), IOS Press, Amsterdam, 298–302, 2004.

[15] B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler, Modular Reuse of Ontologies: Theory and Practice, J. Artif. Intell. Res. 31 (2008) 273–318.

[16] H. de Nivelle, R. A. Schmidt, U. Hustadt, Resolution-Based Methods for Modal Logics, Logic Journal of the IGPL 8 (2000) 265–292.

[17] H. Ganzinger, 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), IEEE Computer Society, Washington, 295–305, 1999.

[18] T. Gardiner, D. Tsarkov, I. Horrocks, Framework For an Automated Comparison of Description Logic Reasoners, in: Proc. of the 5th International Semantic Web Conference (ISWC'06), vol. 4273 of *LNCS*, Springer, Heidelberg, 654–667, 2006.

[19] L. Georgieva, U. Hustadt, R. A. Schmidt, Hyperresolution for Guarded Formulae, J. Symb. Comput. 36 (2003) 163–192.

[20] B. Glimm, I. Horrocks, B. Motik, R. Shearer, G. Stoilos, A Novel Approach to Ontology Classification, J. of Web Semant. 14 (2012) 84–101.

[21] C. Green, Theorem proving by resolution as a basis for question-answering systems, in: B. Meltzer, D. Michie (Eds.), Proc. of the 4th Annual Machine Intelligence Workshop, Edinburgh University Press, Edinburgh, 183–208, 1969.

[22] V. Haarslev, K. Hidde, R. Möller, M. Wessel, The RacerPro knowledge representation and reasoning system, J. Web Semant. 3 (2012) 267–277.

[23] J. Hladik, A tableau system for the Description Logic $\mathcal{SHIO}$, in: U. Sattler (Ed.), Contributions to the Doctoral Programme of the 2nd Int. Joint Conf. on Automated Reasoning (IJCAR'04), vol. 106 of *CEUR Workshop Proceedings*, CEUR-WS.org, 21–25, 2004.

[24] I. Horrocks, O. Kutz, U. Sattler, The Even More Irresistible $\mathcal{SROIQ}$, in: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'06), AAAI Press, Palo Alto, 57–67, 2006.

[25] I. Horrocks, U. Sattler, Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms, Artif. Intell. 160 (2004) 79–104.

[26] I. Horrocks, U. Sattler, A Tableaux Decision Procedure for $\mathcal{SHOIQ}$, in: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05), AAAI Press, Palo Alto, 448–453, 2005.

[27] U. Hustadt, B. Motik, U. 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), AAAI Press, Palo Alto, 152–162, 2004.

[28] U. Hustadt, B. Motik, U. Sattler, Deciding Expressive Description Logics in the Framework of Resolution, Inf. Comput. 206 (2008) 579–601.

[29] U. Hustadt, R. A. Schmidt, Issues of Decidability for Description Logics in the Framework of Resolution, in: R. Caferra, G. Salzer (Eds.), Selected Papers from Automated Deduction in Classical and Non-Classical Logics, vol. 1761 of *LNAI*, Springer, Berlin, 191–205, 1999.

[30] U. Hustadt, R. A. Schmidt, On the Relation of Resolution and Tableaux Proof Systems for Description Logics, in: T. Dean (Ed.), Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 110–117, 1999.

[31] U. Hustadt, R. A. Schmidt, Issues of Decidability for Description Logics in the Framework of Resolution, in: R. Caferra, G. Salzer (Eds.), Automated Deduction in Classical and Non-Classical Logics, vol. 1761 of *Lecture Notes in Artificial Intelligence*, Springer, 191–205, 2000.

[32] U. Hustadt, R. A. Schmidt, Using Resolution for Testing Modal Satisfiability and Building Models, J. Autom. Reason. 28 (2002) 205–232.

[33] N. Z. Karahroodi, V. Haarslev, A Consequence-based Algebraic Calculus for $\mathcal{SHOQ}$, in: Proceedings of the 30th International Workshop on Description Logics, vol. 1879 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[34] Y. Kazakov, Consequence-Driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies, in: C. Boutilier (Ed.), Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09), AAAI Press, Palo Alto, 2040–2045, 2009.

[35] Y. Kazakov, P. Klinov, Bridging the Gap between Tableau and Consequence-Based Reasoning, in: M. Bienvenu, M. Ortiz, R. Rosati, M. Simkus (Eds.), Proc. of the 27th Int. Workshop on Description Logics (DL'14), vol. 1193 of *CEUR Workshop Proceedings*, CEUR-WS.org, Vienna, 579–590, 2014.

[36] Y. Kazakov, M. Kroetzsch, F. 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), AAAI Press, Palo Alto, 2012.

[37] Y. Kazakov, M. Krötzsch, F. Simančík, The Incredible ELK — From Polynomial Procedures to Efficient Reasoning with $\mathcal{EL}$ Ontologies, J. Autom. Reason. 53 (2014) 1–61.

[38] Y. Kazakov, B. Motik, A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$, in: Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06), Springer, Heidelberg, 662–677, 2006.

[39] C. 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.

[40] C. Lutz, C. Areces, I. Horrocks, U. Sattler, Keys, Nominals, and Concrete Domains, LTCS-Report 02-04, Institute for Theoretical Computer Science, Dresden University of Technology, 2002.

[41] A. Metke-Jimenez, M. Lawley, Snorocket 2.0: Concrete Domains and Concurrent Classification, in: S. Bail, B. Glimm, R. S. Gonçalves, E. Jiménez-Ruiz, Y. Kazakov, N. Matentzoglu, B. Parsia (Eds.), Proc. of the 2nd Int. Workshop on OWL Reasoner Evaluation (ORE-2013), vol. 1015 of *CEUR Workshop Proceedings*, Ulm, Germany, 32–38, 2013.

[42] B. Motik, Reasoning in description logics using resolution and deductive databases, Ph.D. thesis, Karlsruhe Institute of Technology, Germany, 2006.

[43] B. Motik, KAON2 - Scalable Reasoning over Ontologies with Large Data Sets, ERCIM News 2008.

[44] B. Motik, R. Shearer, I. Horrocks, Hypertableau Reasoning for Description Logics, J. Artif. Intell. Res. 36 (2009) 165–228.

[45] L. A. Nguyen, A Tableau Method with Optimal Complexity for Deciding the Description Logic $\mathcal{SHIQ}$, in: N. T. Nguyen, T. V. Do, H. A. L. Thi (Eds.), Advanced Computational Methods for Knowledge Engineering, vol. 479 of *SCI*, Springer, Heidelberg, 331–342, 2013.

[46] L. A. Nguyen, ExpTime tableaux with global state caching for the description logic $\mathcal{SHIO}$, Neurocomputing 146 (2014) 249–263.

[47] L. A. Nguyen, J. Golińska-Pilarek, An ExpTime Tableau Method for Dealing with Nominals and Qualified Number Restrictions in Deciding the Description Logic $\mathcal{SHOQ}$, Fundam. Inform. 135 (2014) 433–449.

[48] R. Nieuwenhuis, A. Rubio, Theorem Proving with Ordering and Equality Constrained Clauses, J. of Symb. Comput. 19 (1995) 312–351.

[49] R. Nieuwenhuis, A. Rubio, Paramodulation-Based Theorem Proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier Science, Amsterdam, 371–443, 2001.

[50] A. Nonnengart, C. Weidenbach, Computing Small Clause Normal Forms, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier Science, Amsterdam, 335–367, 2001.

[51] M. Ortiz, S. Rudolph, M. Simkus, Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2, in: F. Lin, U. Sattler, M. Truszczynski (Eds.), Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'10), AAAI Press, Palo Alto, 269–279, 2010.

[52] OWL 2 Overview, OWL 2 Web Ontology Language Overview, W3C Recommendation. `http://www.w3.org/TR/owl2-overview`, accessed 02 February 2019, 2009.

[53] OWL 2 Profiles, OWL 2 Web Ontology Language Profiles, W3C Recommendation, URL `http://www.w3.org/TR/owl2-profiles/`, available at `http://www.w3.org/TR/owl2-profiles/`, 2009.

[54] R. A. Schmidt, U. Hustadt, First-Order Resolution Methods for Modal Logics, in: A. Voronkov, C. Weidenbach (Eds.), Programming Logics—Essays in Memory of Harald Ganzinger, vol. 7797 of *LNCS*, Springer, Berlin, 345–391, 2013.

[55] S. Schulz, Simple and Efficient Clause Subsumption with Feature Vector Indexing, in: M. P. Bonacina, M. E. Stickel (Eds.), Automated Reasoning and Mathematics - Essays in Memory of William W. McCune, vol. 7788 of *Lecture Notes in Computer Science*, Springer, Berlin, 45–67, 2013.

[56] R. Shearer, B. Motik, I. Horrocks, HermiT: A Highly-Efficient OWL Reasoner, in: Proc. of the 5th Int. Workshop on OWL Experiences and Directions (OWLED'08), vol. 432 of *CEUR Workshop Proceedings*, CEUR-WS.org, 26–27, 2008.

[57] F. Simančík, Elimination of Complex RIAs without Automata, in: Y. Kazakov, D. Lembo, F. Wolter (Eds.), Proc. of the 25th Int. Workshop on Description Logics (DL'12), vol. 846 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2012.

[58] F. Simančík, Y. Kazakov, I. Horrocks, Consequence-Based Reasoning beyond Horn Ontologies, in: T. Walsh (Ed.), Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11), AAAI Press, Palo Alto, 1093–1098, 2011.

[59] F. Simančík, B. Motik, I. Horrocks, Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics, Artif. Intell. 209 (2014) 29–77.

[60] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, J. Web Semant. 5 (2007) 51–53.

[61] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, O. Consortium, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, S. Lewis, The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, Nat. Biotechnol. 25 (2007) 1251–1255.

[62] A. Steigmiller, T. Liebig, B. Glimm, Konclude: System description, J. Web Semant. 27 (2014) 78–85.

[63] G. Sutcliffe, The CADE ATP System Competition - CASC, AI Magazine 37 (2) (2016) 99–101.

[64] D. Tena Cucala, B. Cuenca Grau, I. Horrocks, Consequence-based Reasoning for Description Logics with Disjunction, Inverse Roles and Nominals, in: A. Artale, B. Glimm, R. Kontchakov (Eds.), Proceedings of the 30th International Workshop on Description Logics (DL'17), vol. 1879 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[65] D. Tena Cucala, B. Cuenca Grau, I. 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), AAAI Press, Palo Alto, 1970–1976, 2018.

[66] S. Tobies, Complexity Results and Practical Algorithms for Logics in Knowledge Representation, Ph.D. thesis, RWTH Aachen University, 2001.

[67] D. Tsarkov, I. Horrocks, FaCT++ Description Logic Reasoner: System Description, in: Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR'06), vol. 4130 of *LNAI*, Springer, Heidelberg, 292–297, 2006.

[68] J. Vlasenko, M. Daryalal, V. Haarslev, B. Jaumard, A Saturation-based Algebraic Reasoner for $\mathcal{ELQ}$, in: P. Fontaine, S. Schulz, J. Urban (Eds.), Proc. of the 5th Workshop on Practical Aspects of Automated (PAAR'16), vol. 1635 of *CEUR Workshop Proceedings*, CEUR-WS.org, 110–124, 2016.

Appendix A. Admissible Context Orders

In this section, we prove that given an a-admissible total order $\gtrdot$ on $\Sigma_f \cup \Sigma_u$, the context order $\succ$ defined in Section 4.2 is admissible with respect to $\gtrdot$.

Let $\succ_{\mathsf{a}}$ be the order induced by $\succ$ on context a-terms; it is straightforward to verify that $\succ_{\mathsf{a}}$ is a strict order. Next, we prove that $\succ$ is a strict order. Clearly, $\succ$ is irreflexive because there is no comparison of the form $t \succ t$ in the definition of $\succ$. To see that $\succ$ is transitive, let s_1, s_2, s_3 be arbitrary context terms satisfying $s_1 \succ s_2$ and $s_2 \succ s_3$; we show that $s_1 \succ s_3$. Observe that $s_1 \succ s_2$ implies one of two possibilities: either $s_1 \succ_{\mathsf{a}} s_2$, or $s_1 \succ s_2$ is a comparison in Items 4 to 7 in the definition of $\succ$. This also holds for $s_2 \succ s_3$. Therefore, we proceed by examining all these cases one by one. Furthermore, we assume that $s_3 \neq \mathsf{true}$ and $s_2 \neq y$. Indeed, if $s_3 = \mathsf{true}$, then we have $s_1 \succ s_3$ already by Item 4 of the definition, and $s_2 = y$ implies $s_3 = \mathsf{true}$.

- If $s_1 \succ_a s_2$, then s_2 is an a-term, and therefore s_3 must also be an a-term. Hence, $s_2 \succ_{\mathsf{a}} s_3$, and thus we have $s_1 \succ_{\mathsf{a}} s_3$ because $\succ_{\mathsf{a}}$ is transitive. But $\succ$ includes $\succ_{\mathsf{a}}$, so we have $s_1 \succ s_3$.
- If $s_1 \succ s_2$ is a comparison in Item 4, then $s_2 = \mathsf{true}$, which leads to a contradiction since true is minimal in $\succ$.
- If $s_1 \succ s_2$ is a comparison in Item 5, then s_1 is a context p-term A and $s_2 = x$. This implies $s_3 = y$, and $A \succ y$ by Item 5.
- If $s_1 \succ s_2$ is a comparison in Item 6 then s_1 is a context p-term A and there exists a position p such that $A|_p \succ s_2$. Note that $A|_p$ and s_2 are both context a-terms, so $A|_p \succ_{\mathsf{a}} s_2$. Thus, s_3 is also an a-term, and $s_2 \succ_{\mathsf{a}} s_3$. Therefore, $A|_p \succ_{\mathsf{a}} s_3$ since $\succ_{\mathsf{a}}$ is transitive, and thus $A \succ s_3$ by Item 6.
- If $s_1 \succ s_2$ is a comparison in Item 7 and s_3 is a context p-term, then $s_2 \succ s_3$ must also be in Item 7. But then, s_3 is obtained by replacing context a-terms in s_1 with smaller terms with respect to $\succ_{\mathsf{a}}$, and thus, $s_1 \succ s_3$ by Item 7. If s_3 is an a-term, then $s_2 \succ s_3$ must be in Item 5, or in Item 6. In the former case, $s_3 \in \{x, y\}$, and since s_1 is a context p-term, we have $s_1 \succ s_3$ by Item 5. In the latter case, there exists a position p such that $s_2|_p \succ_{\mathsf{a}} s_3$. Furthermore, s_2 is obtained by replacing terms of s_1 by smaller terms in $\succ_{\mathsf{a}}$. Therefore, we have that $s_1|_p \succeq_{\mathsf{a}} s_2|_p$. Hence, $s_1|_p \succ_{\mathsf{a}} s_3$, which implies $s_1 \succ s_3$ by Item 6.

We have shown that $\succ$ is a strict order. Now we show that it is an admissible context order with respect to $\gtrdot$.

- Conditions (1) and (2) are satisfied due to Item 4 and Item 5, and the fact that no comparison in $\succ$ is of the form $u \succ A$ for $u \in \Sigma_u$ and A a context p-term.
- Condition (3) is satisfied by the inclusion in $\succ$ of the lexicographic path order induced by $\gtrdot$, Items 1 to 3, and the fact that no comparison in $\succ$ is of the form $s \succ u$ or $u \succ s$ for $s \in \{x, y\}$.
- For condition (4), in order to see that $\succ$ is monotonic on context terms, suppose that t is a context term, and consider a-terms s_1, s_2 such that $s_1 \succ s_2$, and both $t[s_1]_p$ and $t[s_2]_p$ are context terms. Notice that we have $s_1 \succ_{\mathsf{a}} s_2$. If t is a context p-term, then, since $t[s_2]_p$ is obtained from $t[s_1]_p$ by replacing terms with smaller terms with respect to $\succ_{\mathsf{a}}$, we have $t[s_1]_p \succ t[s_2]_p$ because of Item 7. If t is a context a-term, it is easy to check by cases that t has to be a ground term, and then $t[s_1]_p \succ_{\mathsf{a}} t[s_2]_p$, since any lexicographic path order is monotonic. This implies $t[s_1]_p \succ t[s_2]_p$ since $\succ_{\mathsf{a}}$ is contained in $\succ$. Finally, observe that the order $\succ$ satisfies the subterm property due to Items 3, 5 and 6.
- To see that condition (5) is satisfied, suppose we have a comparison $A \succ s$ violating this property i.e. A is a function-free context p-term mentioning y or a constant in Σ_u, but s is not obtained by replacing a-terms in A with smaller a-terms with respect to $\succ$, and s is not an element in $\{x, y, \mathsf{true}\} \cup \Sigma_u$. Since A is a p-term, this comparison can only be of the forms in Items 4 to 7. However, $A \succ s$ cannot be a comparison in Item 4 or Item 5 since $s \notin \{x, y, \mathsf{true}\}$. If the comparison is of the form in Item 6, the fact that A mentions only variables x and y or constants of Σ_u implies that for any proper position p of A, if $A|_p \succ s$, then s must be in $\{x, y\} \cup \Sigma_u$, which also leads to a contradiction. Finally, if $A \succ s$ is of the form in Item 7, the fact A is function-free and mentions either y or a constant in Σ_u, together with the fact that s is a context p-term, imply that s must be obtained by replacing a term in A by a smaller term with respect to $\succ$, but this contradicts our main assumption.

$\square$

Appendix B. Proof of Soundness

In this section we prove Theorem 1. For this section, we say that an interpretation $\mathcal{I}$ 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 τ is a substitution with range in the domain of $\mathcal{I}$, then $\mathcal{S}\tau$ is the set obtained by applying the substitution τ 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 $\mathcal{D}$ for $\mathcal{O}$ which is sound and an expansion strategy for $\mathcal{O}$, the application of a rule from Table 2 or Table 3 to $\mathcal{D}$ yields a context structure for $\mathcal{O}$ that is also sound.*

1	$\mathcal{ALCHOIQ}^+$ atoms	$B(z_i)$	$B(x)$	$B(u)$	
		$S(x, z_i)$	$S(x, x)$		$S(x, f(x))$
		$S(z_i, x)$			$S(f(x), x)$
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)	$B(y)$	$B(x)$	$B(u)$	$B(f(x))$
		$S(x, y)$	$S(x, x)$	$S(x, u)$	$S(x, f(x))$
		$S(y, x)$		$S(u, x)$	$S(f(x), x)$
				$S(u_1, u_2)$	
4	Context a-literals (non-root)	$f(x) \bowtie y$	$f(x) \bowtie x$	$f(x) \bowtie u$	$f(x) \bowtie g(x)$
		$u \bowtie y$	$u \bowtie x$	$u_1 \bowtie u_2$	
		$x \bowtie y$	$x \bowtie x$		
		$y \bowtie y$			
5	Context atoms (root)	$B(y)$		$B(u)$	$B(f(u))$
		$S(u, y)$		$S(u_1, u_2)$	$S(u, f(u))$
		$S(y, u)$			$S(f(u), u)$
6	Context a-literals (root)	$f(u) \bowtie y$		$f(u_1) \bowtie u_2$	$f(u) \bowtie g(u)$
		$u \bowtie y$		$u_1 \bowtie u_2$	
		$y \bowtie y$			

Table B.5: Reference tables for literals, where $\bowtie \in \{\approx, \not\approx\}$.

Proof. Let O be an ontology, and let $\mathcal{D}$ be a context structure $\langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \theta \rangle$ for O that is sound. Let $C_{\mathcal{D}}$ be the set of clauses defined in Definition 6. We next show that applying an inference rule from Table 2 or Table 3 to $\mathcal{D}$ using an arbitrary expansion strategy produces a sound context structure for O. First, we show that each clause derived in an application of the inference rules is a context clause according to Definition 2. It is then straightforward to verify that such clause is in the signature of O. Finally, 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 2 and 3. The proof also uses the soundness property of hyperresolution [8]: let ω, ϕ_i, ψ_i, and γ_i for each i with $1 \leq i \leq n$ be arbitrary formulas with free variables in $\vec{x}$. Then, we have:

$$\left\{\forall \vec{x}. \left[\bigwedge_{i=1}^{n} \phi_i \rightarrow \omega\right]\right\} \cup \bigcup_{i=1}^{n} \{\forall \vec{x}. [\gamma_i \rightarrow \psi_i \vee \phi_i]\} \models \forall \vec{x} \left[\bigwedge_{i=1}^{n} \gamma_i \rightarrow \omega \vee \bigvee_{i=1}^{n} \psi_i\right] \tag{B.1}$$

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

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 B.5, 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 B.5. Atoms from Δ_i for each i with $1 \leq i \leq n$ are context literals by our assumption. Notice that literals in Δ must be of the forms in Block 1 or Block 2 of Table B.5. 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 B.5, 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 σ of literals in Block 1 or Block 2 of Table B.5 correspond to literals of the form in Block 3 or Block 4 of Table B.5. Similarly, if $v = v_r$, we have $\sigma(x) = u_1$ for some $u_1 \in \Sigma_u^O$, and looking at Block 5 of Table B.5, we see that z_i can only be mapped to $y, f(u_1)$, or u_2 for some $u_2 \in \Sigma_u^O$ (possibly equal to u_1). Once again, we can check that images by any such σ of literals in Block 1 or Block 2 of Table B.5 correspond to literals of the form in Block 3 or Block 4 of Table B.5. Finally, condition S1 follows because, by assumption, we have: $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_i \rightarrow \Delta_i \vee A_i\sigma$ for each $1 \leq i \leq n$, and since $\bigwedge_{i=1}^{n} A_i \rightarrow \Delta$ is an ontology clause of O, we have $O \wedge C_{\mathcal{D}} \models \bigwedge_{i=1}^{n} A_i\sigma \rightarrow \Delta\sigma$. 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 B.5. 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 B.5. An analogous argument proves the equivalent result in the case $v = v_r$, using Block 5 and Block 6 of Table B.5. 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: $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_1 \to \Delta_1 \vee s_1 \approx t_1$, and $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_2 \to \Delta_2 \vee s_2 \approx t_2$. 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 $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1] \approx t_2$ and $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[s_1/t_1] \approx t_2$.

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

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 B.5, 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 $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma \to \Delta \vee s \approx t_1 \vee s \approx t_2$ 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 $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma \to \Delta \vee t_1 \not\approx t_2 \vee s \approx t_2$.

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 $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_1 \to \Delta_1 \vee A$ and $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge A \wedge \Gamma_2 \to \Delta_2$, by the soundness of hyperresolution, we have $O \wedge C_{\mathcal{D}} \models \mathsf{core}_v \wedge \Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2$.

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

$$O \wedge C_{\mathcal{D}} \models B_1(x) \wedge \bigwedge_{i=1}^{n+1} S_{B_2}(x, z_i) \to \bigvee_{1 \leq i < j \leq n} z_i \approx z_j,$$

because this an ontology clause. To prove $O \wedge C_{\mathcal{D}} \models C$, let $\mathcal{I}$ be an arbitrary model of O, and let τ 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 $\mathcal{I} \not\models \Gamma_i\tau \to \Delta_i\tau$ for $i \in \{1, 2\}$. By assumption, we have $O \wedge C_{\mathcal{D}} \models \Gamma_1 \to \Delta_1 \vee B_1(o_\rho)$ and $O \wedge C_{\mathcal{D}} \models \Gamma_2 \to \Delta_2 \vee S_{B_2}(o_\rho, x)$, 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 $C_{\mathcal{D}}$, we have:

$$O \wedge C_{\mathcal{D}} \models B(o_\rho) \wedge S_{B_2}(o_\rho, z) \to \bigvee_{i=1}^{n} z \approx o_{\rho \cdot S^i}, \tag{B.2}$$

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 $O \wedge C_{\mathcal{D}} \models \mathsf{core}_w \wedge A' \to A'$ 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 $v \neq v_r$ by our assumption.

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

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

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

Due to the first of these claims we have:

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

and, by soundness of hyperresolution,

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

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

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

The argument for $w = v_r$ is analogous, albeit with a small difference: condition S2 of Definition 6 does not guarantee $O \wedge C_{\mathcal{D}} \models \mathsf{core}_w \rightarrow \mathsf{core}_v\sigma$. 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 (B.3) and therefore obtain a clause analogous to (B.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 $O \wedge C_{\mathcal{D}} \models \mathsf{core}_w \wedge A \rightarrow A$ trivially. Furthermore, edges added in an inference by this rule are not labelled with elements of Σ_f^O, 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 $\mathsf{Pr}^r(O)$ 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 6, but this is not an issue since $\mathsf{core}_v = \emptyset$.

□

Appendix C. Rewrite systems

This section introduces basic notions of rewrite systems. We borrow the notation from [6] 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.

Appendix C.1. Basic definitions

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 $s_1 \Rightarrow s_2 \in R$, $t \in$ 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 and transitive closure of $\xrightarrow{*}_R$.

We say that a term $s \in$ HU is *irreducible by a rewrite system R* if and only if there is no term $t \in$ 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 $s, t \in$ HU, we have

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

Appendix C.2. Church-Rosser rewrite systems

Let R be a rewrite system. We say that R is *terminating* if and only if there is no infinite sequence of terms $s_1, s_2, \dots$ 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 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:[6]

Lemma 3. *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 4. *If R is terminating and left-reduced, then it is Church-Rosser.*

Lemma 5. *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 conditions of Lemma 3 with respect to a rewrite system R, we say that $\succ$ *embeds* the rules of R. To conclude the section, we state and prove the following lemma, which we use in the completeness proof:

Lemma 6. *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 $\hat{R} \models t_1 \not\approx t_2$.*

[6]We refer the reader to [6] for proofs of these results.

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. □

Appendix D. Proof of Completeness

This section offers a proof of Theorem 2, which states that the calculus is complete for (multiple) concept subsumption. Our strategy consists in showing that there is a countermodel for each target subsumption that is not present in a saturated context structure. The section is structured as follows: in Appendix D.1 we offer a high-level overview of the proof; in Appendix D.2 we show how to build a fragment of the countermodel covering the "neighbourhood" of a single element t; then, in Appendix D.3 we prove a key property of these fragments: if a term in a fragment is connected to a constant via an inverse role that appears in an "at-most" restriction in the ontology, then such term is equal to a constant; finally in Appendix D.4 we show how to combine the model fragments of Appendix D.2 into a unified countermodel.

Appendix D.1. Proof Overview

Let $\mathcal{O}$ be a fixed ontology, $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \mathsf{core}, \theta \rangle$ a context structure for $\mathcal{O}$ derivable from a sound context structure for $\mathcal{O}$, and $\gg$ an a-admissible order on $\Sigma_f \cup \Sigma_u$ such that every context $v \in \mathcal{V}$ is assigned a context order $\succ_v$ that is admissible with respect to $\gg$. Suppose also that no rule in Table 2 or Table 3 can be applied to $\mathcal{D}$. Finally, suppose that the parameter Λ in the Nom rule satisfies $\Lambda \geq 2^{\tau_{\mathsf{Su}}} \cdot 2^{\tau_{\mathsf{Pr}}} \cdot \omega$, with τ_{Su} and τ_{Pr} defined as in Theorem 2.

Let $\Gamma_Q \to \Delta_Q$ be a query clause and q a context in $\mathcal{V}$ which together satisfy conditions C1 and C2 of Theorem 2. We show the contrapositive of the theorem's statement: we assume that $\Gamma_Q \to \Delta_Q \,\hat{\notin}\, \mathcal{S}_q$, and we show $\mathcal{O} \not\models \Gamma_Q \to \Delta_Q$ by constructing a Herbrand equality model R^* of $\mathcal{O}$ containing an element $a \in \Sigma_u$ which satisfies $\Gamma_Q\{x \mapsto a\}$ but not $\Delta_Q\{x \mapsto a\}$. The element a will either be a constant in $\Sigma_u^{\mathcal{O}}$ or a fresh constant $c \notin \Sigma_u^{\mathcal{O}}$. We assume $c \gg u$ for all $u \in \Sigma_u$. This is without loss of generality since c cannot appear in the context structure.

Our proof technique is similar to that used in [11] to prove completeness of the $\mathcal{ALCHIQ}^+$ CB calculus. The completeness proof of the $\mathcal{ALCHIQ}^+$ calculus constructs the model R^* piecewise and inductively. The base case considers fresh constant c and constructs a *model fragment* R_c^* that covers the "neighbourhood" of c. This construction uses the context clauses in q to ensure that R_c^* satisfies $\mathcal{O}$ and $R_c^* \not\models \Gamma_Q \to \Delta_Q$. Then, each induction step selects a term t appearing in some previous fragment, and such that t is not equal to any other term in a fragment already constructed. Next, it chooses a context v in $\mathcal{D}$ representing t, and then uses context clauses in $\mathcal{S}_v$ to build a model fragment R_t^* that satisfies $\mathcal{O}$. The fragment is then added to the general model. The construction of each model fragment R_t^* uses a pair of parameters (Γ_t, Δ_t). The parameter Γ_t is the set of atoms that already appear in previous fragments and unify with the body of some clause in $\mathcal{O}$ via a substitution mapping x to t. The parameter Δ_t consists of equality $t \approx t'$, where t' is the predecessor of t, and all the atoms that unify with the body of some clause in $\mathcal{O}$ via a substitution mapping x to the predecessor t' of t and have *not* appeared in $R_{t'}^*$. The construction ensures that R_t^* *satisfies all atoms in* Γ_t, and *none of the literals in* Δ_t. This will have a twofold effect on the global model R^*: on the one hand, it will ensure that key inequalities like $t \not\approx t'$ remain true when the union of fragments is taken; on the other hand, it will ensure that each R_t^* will include the set H_t of all atoms in R^* that unify with the body of some clause in $\mathcal{O}$ via a substitution mapping x to t. These properties suffice to ensure that the union of all fragments is a model of $\mathcal{O}$.

Unfortunately, when $\mathcal{O}$ contains constants, ensuring that the union of fragments is still a model is much harder. There are two reasons for this:

- *Non-local constraints.* In the $\mathcal{ALCHIQ}^+$ calculus, a fragment R_t^* mentions only ground terms t, t', and terms of the form $f(t)$, for t' the predecessor of t. Because of this, the pair (Γ_t, Δ_t) can be easily generated from $R_{t'}^*$, which is always constructed before R_t^*. In our calculus, however, R_t^* can mention arbitrary constants in $\Sigma_u^{\mathcal{O}}$. This makes the compatibility conditions between fragments significantly more complex.

- *Forest-like models.* The $\mathcal{ALCHIQ}^+$ calculus does not support constants. The model generated in the proof by Bate et al. consists of a single (possibly infinite) tree rooted in the fresh constant c. By contrast, our model has a cluster of "root" elements in Σ_c which can be arbitrarily interconnected amongst themselves. Furthermore, each of these root elements can spawn a (possibly infinite) tree. Finally, anonymous elements in the tree can be directly linked to arbitrary root elements by a binary predicate. Because of this, if we wish to prove that the union of fragments is still a model by ensuring (as in the $\mathcal{ALCHIQ}^+$ calculus) that for each term t all atoms in H_t appear in the fragment R_t^*, then all (possibly infinitely many) atoms of the form $S(u, s)$ or $S(s, u)$ in H_u in a model, for s an arbitrary term, would have to appear in R_u^*. This would make the construction of fragment R_u^* very hard.

To address these issues, we modify the proof of the $\mathcal{ALCHIQ}^+$ calculus in the following ways, which correspond to the two items listed above, respectively:

- After constructing the first fragment of our model, we use it to determine the set R^*_{Rt} of function-free ground atoms and equalities between constants in Σ^O_u that will hold in the global model R^*. Then, for each model fragment R^*_t we build afterwards, we define the sets Γ_t and Δ_t in a way that ensures that R^*_t will be compatible with R^*_{Rt}. This will ensure that all fragments agree on "non-local" literals.
- We construct the fragments in the order dictated by $\gtrdot$, albeit with one exception: c is chosen before the constants in Σ^O_u because this allows us to define R^*_{Rt} simultaneously with the construction of fragment R^*_c. Then, if c is found to be equal to some constant $u \in \Sigma^O_u$, we discard the fragment R^*_c.

  Furthermore, we modify the construction of model fragments so that the following condition is satisfied by the global model: each atom in H_t appears in R^*_t or is of the form $S(s,t)$ or $S(t,s)$ for some term s and appears in R^*_s together with other body atoms in the grounding of ontology clauses mapping x to t that mention S in the body. This condition suffices to guarantee that the union of model fragments is itself a model. Moreover, ensuring that model fragments satisfy this condition instead of the similar condition in the proof of the $\mathcal{ALCHIQ}^+$ calculus (i.e., each atom in H_t must appear in R^*_t) is much easier. Indeed, when $t = u$, we do not need to have each atom of the form $S(u,s)$ or $S(s,u)$ in R^*_u; instead, we can have them in R^*_s. Then, for all DL-clauses other than those of the form DL4, additional body atoms in the grounding of an ontology clause mentioning S in the body and where x has been mapped to u will be of the form $B(u)$. Such atoms can appear in context clauses in any context, so it is easy to take them into account when constructing R^*_s.

  The body atoms in groundings of ontology clauses of the form DL4 that map x to u may mention arbitrary ground terms $s_1, \dots, s_{n+1}$. In these cases, the solution described in the previous paragraph no longer works, because for each $i \in \{1, \dots, n+1\}$, we would have atom $S_{B_2}(u, s_i)$ in a different model fragment $R^*_{s_i}$. This would make it very difficult to ensure that the union of fragments is still a model of O. Instead, we use the property of saturation with respect to the Nom role to prove that each s_i is equal to a constant u_i, for $i \in \{1, \dots, n+1\}$. This makes it much easier to construct the fragment R^*_u, since all atoms of the form $S(u, u_i)$ can appear in context clauses in the same context.

To construct a fragment R^*_t we choose a context v satisfying the property that $A \to A \hat{\in} \mathcal{S}_v$ for each $A \in \Gamma_t$; this ensures that all context clauses that are necessary to construct R^*_t appear in v. The fragment R^*_t is then constructed using a variant of the standard deterministic procedure for proving correctness of resolution-based calculi [8], which uses a total order $>_t$ on the ground terms which is compatible with the context order $>_v$ assigned to the context v. The procedure grounds the context clauses of v by $\{x \mapsto t, y \mapsto t'\}$ and selects those with a body in Γ_t. The procedure then iterates through clause heads, from smaller to bigger with respect to $>_t$, starting with an empty model fragment, and adding to the fragment only those literals needed to satisfy a clause, avoiding picking literals from Δ_t. After this, the fragment is completed in a way that ensures that the properties of equality are satisfied. We illustrate this process in the following example.

Example 3. *Let O_3 be the ontology in Figure 6. We show how to construct a model showing $O_3 \not\models A(x) \to \bot$, for $\mathcal{D}$ the saturation of the context structure in Figure 7 using the rules of our calculus with the cautious expansion strategy. The saturated context v_A contains clause (93) and clauses (97) through (99), while clauses (94) through (96) are removed by* Elim *because of clause (99). It contains also clause $\top \to C(o)$, which is propagated from v_B via* Pred, *and clause $\top \to D(o) \vee F(o)$, propagated from v_r via r-*Pred *using $\top \to C(o)$ as side premise, after $C(o) \to C(o)$ in v_r has been resolved with ontology clause (87). We also assume that all contexts use the context order defined in Appendix A. One can easily check that v_A satisfies the conditions of the completeness theorem with respect to our query $A(x) \to \bot$.*

We first consider constant c. Since we would like to disprove $A(c) \to \bot$, we need R^*_c to satisfy $A(c)$, so we define $\Gamma_c = \{A(c)\}$. Since the head of the clause we wish to disprove is empty, we do not need to forbid any atom from appearing in R^*_c, so we let $\Delta_c = \{\}$. To define a total order $>_c$ on ground terms, we use the lexicographic order on ground terms induced by: $f > R > F > D > C > B > A > c > o >$ true, *which is compatible with the context order for v_A. We then ground the clauses of v_A via $\{x \mapsto c\}$, which results in the following clauses, ordered from the smallest head to biggest:*

$$\top \to A(c), \quad \top \to C(o), \quad \top \to D(o) \vee F(o), \quad \top \to F(c), \quad \top \to B(f(c)), \quad \top \to R(c, f(c)).$$

The fragment R^*_c consists of the atoms in $\{A(c), C(o), F(o), F(c), B(f(c)), R(c, f(c))\}$ plus all necessary atoms to satisfy the properties of equality, such as true $\approx$ true, *$A(c) \approx C(o)$, etc. Fragment R^*_c determines the set R^*_{Rt} of function-free ground atoms and equalities with constants in Σ^O_u that will hold in R^*. In our case, this is $\{C(o), F(o)\}$ union all trivial equalities.*

Next, we construct the fragment R^*_o. We define $\Gamma_o = \{C(o), F(o)\}$, because we already know that these atoms must hold in R^*. Then, Δ_o contains all other function-free ground atoms and equalities with constants of Σ^O_u that do not hold in R^*_{Rt}, including, for instance, $D(o)$ and $E(o)$. We then repeat the procedure above with the clauses of v_r. It is not hard to verify that, in the saturation of v_r, the only clauses with bodies in Γ_o are $C(o) \to C(o)$, $C(o) \to D(o) \vee F(o)$ (obtained from the previous via Hyper*)* *and $F(o) \to F(o)$ (obtained from clause $\top \to D(o) \vee F(o)$ in v_A via r-Succ). The constructed fragment R^*_o contains the atoms $\{C(o), F(o)\}$ plus all required atoms to satisfy the properties of equality.*

After this, we consider the next ground term mentioned in the fragments defined thus far, namely $f(c)$. We define $\Gamma_{f(c)} = \{B(f(c)), R(c, f(c)), C(o), F(o)\}$, while $\Delta_{f(c)}$ is equal to the union of $\{A(f(c)), C(f(c)), D(f(c)), E(f(c)), F(f(c)), S(c, f(c))\}$ with Δ_o and $\{f(c) \approx c, c \approx o, f(c) \approx o\}$, where the literals in the latter set would lead to a contradiction since we are treating $f(c)$ as different from c and o, and c as distinct from o. We construct the fragment $R^*_{f(c)}$ by selecting context v_B. Indeed, the grounding via $\{x \mapsto f(c), y \mapsto c\}$ of the saturated set of context clauses of v_B contains up to redundancy the clauses $B(f(c)) \to B(f(c))$, $R(c, f(c)) \to R(c, f(c))$, $C(o) \to C(o)$, and $F(o) \to F(o)$ (the latter is derived from v_r via r-Pred), as required for $\Gamma_{f(c)}$. We then follow a procedure analogous to that for v_A and we generate a model fragment that contains $\{C(o), F(o), B(f(c)), R(c, f(c)), C(g(f(c)), S(f(c), g(f(c)), g(f(c)) \approx o, S(f(c), o)\}$. Note that the new ground term appearing in the fragment, $g(f(c))$, is equal to o, so no further fragments need to be constructed.

The complete model R^* includes all fragments defined thus far plus any additional literals required to satisfy the properties of equality. It can be summarised as:

$$\{A(c), C(o), F(o), F(c), B(f(c)), R(c, f(c)), C(g(f(c)), F(g(f(c))), S(f(c), g(f(c)), S(f(c), o), g(f(c)) \approx o\}.$$

□

Appendix D.2. The Model Fragment R^*_t

This section defines, for each a-term t in the countermodel, the model fragment R^*_t which covers the "neighbourhood" of t and disproves $\Gamma_t \to \Delta_t$, which is either a grounding of target query, or expresses a compatibility condition with the fragments defined previously. The section is organised as follows: in Appendix D.2.1 we describe the input parameters required for the construction of the model fragment R^*_t and the conditions that they satisfy; in Appendix D.2.2 we define formally the neighbourhood of t and establish a total order on the terms in the neighbourhood, which is necessary for the fragment-building method; in Appendix D.2.3 we prove several preliminary lemmas on the set of clauses N_t that we will use to build the model fragment for t, using the set of context clauses in the context selected for t; in Appendix D.2.4 we construct the fragment; in Appendix D.2.5 we show that the fragment satisfies the clauses in N_t; finally, in Appendix D.2.6 we show that the model fragment refutes $\Gamma_t \to \Delta_t$.

Appendix D.2.1. Input Parameters

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

- A term t.
- A context v in $\mathcal{D}$, which is q if $t = c$ and v_r if $t \in \Sigma^O_u$.
- A conjunction of equalities Γ_t.
- A disjunction of literals Δ_t which is disjoint with Γ_t.

Furthermore, if $t \neq c$, we also assume that we are also given a Church-Rosser rewrite system R_{Rt} embedded in $\succ$ which contains only rules of the form $B(u) \Rightarrow \mathsf{true}$, $S(u_1, u_2) \Rightarrow \mathsf{true}$, or $u_1 \Rightarrow u_2$, for $u, u_1, u_2 \in \Sigma^O_u$. We define Γ_{Rt} as the set of equalities $s \approx l$ for each $s \Rightarrow l \in R_{\mathsf{Rt}}$, and we let R^*_{Rt} be the Herbrand equality interpretation induced by this system. Similarly, we assume that we are given a set of literals Δ_{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^*_{Rt} and Δ_{Rt} represent a partition on the set of function-free ground equalities built using only constants in Σ^O_u where R^*_{Rt} contains those satisfied by the model, and Δ_{Rt} the rest; Δ_{Rt} also contains all inequalities corresponding to a-equalities not in R^*_{Rt}. To enforce this, we assume that the following properties are satisfied:

L1. For any u_1, u_2 in Σ^O_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 $u \in \Sigma^O_u$ since $u \approx u \in R^*_{\mathsf{Rt}}$ always. The following preliminary definitions will help us state the additional conditions on the parameters.

Definition 10. *We define the* grounding substitution σ_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. Furthermore, Su_t is the set $\{A\sigma_t \mid A \in \mathsf{Su}(O)$ and $A\sigma_t$ is ground $\}$, Pr_t is the set $\{A\sigma_t \mid A \in \mathsf{Pr}(O)$ and $A\sigma_t$ is ground $\}$, Ref_t is the set $\{S(t, t) \mid S$ is a binary predicate $\}$, Nom_t is the set $\left\{A\sigma_t \mid A \in \{S(x, u), S(u, x)\} \text{ for some } u \in \Sigma^O_u \text{ and } S \in \Sigma^O_S\right\}$, Z_t is the set $\{A\sigma_t \approx \mathsf{true} \mid A$ is a function-free context p-term with one occurrence of y or a constant from $\Sigma^O_u\}$, and* Rt *is the set of all literals built using only* true*, a-terms in Σ^O_u, and function-free ground p-terms mentioning only constants in Σ^O_u*

For Γ_t and Δ_t, the following conditions are assumed:

L3. If $t = c$, then $\Delta_c = \Delta_Q\sigma_t$; if $t \in \Sigma_u^O$, 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 $t \in \Sigma_u^O$, then we have $t \approx u \in \Delta_t$ for every $u \in \Sigma_u^O$ with $t \gtrdot u$. If $t = f(t')$ we have $t \approx u \in \Delta_t$ for each $u \in \Sigma_u^O$, $t' \approx u \in \Delta_t$ for each $u \in \Sigma_u^O$ with $t' \gtrdot u$, and $t \approx t' \in \Delta_t$.

Let π_t be a function which removes all elements in Δ_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 $A \to A \,\hat{\in}\, \mathcal{S}_v$.

Appendix D.2.2. Simplification Order on the Neighbourhood of t

In this section, we first define the neighbourhood of a term t, and then we define a total simplification order on it.

Definition 11. *We define the* a-neighbourhood *of each term $t \in$ HU as the set which contains term t, the term $f(t)$ for each $f \in \Sigma_f^O$, the predecessor t' of t if it exists, and every constant in Σ_u^O.*

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 12. *We define the* p-neighbourhood *of each term $t \in$ 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 $f \in \Sigma_f^O, B \in \Sigma_A^O, S \in \Sigma_S^O, \{u_1, u_2\} \subseteq \Sigma_u^O$; and also, if t' exists, the* p*-terms $S(t, t')$, $S(t', t)$, $B(t')$ for each $B \in \Sigma_A^O$, $S \in \Sigma_S^O$.*

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

For convenience, we will say that if a term s_2 is obtained from a term s_1 by replacing context a-terms in s_1 with smaller a-terms relative to $\gtrdot$, then s_2 is an a-reduction of s_1 with respect to $\gtrdot$. 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 $s_1 >_v s_2$, 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 $s_2\sigma_c \in \{c, \mathsf{true}\} \cup \Sigma_u^O$, $s_2\sigma_c \approx \mathsf{true} \in \Delta_c$, or $s_2\sigma_c$ is an 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 $s_2\sigma_t \in \{t, t', \mathsf{true}\} \cup \Sigma_u^O$, or $s_2\sigma \approx \mathsf{true} \in \Delta_t$, or $s_2\sigma_t$ is an 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 $s_1 >_v s_2$. 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 σ_t is surjective except for t' (an a-term which, if $t' \in \Sigma_u^O$, can be obtained via $y\sigma_t$ or $t'\sigma_t$), but conditions (3) and (5) of Definition 7 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), $>_v$ already defines a total order on the a-neighbourhood of t. Hence, for any totalisation of $\succ_t$ and any a-terms in the neighbourhood s_1, s_2 with $s_1 >_t s_2$, we have $s_1 >_v s_2$. Thus, for any term r in the p-neighbourhood of t and any position p, we have $r[s_1]_p >_v r[s_2]_p$ due to condition (4); by definition of $\succ_t$, this implies $t[s_1]_p \succ_t t[s_2]_p$. To show that $\succ_t$ has the subterm property, notice that condition (4) ensures that we have $s >_v s|_p$ for any term s in the p-neighbourhood, and hence by definition of $\succ_t$, $s \succ_t s|_p$. 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 true by condition (1). We then place the elements of Σ_u^O 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 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 $>_v$, 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), function-free ground p-terms mentioning only constants in Σ_u^O 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 $>_v$, as well as the orderings between p-terms and their a-reductions w.r.t. $\gtrdot$, the terms in Q and function-free ground p-terms mentioning only constants in Σ_u^O can be ordered in any way. Finally, the remaining terms are totalised arbitrarily.

- **(Case** $t = u$ **for** $u \in \Sigma_u^O$**)** To define $>_u$, we totalise $>_u$ as follows. It is easy to see that the first element in $>_u$ is true, followed by all elements of Σ_u^O 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 Δ_{Rt}, respecting the ordering between terms in Q imposed by $>_u$, as well as the orderings between p-terms and their a-reductions w.r.t. $\gtrdot$. This is possible because by condition (5), function-free ground p-terms mentioning only constants in Σ_u^O can only be greater than their respective a-reductions w.r.t. $\gtrdot$. Thus, we ensure condition O3. To conclude, totalise the remainder of $>_t$ arbitrarily.

- **(Case** $t = f(t')$**)** We add true as the smallest element in $>_t$, followed by all elements of Σ_u^O, followed by t' (unless t' already in Σ_u^O), 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 Δ_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 is a function-free context p-term mentioning variable y or a constant in Σ_u^O, and condition (5) requires that such terms 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.

Appendix D.2.3. Grounding of clauses

Let us write the clauses in N_t as $\{C^1, \dots, C^n\}$. 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 7 (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 $\{t, t'\} \cup \Sigma_u^O$. Then, there exits a clause $\Gamma \to \Delta_1 \vee \Delta_2$ in $\mathcal{S}_v$ such that*

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

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 $L >_v L'$ for some $L' \in \Delta_2$. Due to the form of Δ_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 7 requires L' must be a reduction of L, or an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u^O$. But this contradicts our assumption on the form of L^i.

- L is an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u^O$. But then condition (5) of Definition 7 requires L' must also be an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u^O$, 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 7 requires L' can only be an a-equality or inequality between terms in $\{x, y\} \cup \Sigma_u^O$, 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 Δ_2 contains more than one literal. The following lemma shows that this can only happen if $t = f(u)$ for some $f \in \Sigma_f^O$ and $u \in \Sigma_u^O$; furthermore, literals in Δ_2 can only differ in replacements of term y by u and vice versa.

Lemma 8. *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 $u \in \Sigma_u^O$ and $f \in \Sigma_f^O$; 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 σ_t are t' and $f(t)$. If $s_2 = u$ for $u \in \Sigma_u^O$, by hypothesis we have that $u = t$. However, since x is a context term, $v \neq v_r$, so $t \notin \Sigma_u^O$, 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 $u \in \Sigma_u^O$, $f \in \Sigma_u^O$. 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 Σ_u^O. 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 $t \in \Sigma_u^O$, and here $t \notin \Sigma_u^O$ because it is of the form $f(t')$ for some $f \in \Sigma_f^O$. Thus, it can only be the case that $s_2 \in \Sigma_u^O$; however, the fact that $s_1\sigma_t = s_2\sigma_t$ ensures $t' \in \Sigma_u^O$ 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 Σ_u^O, 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 $C' \mathrel{\hat{\in}} \mathcal{S}_v$ 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 [8]. 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 10. This suffices for our purposes since, as Lemma 9 proves, if $C' \mathrel{\hat{\in}} \mathcal{S}_v$, then $C'' \mathrel{\hat{\in}} N_t$.

Lemma 9. *For any clause* $\Gamma \to \Delta \mathrel{\hat{\in}} \mathcal{S}_v$ *such that* $\Gamma\sigma_t \subseteq \Gamma_t$ *and* $\Gamma\sigma_t \to \Delta\sigma_t$ *is ground, we have that* $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$.

Proof. If clause $\Gamma \to \Delta \mathrel{\hat{\in}} \mathcal{S}_v$ 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 $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$.
- The literals $l \approx r$ and $l \not\approx r$ are in Δ, 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 $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$.
- 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 $\Gamma\sigma_t \to \Delta\sigma_t \mathrel{\hat{\in}} N_t$.

□

Lemma 10. *Let $\mathcal{I}$ be an arbitrary interpretation of 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 Δ cannot contain a tautology, so $C \mathrel{\hat{\in}} N_t$ 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 11. *If there is a clause* $\Gamma \to \Delta \vee l \not\approx l \in N_t$, *then* $\Gamma \to \Delta \mathrel{\hat{\in}} N_t$.

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 $u \in \Sigma_u^O$, Lemma 8 ensures that Δ'_2 is of the form $l' \not\approx l'$ with $l'\sigma_t = l$. If $t = u$ for some $u \in \Sigma_u^O$, 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 8 then ensures that Δ'_2 is of the form $l' \not\approx l'$ with $l'\sigma_t = l$. Since rule Ineq is not applicable to $\Gamma' \to \Delta'_1 \vee l' \not\approx l'$, we have that $\Gamma' \to \Delta'_1 \mathrel{\hat{\in}} \mathcal{S}_v$; by Lemma 9, this implies $\Gamma \to \Delta \mathrel{\hat{\in}} N_t$. □

Appendix D.2.4. 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, all atoms in Γ_t, and no literal in Δ_t, we prove some properties of the series of the rewrite systems R_t^i. Lemmas 12 to 14 follow simply from the definition of R_t^*:

Lemma 12 (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 3. 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 13 (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 12, Lemma 6 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 14 (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 Δ^i. If L is an inequality, then $(R_t^{i-1})^* \not\models L$ implies R_t^* by Lemma 13. 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 12, Lemma 6 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 6 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 $R_t^* \not\models l \approx r$. □

Appendix D.2.5. 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 15 to 17.

Lemma 15. *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 11, there exists a clause of the form $\Gamma' \to \Delta' \vee l' \not\approx l'$ in $\mathcal{S}_v$. Since Ineq is not applicable, we have $\Gamma' \to \Delta' \hat{\in} \mathcal{S}_v$. Furthermore, the fact that Elim is not applicable to $\Gamma' \to \Delta' \vee l' \not\approx l' \in \mathcal{S}_v$ implies that Δ' 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 Elim is not applicable to $\Gamma' \to \Delta' \vee l' \not\approx l'$.

□

Lemma 16 (Rt-Equality Invariance). *If $t \neq c$ and u_1, u_2 are elements of Σ_u^O 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 $u_1, u_2 \in \Sigma_u^O$ 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 Σ_u^O, 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 17 (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 $u \in \Sigma_u^O$ and $t' >_t u$. If $t' \in \Sigma_u^O$, then $t' >_t u$ implies $t' \succ u$, and by condition L5 we have $t' \approx u \in \Delta_t$, so we reach a contradiction. If $t' \notin \Sigma_u^O$, 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 18 (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 13 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 $l^j \in \Sigma_u^O$, then by Lemma 16, $R_{\mathsf{Rt}}^* \models l^j \approx r^j$. Hence, there exists some $r_j \in \Sigma_u^O$ such that $l^j \approx r_j \in \Gamma_{\mathsf{Rt}}$. Hence, by condition L7, we have $\top \to l^j \approx r_j \in \mathcal{S}_v$ (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 $\Gamma_{\mathsf{Rt}} \subseteq \Gamma_t$ 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 (D.1) below, with the same properties. Otherwise, since l^j and r^j are a-terms, l^j must be of the form $f(t)$ for some $f \in \Sigma_f^O$. If $r^j \neq t' \in \Sigma_u^O$, then Lemmas 7 and 8 ensure that there exists a clause

$$\Gamma' \to \Delta' \vee l' \approx r' \in \mathcal{S}_v, \tag{D.1}$$

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 $\Delta' \not\succeq_v l' \approx r'$. We now prove that such a clause can also be found when $r^j = t' \in \Sigma_u^O$. Indeed, suppose there is no clause of the form (D.1). Then, Lemmas 7 and 8 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, $\Delta' \not\succeq_v f(x) \approx y \vee f(x) \approx u$. But since Factor is not applicable, then there exists a clause $\Gamma' \to \Delta' \vee y \not\approx u \vee f(x) \approx u \mathrel{\hat{\in}} \mathcal{S}_v$ (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 σ_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 (D.1). Hence, the existence of a clause of the form (D.1) 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 $u' \in \Sigma_u^O$ with $u \succ u'$, 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 $L \not\succeq_q u \approx u'$ due to $u \approx u' >_t \Delta_t$; otherwise $L \in \Delta_Q$, and then by condition C2 we have $L \not\succeq_q u$ and hence $L \not\succeq_q u \approx u'$. This is our clause of the form (D.1). 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 17. If $l^i \in \Sigma_u^O$, then by definition of $>_t$, we have $r^i \in \Sigma_u^O$ 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 Γ_{Rt} 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 Σ_u^O, 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 is a function-free ground context p-term that mentions only constants of Σ_u^O, the argument is analogous. If l^i is of the form $f(t)$, we proceed as in the previous case and obtain a clause

$$\Gamma \to \Delta \vee l \approx r \in \mathcal{S}_v \tag{D.2}$$

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 p-term that mentions a function or c, notice that due to the form of l^j, l^i cannot contain t', so Lemma 8 guarantees the existence of a clause of the form (D.2). In both cases, we are in the conditions of Lemma 7, so $\Delta \not\succeq_v l \approx r$.

If $t = c$ and c is irreducible, then if $l^i \in \Sigma_u^O$, then $r^i \in \Sigma_u^O$ 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 Eq. Observe that due to the form and properties of (D.1) and (D.2), we are in the conditions of the Eq rule, which ensures that there is a clause $\Gamma \wedge \Gamma' \to \Delta \vee \Delta' \vee l[r']_p \approx r \mathrel{\hat{\in}} \mathcal{S}_v$; Lemma 9 then ensures that $\Gamma\sigma_t \wedge \Gamma'\sigma_t \to \Delta^i \vee \Delta^j \vee l^i[r^j]_p \approx r^i \mathrel{\hat{\in}} N_t$; 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 10, 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 14 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 a-terms, since if they were 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 17, and it cannot be in Σ_u^O 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:

  $$\Gamma \to \Delta \vee \Delta' \vee l \approx r \in \mathcal{S}_v \tag{D.3}$$

  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 7, $\Delta \vee \Delta' \not\succeq_v l \approx r$. Furthermore, Δ' 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 $t' \notin \Sigma_u^O$ or $s^i \neq t' \in \Sigma_u^O$; in both cases Lemma 8 ensures that there is only one such literal, name it $l \approx s_1$, or $t' = u \in \Sigma_u^O$ 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 $l^i = u \in \Sigma_u^O$ and $r^i = u' \in \Sigma_u^O$ or l^i is of the form $f(t)$. In all these cases, we have shown that there exists a clause of the form (D.3) with $\Delta \vee \Delta' \not\succeq_v l \approx r$.

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 σ_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 (D.3) where Δ' 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 $u \in \Sigma_u^O$, we have $l^i \notin \Sigma_u^O$. We are in the conditions of Lemma 8, 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 7, so $\Delta \not\succ_v l \not\approx l$. By Lemma 11, we have $\Gamma \to \Delta \,\hat{\in}\, \mathcal{S}_v$, and since $L_i \vee \Delta_i >_t \Delta_i$, by Lemma 9 and Lemma 10, 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 13, 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 $l^i \in \Sigma_u^O$ since by definition of $>_t$, we have $r^i \in \Sigma_u^O$ 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 16.

□

Corollary 19. *For any clause* $\Gamma \to \Delta \,\hat{\in}\, N_t$*, we have* $R_t^* \models \Delta$*.*

Proof. The result follows trivially from Lemma 10 given that choosing $i = n$ satisfies the conditions of the theorem due to Lemma 18.

□

Appendix D.2.6. Compatibility Conditions

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^* satisfies the required compatibility conditions, that is, it satisfies all atoms in Γ_t and none of the literals in Δ_t. We start by proving this for the relevant atoms in Rt.

Lemma 20. $R_t^* \models R_{\mathsf{Rt}}^*$ *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 19. Consider $u_1 \approx u_2 \in \Delta_{\mathsf{Rt}}$, and suppose $R_t^* \models u_1 \approx u_2$. By Lemma 16, $R_t^* \models u_1 \approx u_2$ 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 Δ_{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 $R_t^* \models \Gamma_t$ and $R_t^* \not\models \Delta_t$ (see Lemma 23). However, we also need to prove this for the case when $t = c$ and c is not irreducible w.r.t. R_c, which implies refutation of the query $\Gamma_Q \to \Delta_Q$. To that effect, we prove first Lemmas 21 and 22.

Lemma 21. *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 Γ_c, A is of the form $A(x)$ for $A \in \Sigma_A^O$, 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 $u \in \Sigma_u^O$. 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 18 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 8, 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 $A >_q x \approx u$, 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 Δ_c, and this contradicts condition (2) of Definition 7. 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 a-terms only x, so the preconditions of the Eq rule are satisfied, and since this clause is not applicable, we have $\Gamma \wedge \Gamma' \to \Delta \vee A(u) \mathrel{\hat{\in}} \mathcal{S}_v$. If Δ 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 10, $(R_c^{i-1})^* \models \Delta^j$, 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 14 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 22. *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 $u \in \Sigma_u^O$. Therefore, by Lemma 8, 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 Δ'_2 is in Δ_Q, so it is of the form $B(x)$ for some $B \in \Sigma_A^O$, and by Condition C2, it cannot be the case that $B(x) >_q u$, 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 $u \succ u'$. Notice that since $c >_c u$, we have $j < i$. Thus, by Lemma 8, 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 Δ'_2 is in Δ_Q, so it is of the form $B(x)$ for some $B \in \Sigma_A^O$, and by C2, it cannot be the case that $B(x) >_q u$ or $B(x) >_q u'$. Thus, literal $u \approx u'$ is maximal in this clause. We are therefore in the conditions of the Eq rule, and since this rule is not applicable, there is a clause $\Gamma \wedge \Gamma' \to \Delta \vee \Delta_1 \vee \Delta_2 \vee \Delta'_1 \vee \Delta'_2 \vee x \approx u' \mathrel{\hat{\in}} \mathcal{S}_q$. Notice that $(\Gamma \wedge \Gamma')\,\sigma_c \subseteq \Gamma_c$, and the mapping by σ_c of the head of this clause is ground. Thus, by Lemma 9 we have $\Gamma_i \wedge \Gamma_j \to \Delta_i \vee \Delta_j \vee c \approx u' \mathrel{\hat{\in}} N_c$. Furthermore, we have that either $x \approx u >_c \Delta_i \vee \Delta_j \vee c \approx u'$, so by Lemma 10, 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 13 and Lemma 14 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 Δ_c, we have that L is of the form $B(c)$ for some $B \in \Sigma_A^O$. Since R_c is Church-Rosser, there must be a generative clause C^k of the form $\Gamma_k \to \Delta_k \vee B(u)$ (recall that p-terms in rewrite rules are irreducible w.r.t. R_c due to condition R3). Thus, by Lemma 8 there exists a clause $\Gamma'' \to \Delta''_1 \vee \Delta''_2 \vee B(u)$ 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 Eq rule, and hence there is a clause $\Gamma \wedge \Gamma'' \to \Delta \vee \Delta'' \vee B(x) \mathrel{\hat{\in}} \mathcal{S}_q$. Clearly, the image of the body by σ_c is in Γ_c, and the image of the head by σ_c is ground, so by Lemma 9, we have $\Gamma_i \wedge \Gamma_k \to \Delta_i \vee \Delta_k \mathrel{\hat{\in}} N_c$; 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 10, 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 13 and Lemma 14 ensure $(R_c^{k-1})^* \not\models \Delta_i$. Thus, we reach the desired contradiction. □

Lemma 23. *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 $\Gamma_t \to A \mathrel{\hat{\in}} N_t$, so if $t \neq c$ or c is irreducible w.r.t. R_t, the result follows from Corollary 19. Otherwise, the result follows from Lemma 21.

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 20. 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 22. 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 Δ_t by its definition.

- l and r are 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 15. If it is of the form $t \not\approx t'$, by Lemma 17 we have $R_t^* \models t \not\approx t'$, and hence by Lemma 13 $(R_t^{i-1})^* \models t \not\approx t'$, which makes C^i no longer generative. For $t \not\approx u$ (resp. $t' \not\approx u$) with $t \notin \Sigma_u^O$ (resp. $t' \notin \Sigma_u^O$), the argument is analogous. For the remaining case, $l \in \Sigma_u^O, r \in \Sigma_u^O$, 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 20. Since C^i is generative, Lemma 14 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 Rt. But then, by condition L4, $L \in \Delta_{\mathsf{Rt}}$. However, Lemma 16 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 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 a-reduction of an atom in Δ_t, and hence $l \approx \mathsf{true}$ is an atom in Rt. By condition L2, either

$l \approx \mathsf{true}$ is in Δ_{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 20, $R^*_t \models l \approx \mathsf{true}$; hence by Lemma 14 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 p-term, we have $L = l \approx \mathsf{true}$, and hence $l \approx \mathsf{true} \in \Delta_t$; since $l \approx \mathsf{true} \in \mathsf{Rt}$, 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 16.

□

Appendix D.3. Collapse of named individual-like elements into constants

In this section we prove that if an element t is connected to a constant via a role S, and the inverse of S is affected by an "at-most" restriction, then $R^*_t \models t \approx u$ for some $u \in \Sigma^O_u$. This follows from the lemma:

Lemma 24. *Suppose t is irreducible w.r.t. R_t. For any DL-clause of the form DL4 and $u \in \Sigma^O_u$ irreducible w.r.t. R_t, if $t = c$ or $t = f(t')$ for some $t' \neq u$ with $f \in \Sigma^O_f$, 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 Appendix D.3.1. Next, in Appendix D.3.2, we give preliminary definitions which we will use in the remainder of the section. In Appendix D.3.3 we prove several preliminary results that are necessary for the proof of the lemma. Finally, in Appendix D.3.4, we give the proof of Lemma 24.

Appendix D.3.1. Proof overview

Lemma 24 will be easy to prove when the depth of u (i.e. the length of its nominal label) is smaller than Λ. Indeed, if we assume $R^*_t \models S_{B_2}(u, t)$, it is not hard to show that there exists a clause $\Gamma' \to \Delta' \vee S(u, x) \in \mathcal{S}_v$ 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 Λ, we will see that saturation by the Nom rule ensures that t collapses into auxiliary constants. The crux of the proof consists therefore in showing that in cases where the 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_{ρ_i}, where ρ_i is the prefix of ρ length i. We show 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 $S_i \in \Sigma^O_S$ 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. We then prove, in Lemma 27, that for each o_i, some context in $\mathcal{D}$ will contain up to redundancy a clause of the form:

$$\Gamma_{\mathsf{Rt}} \wedge \Gamma \to \Delta_{\mathsf{Rt}} \vee \Delta \vee \Delta_{\mathsf{Eq}}, \tag{D.4}$$

where:

- $\Gamma_{\mathsf{Rt}} \subseteq \mathsf{Rt}$, $\Delta_{\mathsf{Rt}} \subseteq \mathsf{Rt}$,
- Γ contains only atoms in $\mathsf{Su}(O)$ of the form $B(x)$, $S(y, x)$, or $S(x, y)$, and Δ contains only atoms in $\mathsf{Pr}(O)$ of the form $B(y)$, $S(y, x)$, or $S(x, y)$,
- Δ_{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 $u_i, u'_j \in \Sigma^O_u$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

Furthermore, we show that $R^*_t \models \Gamma_{\mathsf{Rt}}$ and $R^*_t \not\models \Delta_{\mathsf{Rt}}$. Then, we show in Lemma 29 that if the clauses of the form in (D.4) for two different constants o_i, o_j with $1 \leq i < j \leq n$ appear in the same context, they cannot have the same Γ and Δ. Observe that Λ is greater or equal than the number of possible pairs $\langle \Gamma, \Delta \rangle$ multiplied by the total number of contexts. Using this observation and the pigeonhole principle, we conclude that there exists a clause of this form contained up to redundancy in $\mathcal{S}_v$ with $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$. Finally, we show in Lemma 31 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.

Appendix D.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 $\mathcal{V}^i, \mathcal{E}^i, \mathsf{core}^i, \mathcal{S}^i, \theta^i$; 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 $\mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \theta$. We assume that all context structures mentioned in this section are for O. Throughout this proof, it will be useful to refer to literals in a particular context structure of a derivation. To this effect, we introduce the following notion.

Definition 13. *An* occurrence of a literal *L in a derivation is a 4-tuple $\langle \mathcal{D}, v, C, L \rangle$, where $\mathcal{D}$ is a context structure in the derivation, $C \in \mathcal{S}_v$, and L appears in C.*

If C is a clause that occurs in $\mathcal{D}_i$ but not in $\mathcal{D}_{i-1}$, we say that C has been *generated* in the i-th step of the derivation. Also, if $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2$ 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 $\langle \Gamma_1, \Delta_1 \rangle$. We also introduce the notion of *selected literal* to identify the literals in context clauses that play an active role within an inference, and the notion of *generated literal* to identify the new literals introduced in conclusions of inference rules.

Definition 14. *Let C be a premise in an inference by a rule from Tables 2 and 3. 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.

Let C be a conclusion in an inference by a rule from Tables 2 and 3, 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*.*

The notion of *descendance* helps us keep track of occurrences of literals in context clauses that play an active role in deriving other literals in the context structure during the saturation procedure.

Definition 15. *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 7 and the preconditions of inference rules in Tables 2 and 3 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 ground 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 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 16 (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, λ maps nodes of T to clauses, and μ maps nodes of T to literals, which satisfies the following properties:*

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

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 ξ be an inference distinct from Elim which uses C as a premise.

Definition 17. *The* natural successor *of $(C, \langle T, \lambda, \mu \rangle)$ by ξ is the pair $(C', \langle T', \lambda', \mu' \rangle)$ defined as follows:*

- *if ξ uses* Succ *or r*-Succ, *then $C' = C$ and $\langle T', \lambda', \mu' \rangle = \langle T, \lambda, \mu \rangle$.*
- *if the selected literal in C is in Δ but not in Δ_w for any $w \in T$, we define C' as the conclusion of inference ξ, 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 Δ and appears in Δ_w for some $w \in T$, and L is non-ground, observe that by definition of $>_v$ it cannot be the case that $A >_v L$ for some ground equality A, and since L is maximal among the literals of Δ, then is maximal among the literals of Δ_w. In this case, we define C' as the conclusion of ξ, 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 the inference ξ' obtained by replacing C by C_w in ξ and removing unnecessary premises. 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 Γ, then we define C' as the conclusion of ξ, and for each $w \in T$ we replace each C_w by the conclusion of an inference ξ' obtained by replacing C is by C_w in ξ and removing unnecessary premises. 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 correspond to the conclusion of the inference and the result of an analogous inference on the ground decomposition.

Definition 18. *The* alternative successor *of $(C, \langle T, \lambda, \mu \rangle)$ by ξ is defined as the following pair $(C', \langle T', \lambda', \mu' \rangle)$:*

- *if the selected literal in C is in Δ and is of the form $u \not\approx a$ and there exists some w in T where Δ_w^{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 ξ, and define $\langle T', \lambda', \mu' \rangle$ as the following extension of $\langle T, \lambda, \mu \rangle$: since in this case C participates in ξ 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 Δ, appears in Δ_w^{Rt} for some $w \in T$, and is a ground equality, we define C' as the conclusion of ξ 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 ξ' identical to ξ except that C is replaced by $L \to L$; also, $\mu(w') = L$.*
- *if the selected literal L in C is in Γ, and appears in Γ_w^{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" an inference which selects ground atoms. Notice that for any clause C, ground decomposition of C, and inference ξ, if the natural successor is undefined, then the alternative successor is well defined.

Appendix D.3.3. Preliminary Results

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

Lemma 25. *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 Δ does not contain a tautology, then for each $j \geq i$ there exist Γ' and Δ' 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 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 Δ' cannot contain a tautology, for otherwise Δ 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 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 Δ does not contain a tautology. Since $C \hat{\in} \mathcal{S}_v^i$, there exist Γ' and Δ' 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 Δ' 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 26. *For all inference rules in Tables 2 and 3, if the premises are contained u.t.r. in $\mathcal{S}_v$ for a context v in a saturated context structure $\mathcal{D}$ for O, 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 $\Gamma_i \to \Delta_i \vee A_i\sigma \mathrel{\hat{\in}} \mathcal{S}_v$ for some $1 \leq i \leq n$. Due to the form of $A_i\sigma$, it cannot be part of any tautology. If Δ_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 $\bigwedge_{i=1}^n \Gamma'_i \to \bigvee_{i=1}^n \Delta''_i \vee \Delta\sigma \mathrel{\hat{\in}} \mathcal{S}_v$, which entails the desired claim.

- For the Eq rule, suppose $\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1 \mathrel{\hat{\in}} \mathcal{S}_v$. If Δ_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 Δ_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 Δ 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 $\Gamma' \to \Delta'' \mathrel{\hat{\in}} \mathcal{S}_v$, which implies the desired result.

- For Factor, if Δ 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 Δ_1 or Δ_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 $\Gamma'_1 \wedge \Gamma'_2 \to \Delta''_1 \vee \Delta'_2 \mathrel{\hat{\in}} \mathcal{S}_v$, 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 Δ_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-Pred rule, the argument is analogous to that for the Pred rule.

□

Next, we present a lemma which shows that at least n clauses of the form in (D.4) always exist in a derived context structure whenever an auxiliary constant of depth n is mentioned in said context structure.

Lemma 27. *Let o_ρ 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 ρ is of the form $S_1^{i_1} \cdot \dots \cdot S_n^{i_n}$.*

Let $\mathcal{D}$ be a derivation from a context structure for O with no occurrences of auxiliary constants. 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* $C_i = \Gamma_i \rightarrow \Delta_i \vee L_i$, *then* $\Gamma_i \rightarrow \Delta_i \vee [L_i] \hat{\in} \mathcal{D}$;

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* $\Gamma_i \rightarrow \Delta_i \vee S_i(o_i, x)$, *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* 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* $\Gamma_i \rightarrow \Delta_i \vee \Delta_i^{\mathsf{Rt}} \vee x \approx o_i$, *with* $L_{b_i} = x \approx o_i$, Δ_i^{Rt} *is of the form:*

$$\bigvee_{j=1}^{k} x \approx u_j \quad \vee \bigvee_{j=k+1}^{k'} y \approx u_j \qquad\qquad u_j \in \Sigma_u^O \text{ for } 1 \leq j \leq k' \tag{D.5}$$

$\Delta_i \subseteq \mathsf{Pr}_\tau \cup \mathsf{Rt}$, *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* 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 O mentions $o_{n'-1}$, and hence we have two options: (i) there exists at least one occurrence of a literal $\langle \mathcal{D}_i, v_i, C_i, L_i \rangle$ 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 Nom, in which case there exists an occurrence of a literal $\langle \mathcal{D}_i, v_i, C_i, L_i \rangle$ with $L_i = S_{n'-2}(o_{n'-2}, x)$, with C_i a premise used for the 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, $\Gamma_{i+1} \rightarrow \Delta_{i+1} \vee [L_{i+1}] \hat{\in} \mathcal{D}$ for some $1 \leq i < m$, and we consider the clause $\Gamma'_{i+1} \rightarrow \Delta'_{i+1} \vee L_{i+1} \in \mathcal{D}$ with $\Gamma'_{i+1} \subseteq \Gamma_{i+1}$, $\Delta'_{i+1} \subseteq \Delta_{i+1}$. Consider clause $\Gamma_i \rightarrow \Delta_i \vee L_i$, which we know is in $\mathcal{D}_i$. By Lemma 25, 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 Elim is applied, 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 Elim is applied 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 $\Gamma_{i+1} \rightarrow \Delta_{i+1} \vee [L_{i+1}] \hat{\in} \mathcal{D}$ implies $\Gamma_i \rightarrow \Delta_i \vee [L_i] \hat{\in} \mathcal{D}$.

Now, if $\Delta_i \vee L_i$ does not contain a tautology, suppose there is $\Gamma'_i \rightarrow \Delta'_i$ 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 Δ'_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, Δ'_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 Δ'_i does not contain L_i. We then consider the other premises of the inference rule at step i. By Lemma 25, all of them are contained up to redundancy in $\mathcal{D}$; by the application of 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 Δ'_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 $\mathcal{D}_0$ has no occurrences of auxiliary constants, 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 Pred, for otherwise the only other inference rule that can select this literal is 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 Pred inference is carried out, and the claims on the form of C_{b_i} follow from the form of the main premise in Pred. $\square$

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 28. *Let C be a clause with $C \mathrel{\hat{\in}} \mathcal{S}_v$ 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 19 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 $R_t^* \not\models \Delta_v^{\mathsf{Rt}}$. $\square$

The following lemma illustrates a sufficient condition to ensure that all elements in a "chain" of auxiliary constants are irreducible with respect to the model fragment R_t^*.

Lemma 29. *Given o_ρ, let $n = |\rho|$, and define o_i with $1 \leq i \leq n$ as in Lemma 27, and $o_0 = o$. If o_ρ 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 Δ' mentions o_n. We are in the conditions of Lemma 27, 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 16, we have that $o_i \approx a \in \Gamma_{\mathsf{Rt}}$ for some $a \in \Sigma_u^O$. Arguing as in the proof of condition L7 in Appendix D.4.1 we can conclude that there is a clause $o_i \approx a \to o_i \approx a \mathrel{\hat{\in}} \mathcal{S}_{v_{a_i}}$, 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 Eq is not applicable to that clause and C_{a_i}, we have $\Gamma \wedge o_i \approx a \to \Delta \vee S(a, x) \mathrel{\hat{\in}} \mathcal{S}_{v_{a_i}}$.

We now consider the inferences from C_{a_i+1} to C_{b_i}. By Lemma 26, and the fact that none of the atoms in the head of C_{b_i} can be bigger than $S(a, x)$ due to Definition 7, 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 $\Delta' \not\succ_{v_{a_i}} S_i(a, x)$, where Γ' and Δ' 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 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 $e_{\rho' \cdot S_i^1}, e_{\rho' \cdot S_i^2}, \cdots$, 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 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 Eq. We now consider the inferences from C_j to $C_{b_{i+1}}$, and by Lemma 26, 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

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

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 $\Gamma^{\mathsf{Rt}}_w = \top$, $\Delta^{\mathsf{Rt}}_w = \bot$, and $\mu(w)$ is chosen arbitrarily. Consider all inferences from $C_{k'}$ to $\Gamma' \to \Delta'$ in context C_m. Since for any ground atom selected for an inference rule we have $L \to L$ 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 26, 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 14 and by Lemma 23 we have $R_t^* \not\models \Delta'\sigma_t \backslash \Delta$; furthermore, we have $R_t^* \models \Gamma'\sigma_t$ by Lemma 23, and $R_t^* \models o_i \approx a$ by hypothesis, so we can apply Lemma 28 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 (D.4) corresponding to different elements in a "chain" of auxiliary constants appear in the same context and have the same type, then the chain must collapse.

Lemma 30. *Suppose there is a clause $\Gamma \to \Delta \in \mathcal{S}_v$ mentioning a constant o_ρ such that $\Gamma\sigma_t \to \pi_t(\Delta_t\sigma_t) \in N_t$. Let $n = |\rho|$, and consider all clauses defined in this case by Lemma 27. 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_ρ is not irreducible w.r.t. R_t^*.*

Proof. To prove the lemma, consider the biggest constant o_j in Σ_u^O for which this occurs. By property P3, we have that there exist $\Gamma_{b_i} \to \Delta_{b_i} \vee x \approx o_i$ and $\Gamma_{b_j} \to \Delta_{b_j} \vee x \approx o_j$ 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 $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee f(x) \approx o_j \vee \Delta^{\mathsf{Rt}}_{j+1}$, where $\Delta^{\mathsf{Rt}}_{j+1} = \Delta_{b_j}\{x \mapsto f(x), y \mapsto x\}$, be the conclusion of the inference by 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 $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee [f(x) \approx o_j] \vee \Delta^{\mathsf{Rt}}_{j+1} \hat{\in} \mathcal{D}$.

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 Pred rule with C_{b_i} as the main premise, so by Lemma 26, there exists a clause $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee \Delta^{\mathsf{Rt}}_i \hat{\in} \mathcal{S}_w$, with $\Delta^{\mathsf{Rt}}_i = \Delta_{b_i} \backslash \mathsf{Pr}_\tau \vee x \approx o_i\{x \mapsto f(x), y \mapsto x\}$. The head of this clause cannot contain a tautology and must contain at least one literal of Δ^{Rt}_i, due to its form and the fact that $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee [f(x) \approx o_j] \vee \Delta^{\mathsf{Rt}}_{j+1} \hat{\in} \mathcal{D}$. Notice that Δ^{Rt}_i must be of the form $f(x) \approx u_1 \vee \cdots \vee f(x) \approx u_l$, with $u_k \in \Sigma_u^O$ for $1 \leq k \leq l$.

Consider the inference steps from b_{j+1} to d_j, among which there is an inference step where literal $f(x) \approx o_j$ is selected. Indeed, we know that 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 $\Gamma_{d_j} \to \Delta_{d_j} \vee [f(x) \approx o_j] \hat{\in} \mathcal{S}_w$. Consider the premises used in these inference steps and $\Gamma_{b_{j+1}} \to \Delta_{b_{j+1}} \vee \Delta^{\mathsf{Rt}}_i$; by Lemma 26, 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 29, 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 the 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 $u_1 \succ o_j$, we contradict the initial hypothesis that o_j is the biggest constant for which the conditions of the lemma hold, and if $o_j \succ u_1$, then by Lemma 29, we have that o_ρ is not irreducible.

□

Appendix D.3.4. Main proof

In this section we use the auxiliary lemmas from Appendix D.3.3 to show that Lemma 24 holds in the case $|\rho| = \Lambda$, and then we use this result to prove Lemma 24 in general.

Lemma 31. *For any DL-clause of the form DL4 and any $o_\rho \in \Sigma_u^O$ 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 $f \in \Sigma_f^O$, we have that $R_t^* \not\models S_{B_2}(o_\rho, t)$.*

Proof. We use proof by contradiction. Consider the greatest constant o_ρ for which this occurs. Since both t and o_ρ are irreducible, there exists a generative clause $\Gamma^i \to \Delta^i \vee S(o_\rho, t)$. This implies the existence of 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 $\Delta'_2\sigma_t = S_{B_2}(o_\rho, t)$. Now, notice that we are in the conditions of Lemma 7 so $\Delta'_1 \not\succ_v \Delta'_2$. Furthermore, since $t' \neq o_\rho$, Lemma 8 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 27, so there exist Λ clauses $C_{b_1}, \cdots, C_{b_\Lambda}$ satisfying the properties of that lemma contained u.t.r. in $\mathcal{D}$. Furthermore, since o_ρ is not irreducible, we are in the conditions of Lemma 27; this, in turn, allows us to apply Lemma 30, so we have that no more than $2^{\tau_{\mathsf{Su}}} \cdot 2^{\tau_{\mathsf{Pr}}}$ clauses can be contained u.t.r. in each context, so by the pigeonhole principle, there exist $2^{\tau_{\mathsf{Su}}} \cdot 2^{\tau_{\mathsf{Pr}}}$ 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 $\Gamma_b \to \Delta_b \vee x \approx u_1 \vee \cdots \vee x \approx u_n \vee y \approx u'_1 \vee \cdots \vee y \approx u'_{n'}$, with $u_l \in \Sigma^O_u$ for $1 \leq l \leq n$, and $u'_l \in \Sigma^O_u$ 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, Γ^{Rt}_w contains all ground atoms not in the body of Γ', Δ^{Rt}_w contains all ground atoms not in the head of Δ', and $\mu(w)$ is chosen arbitrarily from Δ_w. Consider all inferences from C_b to $\Gamma' \to \Delta'_1 \vee S_{B_2}(o_\rho, x)$ in the derivation of $\mathcal{D}$. Since for any ground atom selected for an inference rule we have $L \to L$ contained u.t.r. in the corresponding context, we can use the inferences from C_b to $\Gamma' \to \Delta'_1 \vee S_{B_2}(o_\rho, x)$ 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 Γ^{Rt}_w or Δ^{Rt}_w is selected, in which case we consider the alternative successor. By Lemma 26, we obtain that all clauses in the resulting ground decomposition for $\Gamma' \to \Delta'_1 \vee S_{B_2}(o_\rho, x)$ are contained u.t.r. in $\mathcal{S}_v$.

By Lemma 28 we conclude that every ground literal in Γ_b not in Γ' is satisfied by R^*_t, and every ground literal in Δ_b not in Δ'_1 is satisfied by R^*_t. Furthermore, if L is a non-ground literal in Γ_b, it is in Su_τ, and hence we have $L \in \Gamma'$, because we chose C_b to have the same type as $\Gamma' \to \Delta'_1 \vee S_{B_2}(o_\rho, x)$, and since $R^*_t \models \Gamma'\sigma_t$ by Lemma 23, we have $R^*_t \models L\sigma_t$. Similarly, if L is a non-ground literal in Δ_b, and since C_b is a premise for a Pred rule, we have $L \in \mathsf{Pr}_\tau$, so we have $L \in \Delta'_1$ since C_b has the same type as $\Gamma' \to \Delta'_1 \vee S_{B_2}(o_\rho, x)$. Thus, the fact that $R^*_t \not\models \Delta^i$ by Lemma 14 and $R^*_t \not\models \Delta'_1\sigma_t \backslash \Delta^i$ by Lemma 23 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 24 in the general case.

Lemma 24. *Suppose t is irreducible w.r.t. R_t. For any DL-clause of the form DL4 and $u \in \Sigma^O_u$ irreducible w.r.t. R_t, if $t = c$ or $t = f(t')$ for some $t' \neq u$ with $f \in \Sigma^O_f$, 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 14. Notice that since $t = c$, we are in the conditions of Lemma 8 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 $A \in \Delta'' >_v B_1(u)$, 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 $\Delta'_1 \not\succ_w B_1(u)$, with w the context used to construct the fragment for t'. If $t \neq c$, then since u is irreducible w.r.t. R_t, Lemma 16 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 $B_1(u) \to B_1(u) \hat{\in} \mathcal{S}_v$. Notice once again that by condition L6, $B_1(u) \to B_1(u) \hat{\in} \mathcal{S}_v$ implies $B_1(u) \to B_1(u) \in \mathcal{S}_v$ or $\top \to B_1(u) \in \mathcal{S}_v$. 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 $u \in \Sigma^O_u$ 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 7 so $\Delta'_1 \not\succ_v \Delta'_2$. Furthermore, since $t' \neq u$, Lemma 8 ensures that $\Delta'_2 = S_{B_2}(u, x)$.

Suppose the depth of u is smaller than Λ. Since the context structure is saturated and the preconditions of the rule 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 9, 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 Δ_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 14 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 $u \in \Sigma^O_u$ since $t \neq u$ and t is irreducible w.r.t. R^*_t by hypothesis. But then we contradict Lemma 10, and hence we conclude the proof.

Finally, if the depth of u is equal to Λ, then the result follows by Lemma 31. □

Appendix D.4. The Composite Model

In this section, we show how to combine the fragments into a single interpretation R^* and then prove that R^* is a model of O that does not satisfy the target query $\Gamma_Q \to \Delta_Q$. In Appendix D.4.1, we describe the order in which we construct the model fragments; furthermore, for each fragment, we give the parameters required to construct it and show that they satisfy the prerequisites from Appendix D.2.1. In Appendix D.4.2, we prove all relevant properties of the model, including the claims that it satisfies O and that it disproves the target query.

Appendix D.4.1. Order of Construction of Model Fragments

The first fragment we build is R^*_c. We then build the rest of the fragments using structural induction on the order dictated by $\succ$. Thus, for every $u \in \Sigma^O_u$, we construct the fragment R^*_u assuming that R^*_c and all fragments $R^*_{u'}$ for $u' \in \Sigma^O_u$ with $u \succ u'$ already exist and satisfy all lemmas from the previous sections. Similarly, for every term t and $f \in \Sigma^O_f$, we define the fragment $R^*_{f(t)}$ assuming that the fragment R^*_t has already been defined and satisfies all lemmas from the previous sections.

Given a term t, if $t \neq c$, let R_{Rt} be the rewrite system consisting of the rewrite rules $l \Rightarrow s$ for each $l \Rightarrow s \in R_c$ such that l, s are both in Σ^O_u, or l is a function-free ground p-term that mentions only constants in Σ^O_u and $s = \mathsf{true}$. We then define Γ_{Rt} as the set of equalities corresponding to rules in R_c, and Δ_{Rt} as the subset of Rt which is satisfied by R^*_{Rt}. Next, we define inductively the context $v \in \mathcal{V}$, the set of equalities Γ_t, and the set of literals Δ_t used as parameters in the construction of R^*_t:

- For c, we define $v = q$, $\Gamma_c = \Gamma_Q\sigma_c$ and $\Delta_c = \Delta_Q\sigma_c$.
- For each $t \in \Sigma^O_u$, if $R^*_c \not\models t \approx u$ for some $u \in \Sigma^O_u$ such that $t \succ u$, then define $v = v_r$, $\Gamma_t = \Gamma_{\mathsf{Rt}}$, 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 $f \in \Sigma^O_f$. 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 $\Gamma^i \rightarrow \Delta^i \vee l^i \approx r^i$; the fact that t is irreducible implies that if $t' = g(t'')$ for some $g \in \Sigma^O_f$, then L^i does not contain t''. Let w be the context selected for t'; since L^i contains $f(t')$ for some $f \in \Sigma^O_f$, we are in the conditions of Lemma 7 as well as Lemma 8, and hence there exists a clause $\Gamma \rightarrow \Delta_1 \vee l \approx r$ 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 Succ rule is not applicable to $\Gamma \rightarrow \Delta_1 \vee l \approx r$, there must be a context v such that $\langle w, v\rangle$ is labelled by f, we have $A \rightarrow A \hat{\in} \mathcal{S}_v$ for each $A \in K_2$ (hence $l \approx r \rightarrow l \approx r \hat{\in} \mathcal{S}_v$), and $\mathsf{core}_v \subseteq K_1$, for K_1 and K_2 defined as in the Succ rule. We then define $\Gamma_t = R^*_t \cap \mathsf{Su}_t$, and $\Delta_t = \Delta_{\mathsf{Rt}} \cup \left(\mathsf{Pr}_t \backslash R^*_{t'}\right)$.

Next we define the global rewrite system R. Let R^u_t be the set of rules in R_t of the form $u_1 \Rightarrow u_2$ for $u_1, u_2 \in \Sigma^O_u$. If c is irreducible w.r.t. R_c, then $R = R_c \cup \bigcup_{t \neq c} (R_t \backslash R^u_t)$, where the union ranges over all fragments that have been defined. If c is not irreducible w.r.t. R_c, we define $R = R^u_c \cup \bigcup_{t \neq c} (R_t \backslash R^u_t)$. We highlight two aspects of this definition: first, notice that we remove the rules in R^u_t from the general system; this is to ensure that R is left-reduced, and therefore Church-Rosser (see Lemma 34). Furthermore, this leaves R^* unaffected, since $R^*_c \models (R^u_t)^*$. The second observation is that we exclude R_c (other than rules in R^u_c) from R when c is not irreducible w.r.t. R_c; this is because we will use u to disprove the main query $\Gamma_Q \rightarrow \Delta_Q$, 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 Γ_{Rt} and Δ_{Rt} will ensure that R^*_u disproves $\Gamma_Q \rightarrow \Delta_Q$ (see Lemma 40).

We show next that during the construction of the composite model as described in the previous section, for every t, v, Γ_t, and Δ_t, the assumptions listed in Appendix D.2.1 are satisfied. Thus, we assume that one of the following is the case:

(i) $t = c$

(ii) $t \in \Sigma^O_u$ with $R^*_c \not\models t \approx u$ for every $u \in \Sigma^O_u$ with $t \succ u$

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

First, for $t \neq c$, we notice that since R_c is Church-Rosser, then R_{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, Γ_{Rt} contains $s \approx l$ for each $s \Rightarrow l \in R_{\mathsf{Rt}}$, and it is verified that Δ_{Rt} contains literals exclusively in Rt. Notice that the definition of Δ_{Rt} straightforwardly implies that conditions L1 and L2 are satisfied.

Condition L3 is trivially true by definition of Γ_t and Δ_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 $t \in \Sigma^O_u$. Otherwise, given any $L \in \Gamma_{\mathsf{Rt}}$ we have $L \in \mathsf{Su}(O)$, so $L \in \mathsf{Su}_t$. Then, if $t' = c$, we have $R^*_{t'} \models \Gamma_{\mathsf{Rt}}$ by definition of Γ_{Rt}, and otherwise by induction hypothesis we have $R^*_{t'} \models \Gamma_{\mathsf{Rt}}$ by Lemma 20. 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 20 we have $R^*_{t'} \models L$. However, by definition of Δ_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 $t \in \Sigma_u^O$ but there exists $u \in \Sigma_u^O$ with $t \gtrdot u$ such that $t \approx u \notin \Delta_t$. Since $t, u \in \Sigma_u^O$, we must have that $t \approx u \in R^*_{\mathsf{Rt}}$, for otherwise we have $t \approx u \in \Delta_{\mathsf{Rt}}$ by condition L1, and $\Delta_{\mathsf{Rt}} \subseteq \Delta_t$ by condition L4, so we obtain a contradiction. But then, by definition of R_{Rt}, $R^*_c \models t \approx u$, and since $t \gtrdot u$, 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 $u \in \Sigma_u^O$ with $t \gtrdot u$ such that $t \approx u \notin \Delta_t$. Since $t \notin \Sigma_u^O$, we have $t \approx u \notin \mathsf{Rt}$ and hence $t \approx u \notin \Delta_{\mathsf{Rt}}$; we also have $x \approx u \in \mathsf{Pr}(O)$ and hence $t \approx u \in \mathsf{Pr}_t$. Thus, by definition of Δ_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 $u \in \Sigma_u^O$ with $t' \gtrdot u$ such that $t' \approx u \notin \Delta_t$. To reach a contradiction, if $t' \in \Sigma_u^O$, we proceed exactly as in Case 1, and otherwise (i.e. $t' \notin \Sigma_u^O$), we proceed as in Case 2.a, since $y \approx u \in \mathsf{Pr}(O)$.
  - Case 2.c : $t \approx t' \notin \Delta_t$. Here, we only need to note that $x \approx y \in \mathsf{Pr}(O)$ 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 Γ'_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 Γ_c and σ_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 Δ' with respect to Δ_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 $t \in \Sigma_u^O$, 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 Δ_{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 8 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 Δ'_i and Δ_Q, by condition C2 we have that if $L >_q A'_i$, 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 $L >_q A'_i$ we have $\Delta_i >_c A_i$, and we reach a contradiction. We therefore conclude that $\Delta'_i \not>_q A'_i$. Since A'_i is ground and r-Succ is not applicable, for every $u \in \Sigma_u^O$ mentioned in A'_i, there exists an edge $\langle q, v_r, u\rangle$ between q and v. But since $\Delta_{\mathsf{Rt}} \subseteq \mathsf{Pr}^r(O)$, with these edges we are in the conditions of the r-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 $1 \le i \le n$, by Lemma 14, we have $\Gamma_i \subseteq \Gamma_c$ for each i with $1 \le i \le n$, and since Δ'_i and Δ' are either ground or in Δ_Q, by Lemma 9 we have:

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

  Furthermore, by Lemma 10, $R^*_c \models \bigvee_{i=1}^{n} \Delta_i \vee \Delta'$. However, due to the fact that $\Gamma_i \to \Delta_i \vee A_i$ is generative for each $1 \le i \le n$, by Lemma 14, we have $R^*_c \not\models \Delta_i$ for each i with $1 \le i \le n$. 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:

$$\bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C_i \to \bot \qquad \begin{array}{l} \text{with } C_i \in \Gamma_t \subseteq (R_{t'})^* \cap \mathsf{Su}_t \text{ for each } 1 \le i \le n, \\ \text{and } A_i \in \Gamma_{\mathsf{Rt}} \subseteq (R_{t'})^* \cap \mathsf{Su}_t, \text{ for each } 1 \le i \le m. \end{array} \tag{D.6}$$

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

$$\bigwedge_{i=1}^{m} A_i \wedge \bigwedge_{i=1}^{n} C'_i \to \bigvee_{i=1}^{k} L'_i \qquad \begin{array}{l} \text{with } C_i = C'_i\sigma_t \text{ for each } 1 \le i \le m, \\ \text{and } L'_i\sigma_t \in \Delta_t \text{ for each } 1 \le i \le k. \end{array} \tag{D.7}$$

By definition of Δ_t, we have that every L'_i is in $\mathsf{Pr}(O)$ or in Δ_{Rt}, or of the form $S(x, u)$ or $S(u, x)$ if $t' = u$ for some $u \in \Sigma_u^O$. Hence, this clause satisfies the preconditions for acting as main premise of a Pred rule inference. Next, we have that for each i with $1 \le i \le n$, $C_i \in \Gamma_t$, and hence $C_i \in \mathsf{Su}_t$, so C_i is either in Γ_{Rt}, or is of the form $B(t)$, $S(t, t')$, or $S(t', t)$, with $B \in \Sigma_A^O$, $S \in \Sigma_S^O$. 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 Γ_{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 $C_i \succ_c \Delta_i$. By Lemma 8 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 Δ'_i and Δ_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 $\Delta_i \succ_c C_i$, and we reach a contradiction. To uniformise the nomenclature, for this i we define $C'_i = C_i$.
- In the second case ($t' \neq c$), by condition L7, for each $C_i \in \Delta_{\mathsf{Rt}}$, there exists a clause $C_i \to C_i \,\hat{\in}\, \mathcal{S}_w$, and by condition L6 we can assume, without loss of generality, that $C_i \to C_i \in \mathcal{S}_w$. To uniformise the nomenclature, for this i we define $\Gamma'_i = \Gamma_i$ and $C'_i = C_i$.

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 17, and t is also irreducible by $R_{t'}$ due to (iii). Therefore, there exists a generative clause:

$$\Gamma_i \to \Delta_i \vee C_i \in N_{t'} \tag{D.8}$$

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

$$\Gamma'_i \to \Delta'_i \vee C'_i \tag{D.9}$$

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 7 and 8 and hence we have $\Delta'_i \not\succ_w C'_i$.

We have therefore shown that for each i with $1 \leq i \leq n$, there exists a clause of the form (D.9) with $\Gamma'_i\sigma_{t'} \subseteq \Gamma_{t'}$, $\Delta'_i\sigma_{t'} \subseteq \Delta_i \cup \Delta_{t'}$ for some Δ_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 Pred rule for i with $1 \leq i \leq n$. Furthermore, if $w = v_r$ because $t' \in \Sigma^O_u$, we define n' and C_i for each i with $n + 1 \leq i \leq n'$ as in the 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 Succ rule with respect to w. 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 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 Γ_{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 $\Delta_{n'+i} \not\succ_q C_{n'+i}$. 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 w to v.

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

$$\bigwedge_{i=1}^{n'} \Gamma'_i \wedge \bigwedge_{i=n'+1}^{n'+m} \to \bigvee_{i=1}^{n'} \Delta'_i \vee \bigvee_{i=n'+1}^{m} \Delta'_i \vee \bigvee_{i=1}^{k} L'_i\sigma \,\hat{\in}\, \mathcal{S}_w, \tag{D.10}$$

where $\sigma = \{x \mapsto f(x),\ y \mapsto x\}$ if $w \neq v_r$ 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 (D.10) is clearly ground, so by Lemma 9, we have:

$$\bigwedge_{i=1}^{n'} \Gamma_i \wedge \bigwedge_{i=n'+1}^{n'+k} \Gamma_i \to \bigvee_{i=1}^{n'} \Delta_i \vee \bigvee_{i=n'+1}^{n'+m} \Delta_i \vee \pi_{t'}\left(\bigvee_{i=1}^{k} L'_i\sigma\sigma_{t'}\right) \hat{\in}\, N_{t'} \tag{D.11}$$

For $1 \leq i \leq n$, since the clause $\Gamma_i \to \Delta_i \vee C_i$ is generative, we have by Lemma 14 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 14, $R^*_{t'} \not\models \Delta_i$. 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 Δ_t, and the fact that $R^*_{t'} \not\models \Delta_{\mathsf{Rt}}$, either by definition of Δ_{Rt} (if $t' = c$) or by Lemma 20 (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 (D.11); but this contradicts Lemma 10. 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 Γ_c. For $t \neq c$, we consider first $A \in \Gamma_{\mathsf{Rt}}$. By definition of Γ_{Rt}, there is a generating clause $\Gamma \to \Delta \vee A$ in N_c. By Lemma 8 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 Δ' and Δ_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 w; otherwise by condition L7, we have $A \to A \,\hat{\in}\, \mathcal{S}_w$, and since $A \in \Gamma_{\mathsf{Rt}}$, using condition L6 it is easy to see that, without loss of generality, we can assume $A \to A \in \mathcal{S}_w$ (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 w with a maximal literal mentioning f. Since Succ is not applicable, and $A \in \mathsf{Su}(O)$, and either $A \to A \in \mathcal{S}_w$ or $\Gamma' \to \Delta' \vee A \in \mathcal{S}_w$, we conclude $A \to A \,\hat{\in}\, \mathcal{S}_v$. 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 a clause $\Gamma' \to \Delta' \vee A'$ in $\mathcal{S}_w$ where $\Gamma'\sigma_{t'} = \Gamma$, $\Delta'\sigma_{t'} \subseteq \Delta \cup \Delta_t$, $A'\sigma_t = A$ and $\Delta \not\succ_w A$. Furthermore, since $A \in \Gamma_t$, by definition of Γ_t we have $A \in \mathsf{Su}_t$, and hence $A' \in \mathsf{Su}(O)$, so $A' \in K_2$, where K_2 is as specified in the Succ rule for w. Since Succ is not applicable, and there is an edge f connecting w and v, we have $A' \to A' \,\hat{\in}\, \mathcal{S}_v$, which implies $A \to A \,\hat{\in}\, N_t$ by Lemma 9.

Appendix D.4.2. Properties of the Model

We first 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 32. *The rewrite system R is terminating.*

Proof. By Lemma 3, 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 $\gtrdot$ to all p-function symbols in a way that makes true the smallest element of the order. Let $\triangleright$ be the lexicographic path order induced by this extension of $\gtrdot$. Since $\gtrdot$ is 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 and condition R2, we have that $l >_t r$. If l and r are 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 $f \gtrdot g$, 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 $f \gtrdot u$ for any $f \in \Sigma_f^O$ and $u \in \Sigma_u^O$, 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 $u' \in \Sigma_u^O$ with $u \gtrdot u'$, 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 17, and the fact that the definition of $>_t$ ensures that constants in Σ_u^O are the smallest a-terms. To conclude this proof, observe that if l and r are p-terms, then $l \neq$ true and $r =$ true. Since true is the smallest element in $\triangleright$ by definition, we trivially have $l \triangleright r$. □

Lemma 33. *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 12, which guarantees that R_s is Church-Rosser, and therefore left-reduced.

If $l|_p$ is a p-term, then $l|_p = l$ and due to the form of the rewrite systems R_t, we have $r' =$ true and $r =$ 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 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 $u_1 \gtrdot u_2$, for the same reasons as in the proof of Lemma 32. In case $u_1 \Rightarrow u_2$, by definition of R and Γ_{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 16 we have $R_s^* \models u_1 \approx u_2$, and since $u_1 \gtrdot u_2$, and hence $u_1 \gtrdot_s u_2$, 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 $l_p \Rightarrow r'$, then, is R_t, since it is the first fragment where elements of the form $f(t)$ can appear, according to the order of construction of fragments, 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 32 and 33 together imply the result we are looking for:

Lemma 34. *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 $t \neq \Sigma_u^O$, we can find in R_t^* the set H_t of all *hyperresolution triggers*, i.e. the irreducible atoms in R^* which unify with the body of a clause in O and a substitution σ such that $x\sigma = t$. For $t \in \Sigma_u^O$, a weaker condition holds: for any term $t \in \Sigma_u^O$, R_t^* contains $\mathsf{H}_t \backslash \mathsf{V}_t$, where V_t contains all atoms in H_t of the form $S(t, s)$ or $S(s, t)$, with s an a-term of the form $s = f(s')$ for some $f \in \Sigma_f^O$. We will later see that each atom of the forms $S(t, s)$ or $S(s, t)$ in V_t is in R_s^*. Notice that due to the form of atoms in R^*, if $t \notin \Sigma_u^O$, $\mathsf{H}_t = \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$, and if $t \in \Sigma_u^O$, $\mathsf{H}_t \backslash \mathsf{V}_t = \mathsf{Su}_t \cup \mathsf{Pr}_{f(t)} \cup \mathsf{Ref}_t \cup \mathsf{Nom}_t$.

Lemma 35 (Local Completeness)**.** *Assume that c is irreducible w.r.t. R_c. For each ground term t, each $f \in \Sigma_f^O$, 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 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 Σ_f^O, and A an atom as described in the lemma. First, notice that if $A \in \mathsf{Rt}$, then by Lemma 20, $R_t^* \models A$. For the remainder of this proof, we assume that $A \notin \mathsf{Rt}$. Since the 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 23, 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 23, 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 36. *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 Σ_u^O are the smallest 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 20, $R_t^* \models u \approx u'$. Furthermore, it cannot be the case that $R_t^* \models u' \approx u''$ for some u'' with $u' \succ u''$, since this violates Lemma 16. 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 36 and the fact that both R_c^u and R_t^u mention only elements in Σ_u^O. 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 not 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 20 we have $R_t^* \models s_n \approx s_{n+1}$. Since $s_n \succ s_{n+1}$, for both s_n and s_{n+1} are in Σ_u^O, and $s_n \succ s_{n+1}$, 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 37. *For any t, $R_t^* = (R_c^u \cup R_t \backslash R_t^u)^*$*

Proof. The corollary follows immediately from Lemma 36. □

Lemma 38 (Structural monotonicity)**.** *Let s_1 and s_2 be both DL-a-terms or DL-p-terms such that $s_1 \bowtie s_2$ is a DL-literal, with $\bowtie \in \{\approx, \not\approx\}$, and let τ be a substitution where the terms in the range of τ 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 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 37 and the fact that $R_c^u \cup R_{x\tau} \backslash R_t^u \subseteq R$. Next, we consider the case where $\bowtie$ is an inequality. Let $t = x\tau$. Since t is irreducible by R, according to the order of construction of fragments outlined at the beginning of Appendix D.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 36, they are also the normal forms of $s_1\tau$ and $s_2\tau$ w.r.t. $R^u_c \cup R_{x\tau} \backslash R^u_t$. If $t \neq c$, $R^u_c \cup R_t \backslash R^u_t \subseteq R^*$, 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 a-neighbourhood of t, they are of the form $t, t', f(t)$ for some $f \in \Sigma^O_f$, or u. Furthermore, since R_t does not contain a-terms outside of the 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 that 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 this implies that t' is also irreducible w.r.t. R by the order of construction for fragments outlined at the beginning of Appendix D.4.1 and Lemma 17. For the case $u \in \Sigma^O_u$, 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 Σ^O_u are smallest in $>_c$, this implies that u is irreducible w.r.t. R^u_c. If $t \neq c$, the fact that u is irreducible w.r.t. R_t implies that it is irreducible w.r.t. $R^u_c \cup R_t \backslash R^u_t$ due to Lemma 36, and since elements of Σ^O_u are smallest in $>_t$ and $>_c$, this implies that u is irreducible w.r.t. R^u_c. Thus, by construction of R, the fact that u is irreducible w.r.t. R^u_c implies that u is irreducible w.r.t. R. Finally, for the case $f(t)$, if $f(t)$ is irreducible by R_t, then by the order of construction of fragments, 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 17 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 39. *For each DL-clause $\Gamma \to \Delta \in \mathcal{O}$, we have $R^* \models \Gamma \to \Delta$.*

Proof. Let τ' be an arbitrary substitution such that $\Gamma\tau' \to \Delta\tau'$ is ground. Let τ be the substitution obtained by replacing each ground term in the range of τ' by its unique normal form w.r.t. R; such unique normal form exists since we know that R is Church-Rosser by Lemma 34. 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 τ 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 Σ^O_u; the last two cases occur only if $t \in \Sigma^O_u$.

If $t \in \Sigma^O_u$ and $A\tau$ is of the form $S(s, t)$ or $S(t, s)$, with $s \notin \Sigma^O_u$, and also $s \neq f(t)$, we have that $S(s, t) \Rightarrow \mathsf{true}$ appears in the fragment R_s due to Lemma 35, 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 24. 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 35, $B(t) \in R_s$ 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 Σ^O_u, or we have $t \in \Sigma^O_u$ but $A\tau$ is not of the form $S(s, t)$ or $S(t, s)$ with $s \notin \Sigma^O_u$ 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 $B(x) \in \mathsf{Su}(\mathcal{O})$. 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 $S(y, x) \in \mathsf{Su}(\mathcal{O})$. 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 35, $A_i\tau \in R_t$, so N_t contains a generative clause of the form

$$\Gamma_i \to \Delta_i \vee A_i \text{ with } A_i >_t \Delta_i \text{ and } \Gamma_i \subseteq \Gamma_t.$$

Let v be the context selected for t. If $A_i \in \mathsf{Rt}$, by Lemma 16 we have $R^*_c \models A_i$, so due to conditions L6 and L7 we can assume, without loss of generality, that $A_i \to A_i \in \mathcal{S}_v$. If $A_i \notin \mathsf{Rt}$, we are in the conditions of Lemma 7. Furthermore, if $t' \in \Sigma^O_u$ 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 $A_i \to A_i \in \mathcal{S}_v$. If $t' \notin \Sigma^O_u$ or t' does not appear in A_i, we are in the conditions of Lemma 8. 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 Hyper rule is not applicable to the ontology DL-clause and these clauses (this includes the case where $n = 0$), we have:

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

with σ the substitution where we replace t 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 9, we have

$$\bigwedge_{i=1}^{n} \Gamma_i \to \Delta\tau \vee \bigvee_{i=1}^{n} \Delta_i \mathrel{\hat{\in}} N_t$$

and by Corollary 19, we have $R_t^* \models \Delta\tau \vee \bigvee_{i=1}^{n} \Delta_i$. However, by Lemma 14 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 38, 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. We have already seen that in this case, A_i is of the form $B(t)$ or $S(s,t)$, with $t \in \Sigma_u^O$. Consider the generative clauses, their lifted forms, and atoms A_i', defined as in the previous paragraph. Since $\mathcal{D}$ is saturated and hence r-Succ is not applicable, we have that for each A_i, $A_i'' \to A_i'' \mathrel{\hat{\in}} \mathcal{S}_{v_r}$, with $A_i'' = A_i'\{x \mapsto y\}$. By Lemma 26 and Hyper, we have that $A_1'' \wedge \cdots \wedge A_n'' \to \Delta'\{x \mapsto y\} \mathrel{\hat{\in}} \mathcal{S}_{v_r}$, and since r-Pred is not applicable, by Lemma 26 we have

$$\bigwedge_{i=1}^{n} \Gamma_i' \to \Delta\sigma \vee \bigvee_{i=1}^{n} \Delta_i' \mathrel{\hat{\in}} \mathcal{S}_v,$$

and we complete the proof for this case as in the previous paragraph, concluding $R_s^* \models \Delta\tau$. Since for ontology clauses of the form DL3, DL4a, DL7, DL8, or DL9, disjunction Δ consists exclusively of equalities, by Corollary 37 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 40. $R^* \not\models \Gamma_Q \to \Delta_Q$.

Proof. Observe that Lemma 23 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 $\mathsf{Pr}(O)$ we have $B(y) \in \mathsf{Pr}(O)$, so $B(c) \in \mathsf{Pr}(f(c))$, and hence by Lemma 35, 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 20, $R_u^* \models B(u)$ and $R_u^* \not\models B'(u)$, so by Corollary 37 we conclude $R^* \models B(u)$. Finally, as argued above, $B'(u) \in \mathsf{Pr}(f(u))$, and hence by Lemma 35, 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. □

Appendix E. Soundness and Completeness for the Variant of the Calculus for Horn Ontologies

Theorem 3 (Soundness)**.** *Assume that O is Horn and K is a conjunction of atoms of the form $B(x)$ for $B \in \Sigma_A^O$. Given a context structure $\mathcal{D}$ for O which is sound for K, the application to $\mathcal{D}$ of a rule from Table 4, a rule from Table 2 other than* Hyper*, or a rule from Table 3 other than r-*Succ*, with the eager context strategy, yields a context structure for O 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 for O which is sound for K, then every model of O where the interpretation of K is not empty verifies that the interpretation of core_v for each $v \in \mathcal{V}$ is also not empty.

Lemma 41. *Let K be a conjunction of atoms of the form $B_1(x) \wedge \cdots \wedge B_n(x)$, with $B_i \in \Sigma_A^O$ for $1 \leq i \leq n$; let $\mathcal{D}$ be a context structure $\langle \mathcal{V}, \mathcal{E}, \mathsf{core}, \mathcal{S}, \theta \rangle$ for O which is sound for K, and let $v \in \mathcal{V}$ be an arbitrary context. For every model of $\mathcal{I}$ of 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 $\mathcal{D}$ as described in the lemma. 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 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 $\mathcal{I}$ of 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 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 $f \in \Sigma_f^O$. 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 Appendix B. For all inference rules other than Succ and r-Succ from Table 4, 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 $O \cup C_{\mathcal{D}}$ such that $\mathcal{I} \not\models B_1(x) \wedge \cdots \wedge B_n(x) \to \bot$, 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 $\mathsf{core}_v \to \mathsf{core}_w\{x \mapsto f(x), y \mapsto x\}$ for every edge with label $f \in \Sigma_f^O$ 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 $\top \to L$ is contained in $\mathcal{S}_{v_r}$. Furthermore, by condition Z3 we have that $\mathcal{I} \models \top \to L$ for each $L \in K_1$. Since we use the eager expansion strategy, $K_1 = \mathsf{core}_v$, so we conclude $\mathcal{I} \not\models \mathsf{core}_v \to \bot$.

To prove condition Z3 when the applied rule is r-Succ from Table 4, let $\top \to A$, with A a ground atom, be a context clause added to $\mathcal{S}_{v_r}$ by this rule; since core_{v_r} is empty, we have to prove that $\mathcal{I} \models \top \to A$. 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 $\top \to A \in \mathcal{S}_v$. By Lemma 41, we have that $\mathcal{I} \not\models \mathsf{core}_v \to \bot$, and by condition Z3, we have $\mathcal{I} \models \mathsf{core}_v \to A$, which together ensure $\mathcal{I} \models \top \to A$.

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 $f \in \Sigma_f^O$, 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 is a context clause, except in the case where Hyper applies to a non-root context with $\sigma(x) = u \in \Sigma_u^O$ and a literal of the form $S(x,u)$ or $S(u,x)$. However, looking at the form of ontology clauses in Table 1, it is easy to see that for any z_i, we have $\sigma(z_i) = x$ or $\sigma(z_i) \in \Sigma_u^O$, 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 $o \in \Sigma_u^O$, all of which are context literals. Therefore, we only have to show that for any new clause $\Gamma \to \Delta$ added to $\mathcal{D}$, $\Gamma = \top$ and Δ 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 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 $L_1 \in \mathsf{Pr}(O)$. 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 σ 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 σ being the corresponding substitution. An analogous argument applies to r-Pred. □

We conclude with a proof of Theorem 4:

Proof. Notice that if no rule from Table 4, or Table 2 except Hyper, or Table 3 except r-Succ, can be applied to $\mathcal{D}$, then no rule from Table 2 or Table 3 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 $u \in \Sigma_u^O$ and $S \in \Sigma_S^O$. Indeed, if all preconditions of the r-Succ rule from Table 2 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 $u \in \Sigma_u^O$ and $S \in \Sigma_S^O$, and we have that r-Succ from Table 4 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 3 is also not applicable.

We can therefore prove Theorem 4 using an argument analogous the proof of Theorem 2 in Appendix D, in the special case $\Gamma_Q = K$. The proof of Theorem 2 only uses the fact that $\mathcal{D}$ is saturated via r-Succ from Table 2 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 $u \in \Sigma_u^O$ and $S \in \Sigma_S^O$, in the final paragraph of Lemma 39, 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-Succ in Table 3, we can use the fact that Hyper from Table 4 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 Λ, 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 24 is always $\langle \top, \bot \rangle$, so we can use Lemma 30 to show that there can only be at most one such clause per context. Furthermore, suppose for the sake of contradiction that the length $n = |\rho|$ of the "chain" defined for o_ρ in the proof of Lemma 24 is greater than τ_{Su}^2. Since no two contexts share the same core, there is at most one context for each subset K_1 of $\mathsf{Su}(O)$. Hence, there must be 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 30 shows that o_j is equal to o_i, and hence by Lemma 29, the topmost constant o_ρ in the chain is no longer irreducible, which leads to a contradiction.

□

Appendix F. Proofs of Termination & Pay-as-you-go Behaviour

This section presents our proof of Theorems 5 to 7, stating the termination and complexity results for the algorithms of Section 5.

Theorem 5 (Termination)**.** *Algorithm 1 is terminating. Furthermore, Algorithm 1 runs in triple exponential time in the size of O if the strategy* strat *selected in Step A3 introduces at most exponentially many contexts on the size of O.*

Proof. Define k as the total number of DL-clauses in O; let m_1 be the maximum number of body atoms in a DL-clause in O, and m_2 the maximum number of literals in a context clause during execution. Observe that the former is linear in O since the number of variables is fixed. Furthermore, only a finite number of constants can be introduced by the algorithm, since the initial context structure does not contain auxiliary constants, and the depth of auxiliary constants introduced by Nom is bounded by Λ.

To prove the first claim of the theorem, simply notice that since the number of possible context clauses per context is bounded, and the number of contexts that are introduced is finite because the selected expansion strategy is admissible, the total number of context clauses in the context structure is bounded. Furthermore, 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 $2^{\tau_{\mathsf{Su}}} \cdot 2^{\tau_{\mathsf{Pr}}}$ is exponential in the size of O, and therefore by definition of Λ, the maximum length of an auxiliary constant appearing in $\mathcal{D}$ is exponential in the size of O. Hence, there are at most double exponentially many of constants in Σ_u^O introduced in 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 Σ_u^O in $\mathcal{D}$. Notice that this is at most exponential on the size of Σ_S^O and the number of constants in Σ_u^O introduced in $\mathcal{D}$; hence, it is triple exponential on the size of 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 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 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 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 O. □

Theorem 6 (Pay-as-you-go Behaviour)**.** *Let* strat *be an expansion strategy for O introducing at most exponentially many contexts. If Step A3 selects* strat*, then Algorithm 1 runs in exponential time in the size of O if this ontology is either $\mathcal{ALCHIQ}^+$, $\mathcal{ALCHOQ}$, or $\mathcal{ALCHOI}$; furthermore, if O is $\mathcal{ELH}$, the algorithm runs in polynomial time in the size of O with either the cautious or the eager strategy.*

Proof. Suppose O is an $\mathcal{ALCHIQ}^+$ ontology. We then have that if $\mathcal{D}$ has no occurrences of elements of Σ_u^O, then no inference rule will introduce a clause with elements of Σ_u^O. Hence, if we initialise the calculus according to Algorithm 1, the rule Nom will never be triggered, and therefore no constants of Σ_u^O 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 O. Furthermore, we see that since no elements of Σ_u^O 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 [11]. This also implies that if O is $\mathcal{ELH}$, then the algorithm runs in at most polynomial size in the size of O, since this result was proved for the $\mathcal{ALCHIQ}^+$ calculus in [11].

Next, consider the case where 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 O.

Now, suppose O is in $\mathcal{ALCHOQ}$ ontology. As in the previous cases, the Nom rule will not be triggered because no atom of the form $S(u, x)$ can be derived, hence the 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 O, to being polynomial in the size of O, and we have already seen that this proves that the algorithm terminates in at most exponential time in the size of O.

Finally, if $\mathcal{O}$ is an $\mathcal{ELHO}$ ontology, the 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 shows that the algorithm terminates in time at most exponential with the size of $\mathcal{O}$. □

Theorem 7 (Termination & Pay-as-you-go Behaviour)**.** *Algorithm 2 is terminating and runs in exponential time in the size of $\mathcal{O}$. Furthermore, if $\mathcal{O}$ is $\mathcal{ELHO}$ and the* Hyper *rule is applied eagerly, then Algorithm 2 runs in polynomial time in the size of $\mathcal{O}$.*

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 Λ 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 Pred and r-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 polynomial since the Nom rule is not fired. Next, observe that due to the form of the clauses in $\mathcal{O}$, we have that in each context, the first time that the Succ rule is applied with some $S(y,x) \in K_1$, for $S \in \Sigma_S^{\mathcal{O}}$, we have that K_1 is either of the form $\{S(y,x)\}$ or $\{S(y,x), B(x)\}$ for some $B \in \Sigma_A^{\mathcal{O}}$. 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 $\{S'(y,x) \mid S(y,x) \rightarrow S'(y,x) \in \mathcal{O}\}$ or $\{B(x)\} \cup \{S'(y,x) \mid S(y,x) \rightarrow S'(y,x) \in \mathcal{O}\}$. 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. □