Extending Consequence-Based Reasoning to $\mathcal{SHIQ}$

Andrew Bate, Boris Motik, Bernardo Cuenca Grau, František Simančík, and Ian Horrocks
University of Oxford
Oxford, United Kingdom
first.last@cs.ox.ac.uk

Abstract. Consequence-based calculi are a family of reasoning techniques for description logics (DLs)—knowledge representation formalisms with numerous applications. Such calculi are very effective in practice because they combine hyper tableau and resolution reasoning in a way that considerably reduces the number of inferences. Up to now, however, they were proposed for either Horn DLs (which do not support disjunctive reasoning), or for DLs without counting quantifiers. In this paper we present a novel consequence-based algorithm for $\mathcal{SHIQ}$—a DL that supports both disjunctions and counting quantifiers. Combining the two features is nontrivial since counting quantifiers require reasoning over equality, which we handle using the framework of ordered paramodulation—a state of the art method for equational theorem proving. We thus obtain a calculus that can handle an expressive DL, while still enjoying the benefits of existing calculi.

1 Introduction

Description logics (DLs) [3] are a family of knowledge representation formalisms with numerous practical applications. $\mathcal{SHIQ}$ is a particularly important DL as it provides the formal underpinning for the Web Ontology Language (OWL). DLs model a domain of interest using concepts (i.e., unary predicate symbols) and roles (i.e., binary predicate symbols). DL applications often rely on subsumption—the problem of checking logical entailment between concepts—and so the development of practical subsumption procedures for DLs such as $\mathcal{SHIQ}$ has received a lot of attention.

Most DLs are fragments of the guarded fragment [1] of first-order logic; however, $\mathcal{SHIQ}$ provides a restricted form of counting that does not fall within the guarded fragment. Moreover, most DLs, including $\mathcal{SHIQ}$, can be captured using the two-variable fragment of first-order logic with counting ($C^2$) [10], but this provides us with neither a practical nor a worst-case optimal reasoning procedure ($C^2$ and $\mathcal{SHIQ}$ are NEXPTIME- and EXPTIME-complete, respectively). Algorithms for more general logics thus do not satisfy the requirements of DL applications, and so numerous alternatives specific to DLs have been explored. Many DLs can be decided in the framework of resolution [17, 13], including $\mathcal{SHIQ}$ [12]. These procedures are usually worst-case optimal and can be practical, but, as we discuss in Section 3, in even very simple cases they can draw unnecessary inferences. Practically successful $\mathcal{SHIQ}$ reasoners, such as FaCT++ [25], HermiT [8], Pellet [24], and Racer [11], use variants of highly-optimised (hyper)tableau algorithms [5]—model-building algorithms that ensure termination by a
variant of blocking [6]. Although worst-case optimal tableau algorithms are known [9], practical implementations are typically not worst-case optimal. While generally very effective, tableau algorithms still cannot process certain ontologies; for example, the GALEN ontology\(^1\) has proved particularly challenging, mainly because tableau calculi tend to construct very large models.

A breakthrough in practical ontology reasoning came in the form of consequence-based calculi. Although not originally presented in the consequence-based framework, the algorithm for the DL EL [2] can be seen as the first such calculus. This algorithm was later reformulated and extended to Horn-$SHI\bar{Q}$ [14] and Horn-$S\bar{R}OI\bar{Q}$ [18]—DLs that support functional roles, but not disjunctive reasoning. Recently, consequence-based calculi were also developed for the DLs $ALCH$ [23] and $ALCI$ [22], which support disjunctive reasoning, but not counting. Consequence-based calculi can be seen as a combination of resolution and hypertableau (see Section 3 for details). As in resolution, they describe ontology models by systematically deriving certain ontology consequences; and as in hypertableau, the ontology axioms can be used to guide the derivation process, and to avoid drawing unnecessary inferences. Moreover, consequence-based calculi are not just refutationally complete, but can classify an ontology in a single pass, which greatly reduces the overall work. These advantages allowed the CB system to be the first to classify all of GALEN [14].

Existing consequence-based algorithms can handle either disjunctions or counting, but not both. As we discuss in detail in Section 3, it is challenging to extend these algorithms to DLs such as $SHI\bar{Q}$ that combine both kinds of construct: counting quantifiers require equality reasoning, which together with disjunctions can impose complex constraints on ontology models. Unlike in existing consequence-based calculi, these constraints cannot be captured using DLs themselves; instead, a more expressive first-order fragment is needed, which makes the reasoning process much more involved.

In Section 4 we present a consequence-based calculus for $SHI\bar{Q}$. Borrowing ideas from resolution theorem proving, we encode the required consequences using a special kind of first-order clause; and to handle equality effectively, we base our calculus on ordered paramodulation [16]—a state of the art calculus for equational theorem proving used in modern systems such as E [21] and Vampire [19]. To make the calculus efficient on $EL$, we have carefully constrained the rules so that, on $EL$ ontologies, it mimics existing $EL$ calculi. Thus, although a practical evaluation of our calculus is still pending, we believe that it is likely to perform well in practice on ‘mostly-$EL$’ ontologies due to is close relationship with existing and well-proven calculi.

2 Preliminaries

First-Order Logic. To simplify the presentation, it is common practice in equational theorem proving to assume that equality ($\approx$) is the only predicate. Atoms are thus encoded as $P(\vec{s}) \approx t$ where $t$ is a fresh constant, and a multi-sorted setting ensures that this does not change the consequences of formulae. Thus, the signature is partitioned into a set $P$ of predicate symbols (which includes $t$), and a set $F$ of function symbols.

\(^1\) http://www.opengalen.org
A term is constructed as usual using variables and the signature symbols, but function symbols cannot contain predicate symbols. Terms containing predicate symbols as their outermost symbol are called $\mathcal{P}$-terms, while all other terms are $\mathcal{F}$-terms. For example, for $P$ a predicate and $f$ a function symbol, $f(P(x))$ is malformed; $P(f(x))$ is a well-formed $\mathcal{P}$-term; and $f(x)$ is a well-formed $\mathcal{F}$-term. An (in)equality is an expression of the form $s \approx t$ ($s \not\approx t$) where $s$ and $t$ are both either $\mathcal{F}$- or $\mathcal{P}$-terms. We assume that $\approx$ and $\not\approx$ are implicitly symmetric—that is, $s \circ t$ and $t \circ s$ are one and the same expression, for $\circ \in \{\approx, \not\approx\}$. A literal is an equality or an inequality. An atom is an equality of the form $P(s) \approx t$, and we write it simply as $P(s)$ whenever it is clear from the context whether $P(s)$ is intended to be a $\mathcal{P}$-term or an atom. A clause is an expression of the form $\Gamma \rightarrow \Delta$ where $\Gamma$ is a conjunction of atoms called the body, and $\Delta$ is a disjunction of literals called the head. We often treat conjunctions and disjunctions as sets (i.e., they are unordered and without repetition) and use them in standard set operations; and we write the empty conjunction (disjunction) as $\top$ ($\bot$). For $\alpha$ a term, literal, clause, or a set thereof, we say that $\alpha$ is ground if it does not contain a variable; $\alpha$ or is the result of applying a substitution $\sigma$ to $\alpha$; and we often write substitutions as $\sigma = \{x \mapsto t_1, y \mapsto t_2, \ldots\}$. We use the standard notion of subterm positions; then, $s|_p$ is the subterm of $s$ at position $p$; moreover, $s[t|_p]$ is the term obtained from $s$ by replacing the subterm at position $p$ with $t$; finally, position $p$ is proper in $t$ if $t|_p \neq t$.

A Herbrand equality interpretation is a set of ground equalities satisfying the usual congruence properties. Satisfaction of conjunctions and disjunctions of literals in an interpretation $I$ is defined as usual; note that a disjunction of literals $\Delta$ may contain inequalities so, even if $\Delta$ is ground, $I \models I \Delta$ does not necessarily imply $I \models \Delta$.

Unless otherwise stated, (possibly indexed) letters $x$, $y$, and $z$ denote variables; $l$, $r$, $s$, and $t$ denote terms; $A$ denotes an atom or a $\mathcal{P}$-term (depending on the context); $L$ denotes a literal; $f$ and $g$ denote function symbols; $B$ denotes a unary predicate symbol; and $S$ denotes a binary predicate symbol.

Orders. A strict order $\succ$ on a universe $U$ is an irreflexive, asymmetric, and transitive relation on $U$; and $\succeq$ is the non-strict order induced by $\succ$. Order $\succ$ is total if, for all $a, b \in U$, we have $a \succ b$, $b \succ a$, or $a = b$. Given $\circ \in \{\succ, \succeq\}$, element $b \in U$, and subset $S \subseteq U$, notation $S \circ a$ abbreviates $\exists b \in S : a \circ b$. The multiset extension $\succ_{\text{mul}}$ of $\succ$ compares multisets $M$ and $N$ on $U$ such that $M \succ_{\text{mul}} N$ if and only if $M \neq N$ and, for each $n \in N \setminus M$, some $m \in M \setminus N$ exists such that $m \succ n$.

A term order $\succ$ is a strict order on the set of all terms. We extend $\succ$ to literals by identifying each $s \not\approx t$ with the multiset $\{s, s, t, t\}$ and each $s \approx t$ with the multiset $\{s, t\}$, and by comparing the result using the multiset extension of $\succ$. We use the symbol $\succ$ for the induced literal order as the intended meaning should be clear from the context.

Description Logic SHIQ. In this paper we represent $SHIQ$ ontologies as $DL$-clauses, which we define next. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be countable sets of unary and binary predicate symbols, and let $\mathcal{F}$ be a countable set of unary function symbols. $DL$-clauses are constructed using the central variable $x$ and variables $z_i$. A $DL$-$\mathcal{F}$-term has the form $x$, $z_i$, or $f(x)$ with $f \in \mathcal{F}$; a $DL$-$\mathcal{P}$-term has the form $B(z_i)$, $B(x)$, $B(f(x))$, $S(x, z_i)$, $S(z_i, x)$, $S(x, f(x))$, $S(f(x), x)$ with $B \in \mathcal{P}_1$ and $S \in \mathcal{P}_2$; and a $DL$-term is a $DL$-$\mathcal{F}$- or a $DL$-$\mathcal{P}$-term. A $DL$-literal has the form $A \approx t$ with $A$ a $DL$-$\mathcal{P}$-term (called a $DL$-atom), or $f(x) \circ g(x)$, $f(x) \circ z_i$, or $z_i \circ z_j$ with $\circ \in \{\approx, \not\approx\}$. A $DL$-clause contains
only DL-atoms of the form $B(x) \land S(x, z_i)$, and $S(z_i, x)$ in the body and only DL-literals in the head, and where each variable $z_i$ occurring in the head also occurs in the body. An ontology $\mathcal{O}$ is a finite set of DL-clauses. A query clause is a DL-clause in which all atoms are of the form $B(x)$. Given an ontology $\mathcal{O}$ and a query clause $\Gamma \rightarrow \Delta$ with free variables $\vec{x}$, the query clause entailment problem is to decide whether $\mathcal{O} \models \forall x.(\Gamma \rightarrow \Delta)$ holds; we often leave out $\forall x$ and write the latter as $\mathcal{O} \models \Gamma \rightarrow \Delta$.

$SHIQ$ ontologies are commonly written using a DL-style syntax, but we can always transform such ontologies into DL-clauses without affecting the entailment of query clauses. First, we normalise ontology axioms as shown on the left-hand side of Table 1: we encode transitivity away as in [20, 7] and then replace all complex concepts with fresh atomic ones; this process is well understood (see, e.g., [14, 22]), so we omit the details. Second, using the well-known correspondence between DLs and first-order logic [3], we translate normalised axioms to DL-clauses as shown on the right-hand side of Table 1. The standard translation of $B_1 \subseteq n S.B_2$ requires atoms $B_2(z_i)$ in clause bodies, which is not allowed in our setting. We address this issue by introducing a fresh role $S_{B_2}$ that we axiomatising as $S(y, x) \land B_2(x) \rightarrow S_{B_2}(y, x)$; this, in turn, allows us to clausify the original axiom as if it were $B_1 \subseteq n S_{B_2}$.

## 3 Motivation

In Section 3.1, we recapitulate the rationale behind consequence-based algorithms and compare them to (hyper)tableau and resolution. Then, in Section 3.2 we discuss the challenges involved in handling both disjunctive clauses and equality stemming from counting quantifiers, which is needed to handle $SHIQ$ ontologies.

### 3.1 Why Consequence-Based Calculi?

Consider the ontology $\mathcal{O}_1$ (written using DL notation) shown in Figure 1. Axiom (3) can be reformulated as $\exists S_1, C_{i+1} \subseteq C_i$, and so $\mathcal{O}_1$ is in $\mathcal{EL}$. One can readily verify that $\mathcal{O} \models B_i \subseteq C_i$ holds for each $1 \leq i \leq n$. 

<table>
<thead>
<tr>
<th>Table 1. Translating Normalised $SHIQ$ Ontologies into DL-Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_1 \subseteq n S.B_2$ $\leadsto$ $B_1(x) \rightarrow S(x, f_i(x))$ for $1 \leq i \leq n$</td>
</tr>
<tr>
<td>$B_1(x) \rightarrow B_2(f_i(x))$ for $1 \leq i \leq n$</td>
</tr>
<tr>
<td>$B_1(x) \rightarrow f_i(x) \not= f_j(x)$ for $1 \leq i &lt; j \leq n$</td>
</tr>
<tr>
<td>$B_1 \subseteq \leq n S.B_2$ $\leadsto$ $B_1(x) \land \forall_{1 \leq i \leq n+1} S(y, x) \land B_2(x) \rightarrow S_{B_2}(y, x)$ for fresh $S_{B_2}$</td>
</tr>
<tr>
<td>$\forall_{1 \leq i \leq n+1} S_{B_2}(x, z_i) \rightarrow \bigvee_{1 \leq i \leq n+1} z_i \approx z_j$</td>
</tr>
<tr>
<td>$\bigwedge_{1 \leq i \leq n \leq m} B_i \leadsto \bigwedge_{1 \leq i \leq n \leq m} B_i(x) \rightarrow \bigvee_{1 \leq i \leq m} B_i(x)$</td>
</tr>
<tr>
<td>$S_1 \subseteq S_2$ $\leadsto$ $S_1(x, z_1) \rightarrow S_2(x, z_1)$</td>
</tr>
<tr>
<td>$S_1 \subseteq S_{2'}$ $\leadsto$ $S_1(x, z_1) \rightarrow S_2(z_1, x)$</td>
</tr>
</tbody>
</table>
\[ \mathcal{O}_1 = \{ B_i \subseteq \exists S_j . B_{i+1} \text{ for } 0 \leq i < n \text{ and } 1 \leq j \leq 2 \} \]
\[ B_n \subseteq C_n \]
\[ C_{i+1} \subseteq \forall S_j . C_i \text{ for } 0 \leq i < n \text{ and } 1 \leq j \leq 2 \}

Moreover, if we extend SHIQ calculi such as \([2, 14, 18, 23]\). We use this framework as basis for our extension to ALCI calculi with the ‘summarisation’ of resolution. In \([22]\), we presented a very general resolution consequences must then be computed in full.

2

More irrelevant conclusions. This problem is exacerbated in satisfiable cases since all

2

be used to constrain model construction, but their effectiveness often depends on the order of rule applications. Thus, model size has proved to be a key limiting factor for (hyper)tableau-based reasoners in practice \([15]\).

In contrast, resolution describes models using universally quantified clauses that ‘summarise’ the model. This prevents redundancy and ensures worst-case optimality of many resolution decision procedures. Nevertheless, resolution can still derive unnecessary clauses. In our example, axioms (1) and (3) are translated into clauses (16) and (17), respectively, which can be used to derive all \(2n^2\) clauses of the form (18).

\[ B_i(x) \rightarrow S_j(x, f_{i,j}(x)) \text{ for } i \in \{1, \ldots, n\} \text{ and } j \in \{1, 2\} \]
\[ S_j(z_1, x) \land C_{k+1}(x) \rightarrow C_k(z_1) \text{ for } k \in \{1, \ldots, n\} \text{ and } j \in \{1, 2\} \]
\[ B_i(x) \land C_{k+1}(f_{i,j}(x)) \rightarrow C_k(x) \text{ for } i, k \in \{1, \ldots, n\} \text{ and } j \in \{1, 2\} \]

Of these \(2n^2\) clauses, only those where \(i = k\) are relevant to proving \(\mathcal{O} \models B_0 \subseteq C_0\).

Moreover, if we extend \(\mathcal{O}\) with additional clauses that contain \(B_i\) and \(C_i\), each of the \(2n^2\) clauses from (18) can participate in further inferences, which can give rise to many more irrelevant conclusions. This problem is exacerbated in satisfiable cases since all resolution consequences must then be computed in full.

Consequence-based calculi combine the goal-directed reasoning of (hyper)tableau calculi with the ‘summarisation’ of resolution. In \([22]\), we presented a very general framework for ALCI ontologies that captures the key elements of consequence-based calculi such as \([2, 14, 18, 23]\). We use this framework as basis for our extension to SHIQ so, before presenting our extension, we explain the main concepts on \(\mathcal{O}_1\). Due to space restrictions we cannot reproduce in full the inference rules from \([22]\); however, these are similar in spirit to our inference rules for SHIQ presented in Table 2.

Our calculus constructs a context structure \(\mathcal{D} = (\mathcal{V}, \mathcal{E}, \mathcal{S}, \text{core}, \succ)\)—a graph whose vertices \(\mathcal{V}\) are called contexts and whose directed edges are labelled with concepts of the

![Fig. 1. Example Motivating Consequence-Based Calculi](image-url)
form $\exists S.B$. Let $I$ be a model of $O$. Instead of representing each element of $I$ individually as in (hyper)tableau calculi, we ‘summarise’ all elements of a certain kind using a single context $v$. Each context $v \in V$ is associated with a (possibly empty) set $\text{core}_v$ of core concepts that hold in all domain elements that $v$ represents; thus, $\text{core}_v$ determines the kind of context $v$. We use a set $S_v$ of clauses to capture additional constraints that the elements represented by $v$ must satisfy; in $\mathcal{ALCL}_v$, we can do so using clauses over DL concepts of the form $\bigwedge B_i \subseteq \bigwedge B_j \cup \exists S_k. B_k \cup \bigvee S_l. B_l$. Thus, unlike in resolution where all clauses belong to a single set, we partition clauses into several sets to reduce the number of inferences. Clauses in $S_v$ are “relative” to $\text{core}_v$: for each $\Gamma \subseteq \Delta \in S_v$, we have $O \models \text{core}_v \cap \Gamma \subseteq \Delta$—that is, we choose not to include $\text{core}_v$ in clause bodies since $\text{core}_v$ always holds. Finally, $\succ$ provides each context $v \in V$ with a concept order $\succ$, that restricts resolution inferences in the presence of disjunctions.

Consequence-based calculi are not just refutation-complete: they actually derive the required consequences. Figure 1 shows how this is achieved for $O_1 \models B_0 \subseteq C_0$; the core and the clauses are shown, respectively, above and below a context. To prove $B_0 \subseteq C_0$, we introduce context $v_{B_0}$ with $\text{core}_{v_{B_0}} = \{B_0\}$ and clause (4) saying that $B_0$ holds in this context. Next, using the Hyper rule, we derive (5) from (1) and (4); this rule performs hyperresolution, but restricted to one context at a time.

Next, the Succ rule satisfies the existential quantifiers in (5). To this end, the rule uses a parameter called an expansion strategy. A strategy is given two sets of constraints that a successor of $v_{B_0}$ must satisfy due to universal restrictions: $K_1$ contains constraints that must hold, and $K_2$ contains constraints that might hold. Given such $K_1$ and $K_2$, the strategy then decides whether to reuse an existing context or create a fresh one, and in the latter case it also determines how to initialise the new context’s core. In our example, there are no universal restrictions and all information in $v_{B_0}$ is deterministic, so $K_1 = K_2 = \{B_1\}$. For $\mathcal{EL}$, a reasonable strategy is to associate with each concept $B_i$ a context $v_{B_i}$ with $\text{core}_{v_{B_i}} = \{B_i\}$, and to always to satisfy existential quantifiers of the form $\exists S.B_i$ using $v_{B_i}$; thus, in our example we introduce $v_{B_1}$ and initialise it with (8). Note that (5) represents two existential quantifiers, both of which we satisfy (in separate applications of the Succ rule) using $v_{B_1}$. Different strategies may be used with more expressive DLs; please refer to [22, Section 3.4] for an in-depth discussion.

We construct contexts $v_{B_2}, \ldots, v_{B_n}$ in a similar way, finally deriving (11) by hyperresolving (2) and (10), and then (12) by hyperresolving (3) and (11). Clause (12) imposes a constraint on the predecessor context, which we propagate backwards using the Pred rule, obtaining (13) and (15). Since, however, clauses in $S_{v_{B_0}}$ are ‘relative’ to $\text{core}_{v_{B_0}}$, clause (15) actually represents our query clause $B_0 \subseteq C_0$.

Thus, like resolution, consequence-based calculi ‘summarise’ models to prevent redundant computation, and, like (hyper)tableau calculi, they differentiate elements in a model of $O$ to prevent the derivation of consequences such as (18).

### 3.2 Extending the Framework to $SHIQ$

Due to an interaction between counting quantifiers and inverse roles, a $SHIQ$ ontology can impose more complex constraints on model elements than $\mathcal{ALCL}$. Let $O_2$ be the $SHIQ$ ontology shown in Figure 2; we argue that $O_2 \models B_0(x) \rightarrow B_1(x)$ holds. To see why, consider a Herbrand interpretation $I$ constructed from $B_0(a)$. Then, (19) and (20)
We must satisfy at least one disjunct in (49). Disjunct $f_2(f_1(a))$ cannot be satisfied due to (21); but then, regardless of whether we satisfy $f_3(f_1(a)) \approx a$ or $f_2(f_1(a)) \approx a$, we derive $B_4(a)$; hence, the inference holds.

To prove this in our consequence-based framework, we must capture constraint (49) and its consequences. However, this cannot be done using standard description logic notation because DL concepts cannot identify specific successors and predecessors of $f_1(a)$—that is, they cannot say ‘either the first or the second successor is equal to the predecessor’. Thus, our main challenges are to devise a method for representing all the relevant constraints that can be induced by $SHIQ$ ontologies, and to ensure that such constraints are correctly propagated between adjacent contexts.

To address these challenges, we skolemise existential quantifiers and transform axioms into DL-clauses. Skolemisation introduces function symbols that act as names for successors. Our clauses thus contain terms of the form $x, f_1(x), y$; variable $x$ represents the elements that a context stands for; $f_i(x)$ represents a successor of $x$; and $y$ represent successors.
represents the predecessor of \( x \). This allows us to represent constraint (49) as

\[
f_2(x) \approx y \lor f_3(x) \approx y \lor f_3(x) \approx f_2(x).
\]

Table 2 shows the inference rules of our calculus that are applicable to such a clausal representation. In each clause, literals are ordered from the smallest to the largest, and so the maximal literal is always on the right; moreover, clause numbers correspond to the order of clause derivation. In the rest of this section, we discuss the rules on so the maximal literal is always on the right; moreover, clause numbers corresponds representation. In each clause, literals are ordered from the smallest to the largest, and

To determine which information to propagate to a successor, Definition 2 introduces a set \( Su(O) \) of successor triggers. In our example, DL-clause (25) contains atoms \( B_1(x) \) and \( S(x, z_i) \) in its body, where \( z_i \) can be mapped to a predecessor or a successor of \( x \), so a context in which hyperresolution is applied to (25) will be interested in information about its predecessors; we reflect this by adding \( B_1(x) \) and \( S(x, y) \) to \( Su(O) \). Thus, the Succ rule introduces context \( v_2 \), sets its core to \( B_1(x) \) and \( S(x, y) \), and initialises the context with (30) and (31).

We next introduce (32)–(35) using hyperresolution, at which point we have sufficient information to apply hyperresolution to (25) to derive (36). Please note how the presence of (30) is crucial for this inference.

We use paramodulation to deal with equality in clause (36). As is common in paramodulation-based theorem proving, we order the literals in a clause and apply inferences only to maximal literals; thus, we derive (37).

Clauses (32), (33), and (37) contain function symbol \( f_2 \), so we again apply the Succ rule and introduce context \( v_2 \). Due to clause (33), we know that \( B_2(x) \) must always hold in \( v_2 \), so we add \( B_2(x) \) to \( core_{v_2} \). However, \( B_3(f_2(x)) \) occurs in clause (37) in a disjunction, so it holds only conditionally in \( v_2 \); we reflect this by including \( B_3(x) \) in the body of clause (41). This allows us derive (42) using hyperresolution.

Clause (42) essentially says ‘\( B_3(f_2(x)) \) should not hold in the predecessor’, so the Pred rule propagates this information to \( v_1 \). This produces clause (43); one can intuitively understand this inference as hyperresolution of (37) and (42), where we take into account that term \( f_2(x) \) in context \( v_2 \) is represented as variable \( x \) in context \( v_3 \).

After two paramodulation steps, we derive clause (45), which essentially says ‘the predecessor must satisfy \( B_2(x) \) or \( B_3(x) \)’. The set \( Pr(O) \) of predecessor triggers from Definition 2 identifies this information as relevant to \( v_1 \); DL-clauses (24) contain atoms...
4 Formalising the Consequence-Based Algorithm for $SHIQ$

We are now ready to formalise our consequence-based algorithm for $SHIQ$. In Section 4.1 we formally introduce the calculus, and in Section 4.2 we present an outline of the completeness proof. Full proofs of all of our technical results can be found in the appendix.

4.1 Definitions

Our calculus manipulates context clauses, which are constructed from context terms and context literals as described in Definition 1. Unlike in general resolution, we restrict context clauses to contain only variables $x$ and $y$, which have a special meaning in our setting: variable $x$ represents a point (i.e., a term) in the model, and $y$ represents the predecessor of $x$; this naming convention is important for the rules of our calculus. This is in contrast to the DL-clauses of an ontology: these can contain variables $x$ and $z_1$, and the latter can refer to either the predecessor or a successor of $x$.

**Definition 1.** A context $\mathcal{F}$-term has the form $x, y$, or $f(x)$ for $f \in \mathcal{F}$; a context $\mathcal{P}$-term has the form $B(y), B(x), B(f(x)), S(x, y), S(y, x), S(x, f(x)), x$, or $S(f(x), x)$ for $B, R \in \mathcal{P}$ and $f \in \mathcal{F}$; and a context term is an $\mathcal{F}$-term or a $\mathcal{P}$-term. A context literal has the form $A \models \top$ (called a context atom), $f(x) \models g(x)$, or $f(x) \models y$, for $A$ a context $\mathcal{P}$-term and $\circ \in \{\approx, \neq\}$. A context clause contains function-free context atoms in the body, and context literals in the head.

Definition 2 introduces sets $Su(O)$ and $Pr(O)$, that identify the information that must be exchanged between adjacent contexts. Intuitively, $Su(O)$ contains atoms that are of interest to a context’s successor, and it guides the Succ rule whereas $Pr(O)$ contains atoms that are of interest to a context’s predecessor and it guides the Pred rule.

**Definition 2.** Let $O$ be an ontology. The set $Su(O)$ of successor triggers of $O$ is the smallest set of atoms such that, for each clause $\Gamma \models \Delta \in O$, we have (i) $B(x) \in \Gamma$ implies $B(x) \in Su(O)$, (ii) $S(x, z_1) \in \Gamma$ implies $S(x, y) \in Su(O)$, and (iii) $S(z_1, x) \in \Gamma$ implies $S(y, x) \in Su(O)$. The set $Pr(O)$ of predecessor triggers is defined as

$$Pr(O) = \{A \models x \mapsto y, y \mapsto x \mid A \in Su(O)\} \cup \{B(y) \mid B \text{ occurs in } O\}.$$ 

Similarly to resolution, our calculus uses a term order $\succ$ to restrict its inferences. Definition 3 specifies the conditions that the order must satisfy. Conditions 1 and 2...
ensure that $F$-terms are compared uniformly across contexts; however, $P$-terms can be compared in different ways in different contexts. Conditions 1 through 4 ensure that mapping $x$ and $y$ to a term $t$ and the predecessor of $t$, respectively, produces a simplification order [4]—a kind of term order commonly used in equational theorem proving. Finally, condition 5 ensures that atoms that might be propagated to a context’s predecessor via the Pred rule are smallest, which is important for completeness.

**Definition 3.** Let $\succ$ be a total, well-founded order on function symbols. A context term order $\succ$ is an order on context terms satisfying the following conditions:

1. for each $f \in F$, we have $f(x) \succ x \succ y$;
2. for all $f, g \in F$ with $f \succ g$, we have $f(x) \succ g(x)$;
3. for all terms $s_1, s_2$, and $t$ and each position $p$ in $t$, if $s_1 \succ s_2$, then $t[s_1]_p \succ t[s_2]_p$;
4. for each term $s$ and each proper position $p$ in $s$, we have $s \succ s[p]$; and
5. for each atom $A \approx \top \in \Pr(O)$ and each context term $s \notin \{x, y\}$, we have $A \not\succ s$.

Order $\succ_v$ is extended to literals, also written $\succ_v$, as described in Section 2.

Note that $\Pr(O)$ contains only atoms of the form $B(y), S(x, y)$, and $S(y, x)$, which we can always make smallest in the ordering; thus, condition 5 does not conflict with the other conditions. Hence, for any given $\succ$, at least one context term order exists.

Apart from orders, effective redundancy elimination techniques are critical to the practical effectiveness of resolution calculi. Definition 4 defines a notion compatible with our setting, and Proposition 1 shows that our calculus is compatible with backward subsumption (which we capture using the Elim rule). Note that tautologies of the form $A \rightarrow A$ are not redundant in our setting as they are used to initialise contexts. However, whenever our calculus derives a clause $A \rightarrow A \lor A'$, the set of clauses will have been initialised with $A \rightarrow A$, which makes the former clause redundant by condition 2 of Definition 4. Moreover, clause heads are subjected to the usual tautology elimination rules; thus, clauses $\gamma \rightarrow \Delta \lor t \approx s$ and $\Gamma \rightarrow \Delta \lor s \approx t \lor t \not\approx s$ are redundant.

**Definition 4.** A set of clauses $U$ contains a clause $\Gamma \rightarrow \Delta$ up to redundancy, written $\Gamma \rightarrow \Delta \in U$, if

1. terms $s$ and $s'$ exist such that $s \approx s \in \Delta$ or $\{s \approx s', s \not\approx s'\} \subseteq \Delta$, or
2. a clause $\Gamma' \rightarrow \Delta' \in U$ exist such that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$.

**Proposition 1.** For $U$ a set of clauses and $C \in U$ a clause such that $C \in U \setminus \{C\}$, for each clause $C'$ with $C' \in U$, we have $C' \in U \setminus \{C\}$.

We are finally ready to formalise the notion of a context structure, as well as a notion of context structure soundness. The latter captures the fact that context clauses from a set $S_v$ do not contain core$_{v}$ in their bodies. We shall later show that our inference rules preserve context structure soundness, which essentially proves that all clauses derived by our calculus are indeed conclusions of the ontology in question.

**Definition 5.** A context structure for an ontology $O$ is a tuple $D = (V, E, S, \text{core}, \succ)$, where $V$ is a finite set of contexts, $E \subseteq V \times V \times F$ is a finite set of edges labelled by function symbols, function core assigns to each context $v$ a conjunction core$_{v}$ of atoms
over the $\mathcal{P}$-terms from $\text{Su}(\mathcal{O})$, function $\mathcal{S}$ assigns to each context $v$ a finite set $\mathcal{S}_v$ of context clauses, and function $\succ$ assigns to each context $v$ a context term order $\succ_v$. A context structure $\mathcal{D}$ is sound for $\mathcal{O}$ if the following conditions both hold:

1. $\mathcal{O} \models \text{core}_v \land \Gamma \rightarrow \Delta$ for each context $v \in \mathcal{V}$ and each clause $\Gamma \rightarrow \Delta \in \mathcal{S}_v$, and
2. $\mathcal{O} \models \text{core}_u \rightarrow \text{core}_v \{x \mapsto f(x), y \mapsto x\}$ for each edge $\langle u, v, f \rangle \in \mathcal{E}$.

Definition 6 introduces an expansion strategy—a parameter of our calculus that determines when and how to reuse contexts in order to satisfy existential restrictions. We have discussed the roles of expansion strategies in Section 3.1; moreover, in [22] we presented several expansion strategies for the DLs contained in $\mathcal{ALCI}$, and adapting these to $\mathcal{SHIQ}$ is straightforward.

Definition 6. An expansion strategy is a polynomial-time computable function strategy that takes a set of atoms $K$ and a context structure $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \text{core}, \succ \rangle$. The result of strategy $(\mathcal{K}, \mathcal{D})$ is a triple $\langle v, \text{core}', \succ' \rangle$ where $\text{core}'$ is a subset of $K$; either $v \notin \mathcal{V}$ is a fresh context, or $v \in \mathcal{V}$ is an existing context in $\mathcal{D}$ such that $\text{core}_v = \text{core}'$; and $\succ'$ is a context term order.

We are now ready to state the formal properties of our calculus.

**Theorem 1.** Applying an inference rule from Table 2 to an ontology $\mathcal{O}$ and a context structure $\mathcal{D}$ that is sound for $\mathcal{O}$ produces a context structure that is sound for $\mathcal{O}$.

**Theorem 2.** Let $\mathcal{O}$ be an ontology, and let $\mathcal{D} = \langle \mathcal{V}, \mathcal{E}, \mathcal{S}, \text{core}, \succ \rangle$ be a context structure such that no inference rule from Table 2 is applicable to $\mathcal{O}$ and $\mathcal{D}$. Then, for each query clause $\Gamma_Q \rightarrow \Delta_Q$ and each context $q \in \mathcal{V}$ that satisfy all of the following conditions, we have $\Gamma_Q \rightarrow \Delta_Q \in \mathcal{S}_q$:

1. $\mathcal{O} \models \Gamma_Q \rightarrow \Delta_Q$.
2. For each atom $A \approx t \in \Delta_Q$ and each context term $s \notin \{x, y\}$ such that $A \succ_q s$, we have $s \approx t \in \Delta_Q \cup \text{Pr}(\mathcal{O})$.
3. For each $A \in \Gamma_Q$, we have $\Gamma_Q \rightarrow A \in \mathcal{S}_q$.

Theorems 1 and 2 show that our calculus is sound and complete, respectively, and thus they suggest the following algorithm for deciding $\mathcal{O} \models \Gamma_Q \rightarrow \Delta_Q$.

1. Create an empty context structure $\mathcal{D}$, introduce a context $q$, and, for each $A \in \Gamma_Q$, add $\Gamma_Q \rightarrow A$ to $\mathcal{S}_q$, in order to satisfy condition C3.
2. Initialise $\succ_q$ in a way that satisfies condition C2, and select an expansion strategy.
3. Saturate $\mathcal{D}$ and $\mathcal{O}$ using the inference rules from Table 2.
4. $\Gamma_Q \rightarrow \Delta_Q$ holds if and only if $\Gamma_Q \rightarrow \Delta_Q \in \mathcal{S}_q$.

**Proposition 2.** For each expansion strategy that introduces at most exponentially many contexts, the consequence-based calculus for $\mathcal{SHIQ}$ is worst-case optimal.

**Proof.** The number $\varphi$ of different context clauses that can be generated using the symbols in $\mathcal{O}$ is clearly at most exponential in the size of $\mathcal{O}$, and the number $m$ of clauses participating in each inference is linear in the size of $\mathcal{O}$. Hence, with $k$ contexts, the number of inferences is bounded by $(k \cdot \varphi)^m$; if $k$ is at most exponential in the size of $\mathcal{O}$, the number of inferences is exponential as well. Thus, our algorithm runs in exponential time, which is worst-case optimal for $\mathcal{SHIQ}$ [3].
Table 2. The rules of the consequence-based calculus for $SHIQ$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>If $A \in \text{core}_v$, then $\top \rightarrow A \notin S_v$, add $\top \rightarrow A$ to $S_v$.</td>
</tr>
<tr>
<td>Hyper</td>
<td>If $\bigwedge_{i=1}^n A_i \rightarrow \Delta \in \mathcal{O}$, $\sigma$ is a substitution such that $\sigma(x) = x$, $\Gamma_i \rightarrow \Delta_i \vee A_i \sigma \in S_v$ with $\Delta_i \nvdash_v A_i \sigma$ for $i \in {1, \ldots, n}$, then $\bigwedge_{i=1}^n \Gamma_i \rightarrow \Delta \vee \bigvee_{i=1}^n \Delta_i \notin S_v$, add $\bigwedge_{i=1}^n \Gamma_i \rightarrow \Delta \vee \bigvee_{i=1}^n \Delta_i$ to $S_v$.</td>
</tr>
<tr>
<td>Eq</td>
<td>If $\Gamma_1 \rightarrow \Delta_1 \vee s_1 \approx t_1 \in S_v$ with $s_1 \not\succeq t_1$ and $\Delta_1 \nvdash_v s_1 \approx t_1$, $\Gamma_2 \rightarrow \Delta_2 \vee s_2 \circ t_2 \in S_v$ with $\circ \in {\approx, \nvdash}$ and $(s_2 \succeq t_2 \circ t_2)$, and $\Gamma_1 \wedge \Gamma_2 \rightarrow \Delta_1 \vee \Delta_2 \vee s_2 \circ t_2$, then $\Gamma \rightarrow \Delta \vee t \nvdash t$ and $\Gamma \rightarrow \Delta \notin S_v$, add $\Gamma \rightarrow \Delta$ to $S_v$.</td>
</tr>
<tr>
<td>Ineq</td>
<td>If $\Gamma \rightarrow \Delta \vee t \nvdash t \in S_v$ with $\Delta \nvdash_v t \nvdash t$, and $\Gamma \rightarrow \Delta \notin S_v$, then $\Gamma \rightarrow \Delta \rightarrow \Delta$ to $S_v$.</td>
</tr>
<tr>
<td>Elim</td>
<td>If $\Gamma \rightarrow \Delta \in S_v$ and $\Gamma \rightarrow \Delta \rightarrow \Delta \rightarrow \Delta$ from $S_v$, then remove $\Gamma \rightarrow \Delta$ from $S_v$.</td>
</tr>
<tr>
<td>Pred</td>
<td>If $(u, v, f) \in \mathcal{E}$, $\bigwedge_{i=1}^n A_i \rightarrow \bigvee_{i=m+1}^{m+n} A_i \in S_v$, $\Gamma_1 \rightarrow \Delta_1 \vee A_i \sigma \in S_v$ with $\Delta_i \nvdash_u A_i \sigma$ for $i \leq m$, $\Gamma_i \in \text{Pr}(\mathcal{O})$ for each $m+1 \leq i \leq m+n$, and $\bigwedge_{i=1}^n \Gamma_i \rightarrow \bigvee_{i=m+1}^{m+n} A_i \sigma \notin S_v$, add $\bigwedge_{i=1}^n \Gamma_i \rightarrow \bigvee_{i=m+1}^{m+n} A_i \sigma$ to $S_v$;</td>
</tr>
<tr>
<td>Succ</td>
<td>If $\Gamma \rightarrow \Delta \in S_v$ where $\Delta \nvdash_u A$ and $A$ contains $f(x)$, and no edge $(u, v, f) \in \mathcal{E}$ exists such that $A \rightarrow A \in S_v$ for each $A \in K_2 \setminus \text{core}_v$, let $(v, \text{core}', \succ') := \text{strategy}(K_1, D)$; if $v \not\in \mathcal{V}$, then let $\succ_v := \succ_v \cap \succ'$, and otherwise let $\mathcal{V} := \mathcal{V} \cup {v}$, $\text{core}_v := \text{core}'$, $\succ_v := \succ'$, and $S_v := \emptyset$; add the edge $(u, v, f)$ to $\mathcal{E}$; and add $A \rightarrow A$ to $S_v$ for each $A \in K_2 \setminus \text{core}_v$;</td>
</tr>
</tbody>
</table>

where $\sigma = \{x \mapsto f(x), y \mapsto x\}$, $K_1 = \{A \in \text{Su}(\mathcal{O}) \mid \Gamma \rightarrow A \sigma \in S_v\}$, and $K_2 = \{A \in \text{Su}(\mathcal{O}) \mid \Gamma' \rightarrow A \sigma \in S_v$ and $\Delta' \nvdash_u A \sigma\}$. |
4.2 An Outline of the Completeness Proof

To prove Theorem 2, we fix an ontology $O$, a context structure $D$, a query clause $\Gamma_Q \rightarrow \Delta_Q$, and a context $q$ such that properties C2 and C3 of Theorem 2 are satisfied and $\Gamma_Q \rightarrow \Delta_Q \notin S_q$ holds, and we construct an interpretation that satisfies $O$ but refutes $\Gamma_Q \rightarrow \Delta_Q$. We reuse techniques from equational theorem proving [16] and represent this interpretation by a rewrite system $R$—a finite set of rules of the form $l \Rightarrow r$.

Intuitively, such a rule says that that any two terms of the form $f_1(...f_n(l)...) \Rightarrow f_1(...f_n(r)...) \Rightarrow$ with $n \geq 0$ are equal, and that we can prove this equality in one step by rewriting (i.e., replacing) $l$ with $r$. Rewrite system $R$ induces a Herbrand equality interpretation $R^*$ that contains each $l \approx r$ for which the equality between $l$ and $r$ can be verified using a finite number of such rewrite steps. The universe of $R^*$ consists of $\mathcal{F}$- and $\mathcal{P}$-terms constructed using the symbols in $\mathcal{F}$ and $\mathcal{P}$, and a special constant $c$; for convenience, let $\mathcal{T}$ be the set of all $\mathcal{F}$-terms from this universe.

We obtain $R^*$ by unfolding the context structure $D$ starting from context $q$: we map each $\mathcal{F}$-term $t \in \mathcal{T}$ to a context $X_t$ in $D$, and we use the clauses in $S_{X_t}$ to construct a model fragment $R_t$—the part of $R$ that satisfies the DL-clauses of $O$ when $x$ is mapped to $t$. The key issue is to ensure compatibility between adjacent model fragments: when moving from a predecessor term $t'$ to a successor term $t = f(t')$, we must ensure that adding $R_t$ to $R_{t'}$ does not affect the truth of the DL-clauses of $O$ at term $t'$; in other words, the model fragment constructed at $t$ must respect the choices made at $t'$. We represent these choices by a ground clause $\Gamma_t \rightarrow \Delta_t$: conjunction $\Gamma_t$ contains atoms that are ‘inherited’ from $t'$ and so must hold at $t$, and disjunction $\Delta_t$ contains atoms that must not hold at $t$ because $t'$ relies on their absence.

The model fragment construction takes as parameters a term $t$, a context $v = X_t$, and a clause $\Gamma_t \rightarrow \Delta_t$. Let $N_t$ be the set of ground clauses obtained from $S_v$ by mapping $x$ to $t$ and $y$ to the predecessor of $t$ (if it exists), and whose body is contained in $\Gamma_t$. Moreover, let $Su_t$ and $Pr_t$ be obtained from $Su(O)$ and $Pr(O)$ by mapping $x$ to $t$ and $y$ to the predecessor of $t$ if one exists; thus, $Su_t$ contains the ground atoms of interest to the successors of $t$, and $Pr_t$ contains the ground atoms of interests to the predecessor of $t$. The model fragment for $t$ can be constructed if the following properties hold:

L1. $\Gamma_t \rightarrow \Delta_t \notin N_t$.
L2. If $t = c$, then $\Delta_t = \Delta_Q$; and if $t \neq c$, then $\Delta_t \subseteq Pr_t$.
L3. For each $A \in \Gamma_t$, we have $\Gamma_t \rightarrow A \notin N_t$.

The model fragment construction produces a rewrite system $R_t$ such that

F1. $R_t^* \models N_t$, and
F2. $R_t^* \notin \Gamma_t \rightarrow \Delta_t$—that is, all of $\Gamma_t$, but none of $\Delta_t$ hold in $R_t^*$, and so the model fragment at $t$ is compatible with the ‘inherited’ constraints.

We construct $R_t$ using the standard techniques from paramodulation-based theorem proving. First, we order all clauses in $N_t$ into a sequence $C^i = \Gamma^i \rightarrow \Delta^i \lor L^i$, for $1 \leq n$, that is compatible with the context ordering $\gg_v$ in a particular way. Next, we initialise $R_t$ to $\emptyset$, and then we examine each clause $C^i$ in this sequence; if $C^i$ does not hold in the model constructed thus far, we make the clause true by adding $L^i$ to $R_t$. To prove condition F1, we assume for the sake of a contradiction that a clause $C^i$ with...
smallest \( i \) exists such that \( R_i^* \not\models C^i \), and we show that an application of the Eq, Ineq, or Factor rule to \( C^i \) necessarily produces a clause \( C^j \) such that \( R_i^* \not\models C^j \) and \( j < i \). Conditions L1 through L3 allow us to satisfy condition F2. Due to condition L2 and condition 5 of Definition 3, we can order the clauses in the sequence such that each clause \( C^i \) capable of producing an atom from \( \Delta_t \) comes before any other clause in the sequence; and then we use condition L1 to show that no such clause actually exists. Moreover, condition L3 ensures that all atoms in \( \Gamma_t \) are actually produced in \( R_i^* \).

To obtain \( R \), we inductively unfold \( D \), and at each step we invoke the model fragment construction with suitably defined parameters. For the base case, we map constant \( c \) to context \( X_c = q \), and we define \( \Gamma_c = \Gamma_Q \) and \( \Delta_c = \Delta_Q \); then, conditions L1 and L2 hold by definition, and condition L3 holds by property C3 of Theorem 2. For the induction step, we assume that we have already mapped some term \( t' \) to a context \( u = X_{t'} \), and, for each function symbol \( f \in F \), we consider term \( t = f(t') \).

- If \( t \) does not occur in an atom in \( R_{t'} \), we let \( R_t = \{ t \Rightarrow c \} \) and thus make \( t \) equal to \( c \). Term \( t \) is thus interpreted in exactly the same way as \( c \), so we stop the unfolding.
- If \( R_{t'} \) contains a rule \( t \Rightarrow s \), then \( t \) and \( s \) are equal, and so we can interpret \( t \) exactly as \( s \); consequently, we stop the unfolding.
- In all other cases, the Succ rule ensures that \( D \) contains an edge \( \langle u, v, f \rangle \) such that \( v \) satisfies all preconditions of the rule, so we define \( X_t = v \). Moreover, we let \( \Gamma_t = R^*_t \cap Su_t \) be the set of atoms that hold at \( t' \) and are relevant to \( t \), and we let \( \Delta_t = Pr_t \setminus R^*_t \) be the set of atoms that do not hold at \( t' \) and are relevant to \( t \). We finally show that such \( \Gamma_t \) and \( \Delta_t \) satisfy condition L1: if that were not the case, then the Pred rule derives a clause in \( N_{t'} \) that is not true in \( R^*_t \).

After processing all relevant terms, we let \( R \) be the union of all \( R_t \) from the above construction. To show that \( R^* \) satisfies \( O \), we consider a DL-clause \( \Gamma' \rightarrow \Delta \in O \) and a substitution \( \tau \) that makes the clause ground. W.l.o.g. we can assume that \( \tau \) is irreducible by \( R \)—that is, it does not contain terms that can we rewritten using the rules in \( R \). Since each model fragment satisfies condition F2, we can evaluate \( \Gamma' \rightarrow \Delta \) in \( R^*_{\tau(x)} \) instead of \( R^* \). Moreover, we show that \( R^*_{\tau(x)} \models \Gamma' \rightarrow \Delta \): if that were not the case, the Hyper rule derives a clause in \( N_{\tau(x)} \) that violates condition F1. Finally, we show that the same holds for the query clause \( \Gamma_Q' \rightarrow \Delta_Q \), which completes our proof.

5 Conclusion

We have presented the first consequence based calculus for SHIQ, a DL that includes both disjunction and counting quantifiers. Our calculus combines ideas from state of the art resolution and (hyper)tableau calculi, including the use of ordered paramodulation for efficient equality reasoning. In spite of its increased complexity, the calculus mimics existing and well proven EL calculi on EL ontologies. Thus, although implementation and evaluation remain as future work, we believe that the calculus is likely to work well on ‘mostly-EL’ ontologies—a type of ontology that occurs commonly in practice.
References

A Proof of Theorem 1

In this chapter, we show that our calculus is sound, as stated in Theorem 1. The proof is analogous to the soundness proof of ordered superposition [16].

**Theorem 1.** Applying an inference rule from Table 2 to an ontology $O$ and a context structure $D$ that is sound for $O$ produces a context structure that is sound for $O$.

**Proof.** Let $O$ be an ontology, let $D = \langle V, E, S, \text{core}, \succ \rangle$ be a context structure sound for $O$, and consider an application of an inference rule from Table 2 to $D$ and $O$. To prove the theorem, we show that each clause produced by the rule is a context clause and that it satisfies conditions S1 and S2 of Definition 5. Condition S1 holds obviously for the theorem, we show that each clause produced by the rule is a context clause and that the edge introduced by the rule satisfies condition S2.

To prove the claim, we consider each rule from Table 2 and assume that the rule is applied to clauses, contexts, and edges as shown in the table; then, we show that the clause produced by the rule satisfies condition S1 of Definition 5; and for the succ rule, we also show that the edge introduced by the rule satisfies condition S2.

**(Core)** For each $A \in \text{core}_v$, we clearly have $O \models \text{core}_v \rightarrow A$.

**(Hyper)** Since $D$ is sound for $O$, we have $O \models \text{core}_v \land \Gamma_i \rightarrow \Delta_i \lor A_i \sigma$ for each $i$ with $1 \leq i \leq n$. By (51), we have $O \models \text{core}_v \land \bigwedge_{i=1}^{n} \Gamma_i \rightarrow \bigvee_{i=1}^{n} \Delta_i \lor \Delta \sigma$. Moreover, substitution $\sigma$ satisfies $\sigma(x) = x$, all premises are context clauses, and $O$ contains only DL-clauses; thus, the inference rule can match an atom $S(x, z_i)$ or $S(z_i, x)$ in an ontology clause to atoms $S(x, y)$ or $S(x, f(x))$ in the context clause, and so $\sigma(z_i)$ is either $y$ or $f(x)$; thus, the result is a context clause.

**(Eq)** Since $D$ is sound for $O$, properties (52) and (53) hold. Moreover, clause in (54) is a logical consequence of the clauses in (52) and (53), so property (54) holds, as required.

$$O \models \text{core}_v \land \Gamma_1 \rightarrow \Delta_1 \lor s_1 \approx t_1$$ (52)

$$O \models \text{core}_v \land \Gamma_2 \rightarrow \Delta_2 \lor s_2 \circ t_2$$ (53)

$$O \models \text{core}_v \land \Gamma_1 \land \Gamma_2 \rightarrow \Delta_1 \lor \Delta_2 \lor s_2[t_1|p \circ t_2$$ (54)

Moreover, term $s_1$ is always of the form $g(f(x))$, term $t_1$ is of the form $h(f(x))$ or $y$, and term $s_2$ is of the form $g(f(x))$, $B(f(g(x)))$, $S(x, f(g(x)))$, or $S(f(g(x)), x)$; thus, $s_2[t_1|p$ is a context term, and so the result is a context clause.

**(Ineq)** Since $D$ is sound for $O$, we have $O \models \text{core}_v \land \Gamma \rightarrow \Delta \lor t \not\approx t$; but then, we clearly have $O \models \text{core}_v \land \Gamma \rightarrow \Delta$, as required.

**(Factor)** Since $D$ is sound for $O$, property (55) holds. Moreover, clause in (56) is a logical consequence of the clause in (55), so property (56) holds, as required.

$$O \models \text{core}_v \land \Gamma \rightarrow \Delta \lor s \approx t \lor s \approx t'$$ (55)
\( O \models \text{core}_v \land \Gamma \rightarrow \Delta \lor t \not\approx t' \lor s \approx t' \) \hspace{1cm} (56)

(Elim) The resulting context structure contains a subset of the clauses than \( D \), so it is clearly sound for \( O \).

(Pred) Let \( \sigma = \{ x \mapsto f(x), y \mapsto x \} \). Since \( D \) is sound for \( O \), properties (57)–(59) hold. Now clause in (60) is an instance of the clause in (57), so property (60) holds. But then, by (51), properties (57) and (58) imply property (61). Finally, properties (59) and (61) imply property (62), as required.

\[
\begin{align*}
O &\models \text{core}_v \land \bigwedge_{i=1}^{m} A_i \rightarrow \bigvee_{j=m+1}^{m+n} A_j & (57) \\
O &\models \text{core}_v \land \Gamma_i \rightarrow \Delta_i \lor A_i \sigma & \text{for } 1 \leq i \leq m \hfill (58) \\
O &\models \text{core}_v \rightarrow \text{core}_v \sigma \hfill (59) \\
O &\models \text{core}_v \land \bigwedge_{i=1}^{m} A_i \sigma \rightarrow \bigvee_{j=m+1}^{m+n} A_j \sigma & (60) \\
O &\models \text{core}_v \land \text{core}_v \land \bigwedge_{i=1}^{m} \Gamma_i \rightarrow \bigvee_{j=m+1}^{m+n} A_j \sigma & (61) \\
O &\models \text{core}_v \land \bigwedge_{i=1}^{m} \Gamma_i \rightarrow \bigvee_{j=m+1}^{m+n} A_j \sigma & (62)
\end{align*}
\]

For each \( m + 1 \leq i \leq m + n \), we have \( A_i \in \text{Pr}(O) \), so \( A_i \) is of the form \( B(y), S(x, y), \) or \( S(y, x) \); but then, the definition of \( \sigma \) ensures that \( A_i \sigma \) is a context atom, as required.

(Succ) Let \( \sigma = \{ x \mapsto f(x), y \mapsto x \} \). For each clause \( A \rightarrow A \) added to \( S \), we clearly have \( O \models \text{core}_v \land A \rightarrow A \), as required for condition S1 of Definition 5. Moreover, assume that the inference rule adds an edge \( \langle u, v, f_k \rangle \) to \( E \); since \( D \) is sound for \( O \), we have (63); by Definition 6, we have \( \text{core}_v \subseteq K_1 \).

\[
\begin{align*}
O &\models \text{core}_v \rightarrow A \sigma & \text{for each } A \in K_1 \hfill (63) \\
O &\models \text{core}_v \rightarrow \text{core}_v \sigma \hfill (64)
\end{align*}
\]

But then, property (64) holds, as required for condition S2 of Definition 5. \( \square \)

### B Preliminaries: Rewrite Systems

In the proof of Theorem 2 we construct a model of an ontology, which, as is common in equational theorem proving, we represent using a ground rewrite system. We next recapitulate the definitions of rewrite systems, following the presentation by [4].

Let \( T \) be the set of all ground terms constructed using a distinguished constant \( c \) (of sort \( \mathcal{F} \)), the function symbols from \( \mathcal{F} \), and the predicate symbols from \( \mathcal{P} \). A (ground) rewrite system \( R \) is a binary relation on \( T \). Each pair \( (s, t) \in R \) is called a rewrite rule and is commonly written as \( s \Rightarrow t \). The rewrite relation \( \rightarrow_R \) for \( R \) is the smallest binary relation on \( T \) such that, for all terms \( s_1, s_2, t \in T \) and each (not necessarily proper) position \( p \) in \( t \), if \( s_1 \Rightarrow s_2 \in R \), then \( t[s_1]_p \Rightarrow t[s_2]_p \). Moreover, \( \Rightarrow_R \) is the reflexive–transitive closure of \( \Rightarrow_R \), and \( \subseteq_R \) is the reflexive–symmetric–transitive closure of \( \Rightarrow_R \). A term \( s \) is irreducible by \( R \) if no term \( t \) exists such that \( s \Rightarrow_R t \); and a literal, clause, or substitution \( \alpha \) is irreducible by \( R \) if no term occurring in \( \alpha \) is irreducible by \( R \). Moreover, term \( t \) is a normal form of \( s \) w.r.t. \( R \) if \( s \Rightarrow_R t \) and \( t \) is irreducible by \( R \). We consider the following properties of rewrite systems.
– R is terminating if no infinite sequence s₁, s₂, … of terms exists such that, for each i, we have sᵢ →ₚ sᵢ₊₁.

– R is left-reduced if, for each s ⇒ t ∈ R, the term s is irreducible by R \ {s ⇒ t}.

– R is Church-Rosser if, for all terms t₁ and t₂ such that t₁ →ₚ t₂, a term z exists such that t₁ →ₚ R z and t₂ →ₚ R z.

If R is terminating and left-reduced, then R is Church-Rosser [4, Theorem 2.1.5 and Exercise 6.7]. If R is Church-Rosser, then each term s has a unique normal form t such that s →ₚ t holds. The Herbrand interpretation induced by a Church-Rosser system R is the set R* such that, for all s, t ∈ T, we have s ≈ t ∈ R* if and only if s ≈ t.

Term orders can be used to prove termination of rewrite systems. A term order Γ is a simplification order if the following conditions hold:

– for all terms s₁, s₂, and t, all positions p in t, and all substitutions σ, we have that
  s₁ ≻ s₂ implies t[s₁σ]p ≻ t[s₂σ]p, and
– for each term s and each proper position p in s, we have s ≻ s|p.

If, for R a rewrite system, a simplification order Γ exists such that s ⇒ t ∈ R implies s ≻ t, then R is terminating [4, Theorems 5.2.3 and 5.4.8], and s →ₚ t implies s ≻ t.

C Proof of Theorem 2

Theorem 2. Let O be an ontology, and let D = (V, E, S, core, ≻) be a context structure such that no inference rule from Table 2 is applicable to O and D. Then, for each query clause \( \Gamma_Q \rightarrow \Delta_Q \) and each context term \( q \in V \) that satisfy all of the following conditions, we have \( \Gamma_Q \rightarrow \Delta_Q \in S_q \):

C1. \( O \models \Gamma_Q \rightarrow \Delta_Q \).

C2. For each atom \( A \approx t \in \Delta_Q \) and each context term \( s \notin \{ x, y \} \) such that \( A \succ_q s \), we have \( s \approx t \in \Delta_Q \cup \text{Pr}(O) \).

C3. For each \( A \in \Gamma_Q \), we have \( \Gamma_Q \rightarrow A \in S_q \).

In this section, we fix an ontology O, a context structure D = (V, E, S, core, ≻), a context \( q \in V \), and a query clause \( \Gamma_Q \rightarrow \Delta_Q \) such that conditions C3 and C2 of Theorem 2 are satisfied, and we show the contrapositive of condition C1: if \( \Gamma_Q \rightarrow \Delta_Q \notin S_q \), then \( O \models \Gamma_Q \rightarrow \Delta_Q \). To this end, we construct a rewrite system R such that the induced Herbrand model R* satisfies all clauses in O, but not \( \Gamma_Q \rightarrow \Delta_Q \). We construct the model using a distinguished constant \( c \), the unary function symbols from F, and the unary and binary predicate symbols from \( \mathcal{P}_1 \) and \( \mathcal{P}_2 \), respectively. Let t and s be terms with \( t = f(s) \); then, s is the predecessor of t, and t is a successor of s; by these definitions, a constant has no predecessor. Let t be a term. The F-neighbourhood of t is the set of F-terms \( t, f(t) \) with \( f \in F \), and the predecessor \( t' \) of t if one exists; the \( \mathcal{P} \)-neighbourhood of t contains \( \mathcal{P} \)-terms \( B(t), S(t, f(t)), S(f(t), t), B(f(t)) \), and, if t has the predecessor \( t' \), also \( \mathcal{P} \)-terms \( S(t', t), S(t, t'), \) and \( B(t') \), for all \( B \in \mathcal{P}_1 \) and \( S \in \mathcal{P}_2 \). Let \( \sigma_t \) be the substitution such that \( \sigma_t(x) = t \) and, if \( t \) has the predecessor \( t' \), then \( \sigma_t(y) = t' \). Finally, for each term t, we define sets of atoms \( \text{Pr}_t \) and \( \text{Su}_t \) as follows:

\[
\text{Su}_t = \{ A\sigma_t \mid A \in \text{Su}(O) \text{ and } A\sigma_t \text{ is ground } \} \tag{65}
\]

\[
\text{Pr}_t = \{ A\sigma_t \mid A \in \text{Pr}(O) \text{ and } A\sigma_t \text{ is ground } \} \tag{66}
\]
C.1 Constructing a Model Fragment

In this section, we show how, given a term \( t \), we can generate a part of the model of \( \mathcal{O} \) that covers the neighbourhood of \( t \). In the rest of Appendix C.1, we fix the following parameters to the model fragment generation process:

- \( t \) is a ground \( \mathcal{F} \)-term,
- \( v \) is a context in \( D \),
- \( \Gamma_t \) is a conjunction of atoms, and
- \( \Delta_t \) is a disjunction of atoms.

Let \( N_t \) be the set of ground clauses obtained from \( S_v \) as follows:

\[
N_t = \{ \Gamma_{t\sigma} \rightarrow \Delta_{t\sigma} | \Gamma \rightarrow \Delta \in S_v, \text{both } \Gamma_{t\sigma} \text{ and } \Delta_{t\sigma} \text{ are ground, and } \Gamma_{t\sigma} \subseteq \Gamma_t \}
\]

We assume that the following conditions hold.

L1. \( \Gamma_t \rightarrow \Delta_t \notin N_t \).
L2. If \( t = c \), then \( \Delta_t = \Delta_Q \); and if \( t \neq c \), then \( \Delta_t \subseteq \Pr_t \).
L3. For each \( A \in \Gamma_t \), we have \( \Gamma_t \rightarrow A \in N_t \).

We next construct a rewrite system \( R_t \) that satisfies \( R_t^* | N_t = R_t^* \not\in \Gamma_t \rightarrow \Delta_t \).

Throughout Appendix C.1, we treat the terms in the \( \mathcal{F} \)-neighbourhood of \( t \) is if they were constants. Thus, even though the rewrite system \( R \) will contain terms \( t \) and \( f(t) \), we will not consider terms with further nesting.

C.1.1 Grounding the Context Order

To construct \( R_t \), we need an order on the terms in the neighbourhood of \( t \) that is compatible with \( \succ_v \). To this end, let \( \succ_t \) be a total, strict, simplification order on the set of ground terms constructed using the \( \mathcal{F} \)-neighbourhood of \( t \) and the predicate symbols in \( \mathcal{P} \) that satisfies the following conditions for all context terms \( s_1 \) and \( s_2 \) such that \( s_1 \sigma_t \) and \( s_2 \sigma_t \) are both ground, and for \( t' \) the predecessor of \( t \) (if one exists).

O1. \( s_1 \succ_v s_2 \) implies \( s_1 \sigma_t \succ_t s_2 \sigma_t \).
O2. \( s_1 \sigma_t \approx t \in \Delta_t \) and \( s_1 \sigma_t \succ_t s_2 \sigma_t \) and \( s_2 \sigma_t \notin \{ t, t' \} \) imply \( s_2 \sigma_t \approx t \in \Delta_t \).

Condition C2 of Theorem 2 and condition 5 of Definition 3 ensure that the order \( \succ_v \) on (nonground) context terms can be grounded in a way compatible with condition L2. Moreover, since in this section we treat all \( \mathcal{F} \)-terms as constants, we can make the \( \mathcal{P} \)-terms of the form \( B(t') \), \( S(t', t) \), and \( S(t, t') \) smaller than other \( \mathcal{F} \) - and \( \mathcal{P} \)-terms (i.e., we need not worry about defining the order on the predecessor of \( t' \) or on the ancestors of \( f(x) \)). Thus, at least one such order exists, so in the rest of this section we fix an arbitrary such order \( \succ_t \). We extend \( \succ_t \) to ground literals (also written \( \succ_t \)) using the multiset extension from Section 2, and we extend \( \succ_t \) to ground disjunctions (also written \( \succ_t \)) be treating disjunctions as multisets and using the multiset extension.
C.1.2 Constructing the Rewrite System $R_t$

We arrange all clauses in $N_t$ into a sequence $C^1, \ldots, C^n$. No $C^i$ can contain $\bot$ in its head as that would contradict condition L1; thus, we can assume that each $C^i$ is of the form $C^i = \Gamma^i \to \Delta^i \lor L^i$ where $L^i \gg_1 \Delta^i$, literal $L^i$ is of the form $L^i = l^i \circ r^i$ with $\circ \in \{\approx, \not\approx\}$, and $l^i \gg r^i$. For the rest of Appendix C.1, we reserve $C^i$, $\Gamma^i$, $\Delta^i$, $L^i$, $l^i$, and $r^i$ for referring to the (parts of) the clauses in this sequence. Finally, we assume that, for all $1 \leq i < j \leq n$, we have $\Delta^i \lor L^i \gg_1 \Delta^j \lor L^j$.

We next define the sequence $R^0_t, \ldots, R^n_t$ of rewrite systems by setting $R^0_t = \emptyset$ and defining each $R^i_t$ with $1 \leq i \leq n$ inductively as follows:

- $R^0_t = R^{i-1}_t \cup \{l^i \to r^i\}$ if $l^i$ is of the form $l^i \approx r^i$ such that
  - R1. $(R^{i-1}_t)^\ast \not\approx \Delta^i \lor l^i \approx r^i$,
  - R2. $l^i \gg r^i$,
  - R3. $l^i$ is irreducible by $R^{i-1}_t$, and
  - R4. $s \approx r^i \not\in (R^{i-1}_t)^\ast$ for each $l^i \approx s \in \Delta^i$;
- $R^i_t = R^{i-1}_t$ in all other cases.

Finally, let $R_t = R^n_t$; we call $R_t$ the model fragment for $t$, $v$, $\Gamma_t$, and $\Delta_t$. Each clause $C^i = \Gamma^i \to \Delta^i \lor l^i \approx r^i$ that satisfies the first condition in the above construction is called generative, and the clause is said to generate the rule $l^i \Rightarrow r^i$ in $R_t$.

C.1.3 The Properties of the Model Fragment $R_t$

Lemma 1. The rewrite system $R_t$ is Church-Rosser.

Proof. To see that $R_t$ is terminating, simply note that, for each rule $l \Rightarrow r \in R_t$, condition R2 ensures $l \gg r$, and that $\gg$ is a simplification order.

To see that $R_t$ is left-reduced, consider an arbitrary rule $l \Rightarrow r \in R_t$ that is added to $R_t$ in step $i$ of the clause sequence. By condition R3, $l \Rightarrow r$ is irreducible by $R^i_t$. Now consider an arbitrary rule $l' \Rightarrow r' \in R_t$ that is added to $R_t$ at any step $j$ of the construction where $j > i$. The definition of the clause order implies $l' \gg r' \gg_1 j \gg r$; since $l' \gg r'$ and $l \gg r$ by condition R2, by the definition of the literal order we have $l' \gg_1 l$. Since $l \Rightarrow r \in R^{i-1}_t$, condition R3 ensures $l \not\gg l'$, and so we have $l' \gg l$; consequently, $l'$ is not a subterm of $l$, and thus $l$ is irreducible by $R^i_t$. \hfill $\square$

Lemma 2. For each clause $\Gamma \to \Delta$ such that $\Gamma \to \Delta \in S_v$ and $\Gamma \sigma_t \subseteq \Gamma_t$ hold, we have $\Gamma \sigma_t \to \Delta \sigma_t \in N_i$.

Proof. Assume that $\Gamma \to \Delta \in S_v$ holds. If $\Gamma \to \Delta$ satisfies condition 1 of Definition 4, then terms $s$ and $s'$ exist such that $s \approx s' \in \Delta$ or $\{s \approx s', s \not\approx s'\} \subseteq \Delta$; but then, $s \approx s' \in \Delta \sigma_t$ or $\{s \approx s' \sigma_t, s \not\approx s' \sigma_t\} \subseteq \Delta$, so we have $\Gamma \sigma_t \to \Delta \sigma_t \in N_i$. Furthermore, if $\Gamma \to \Delta$ satisfies condition 2 of Definition 4, then clause $\Gamma' \to \Delta' \in S_v$ exists such that $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$; but then, due to $\Gamma' \sigma_t \subseteq \Gamma \sigma_t \subseteq \Gamma_t$, we have that $\Gamma' \sigma_t \to \Delta' \sigma_t \in N_i$ holds, and therefore $\Gamma \sigma_t \to \Delta \sigma_t \in N_t$ holds as well. \hfill $\square$

Lemma 3. For each $1 \leq i \leq n$ and each $l \not\approx r \in \Delta^i \lor L^i$, we have $(R^{i-1}_t)^\ast \models l \gg r$ if and only if $R^i_t \models l \gg r$.
Consider an arbitrary clause $C^i = \Gamma^i \rightarrow \Delta^i \lor L^i$ and an arbitrary inequality $l \not\approx r \in \Delta^i \lor L^i$. If $l \approx r \in (R_{l}^{i-1})^*$, then $R_{l}^{i-1} \subseteq R_{l}$ implies $l \approx r \in R_{l}^{i}$, and so we have $R_{l}^{i} \models l \approx r$, as required. Now assume that $l \approx r \not\in (R_{l}^{i-1})^*$. Let $l'$ and $r'$ be the normal forms of $l$ and $r$, respectively, w.r.t. $R_{l}^{i-1}$. Now consider an arbitrary $j$ with $i \leq j \leq n$ such that $l' \approx r'$ is generated by $C^j$. We then have $l' \approx r' \geq_l l \not\approx r$, which by the definition of literal order implies $l' \geq_l l' \lor l$ and $l' \geq_l r \lor r'$; since $>_{l}$ is a simplification order, $l'$ is a subterm of neither $l'$ nor $r'$. Thus, $l'$ and $r'$ are the normal forms of $l$ and $r$, respectively, w.r.t. $R_{l}^{i}$, and so we have $l' \approx r' \not\in (R_{l}^{i})^*$; but then, we have $l \approx r \not\in (R_{l}^{i})^*$, as required. 

**Lemma 4.** For each generative clause $C^i \rightarrow \Delta^i \lor L^i \approx r^i$, we have $R_{l}^{i} \not\models \Delta^i$.

**Proof.** Consider a generative clause $C^i = \Gamma^i \rightarrow \Delta^i \lor L^i \approx r^i$ and a literal $L \in \Delta^i$; condition R1 ensures that $(R_{l}^{i-1})^* \not\models L$. We next show that $(R_{l}^{i-1})^* \not\models L$. Thus, assume that $C^j$ is not generational, then $R_{l}^{i} = R_{l}^{i-1}$, and so $(R_{l}^{i})^* \not\models L$. Thus, assume that $C^j$ is generational. We consider the following two cases.

- $l' = l$. We have the following two subcases.
  1. $j = i$. Condition R4 then ensures $r \approx r^i \not\in (R_{l}^{i-1})^*$. Let $r'$ and $r''$ be the normal forms of $r$ and $r'$, respectively, w.r.t. $R_{l}^{i-1}$; we have $r \approx r'' \not\in (R_{l}^{i-1})^*$. Moreover, $l \vdash r \geq_{l} r'$ and $l \vdash r \geq_{l} r''$ hold; since $>_{l}$ is a simplification order, $l$ is a subterm of neither $r'$ nor $r''$; therefore, $r'$ and $r''$ are the normal forms of $r$ and $r'$, respectively, w.r.t. $R_{l}^{i}$, and therefore $r' \approx r'' \not\in (R_{l}^{i})^*$. Finally, since $l \vdash r^i \in R_{l}^{i}$, term $r''$ is the normal form of $l$ w.r.t. $R_{l}^{i}$, and so $l \approx r \not\in (R_{l}^{i})^*$.
  2. $j > i$. But then, $l' \vdash l' \geq_{l'} l' \approx r \vdash l \approx r$ implies $l' = l = l$. Furthermore, $C^i$ is generational, so we have $l' \vdash r^i \in R_{l}^{i-1}$. But then, $l'$ is not irreducible by $R_{l}^{i-1}$, which contradicts condition R3.

- $l' \geq_{l'} l$. Let $l'$ and $r'$ be the normal forms of $l$ and $r$, respectively, w.r.t. $R_{l}^{i-1}$. Then, we have $l' \geq_{l'} l \geq_{l'} l'$ and $l' \geq_{l'} r \geq_{l'} r'$; since $>_{l}$ is a simplification order, $l'$ is a subterm of neither $l'$ nor $r'$. Thus, $l'$ and $r'$ are the normal forms of $l$ and $r$, respectively, w.r.t. $R_{l}^{i}$, and so $l' \approx r' \not\in (R_{l}^{i})^*$; hence, $l \approx r \not\in (R_{l}^{i})^*$ holds. 

**Lemma 5.** Let $\Gamma \rightarrow \Delta$ be a clause with $\Gamma \rightarrow \Delta \in N_\ell$. Then $R_{l}^{i} \models \Delta$ holds whenever some $i$ with $1 \leq i \leq n + 1$ exists such that

1. for each $1 \leq j < i$, we have $R_{l}^{i} \models \Delta^j \lor L^j$, and
2. $i \leq n$ i.e., $i$ is an index of a clause from $N_\ell$ implies $\Delta^i \lor L^i \geq_{l} \Delta$.

**Proof.** Assume that $\Gamma \rightarrow \Delta \in N_\ell$ holds. If $\Gamma \rightarrow \Delta$ satisfies condition 1 of Definition 4, then we clearly have $R_{l}^{i} \models \Delta$. Assume that $\Gamma \rightarrow \Delta$ satisfies condition 2 of Definition 4 due to some clause $\Gamma^j \rightarrow \Delta^j \lor L^j \in N_\ell$ such that $\Gamma^j \subseteq \Gamma$ and $\Delta^j \lor \{L^j\} \subseteq \Delta$ hold; the latter clearly implies $\Delta \geq_{l} \Delta^j \lor L^j$. Let $i$ be an integer satisfying this lemma’s assumption. If $i = n + 1$, then we clearly have $j < i$; otherwise, $\Delta^i \lor L^i \geq_{l} \Delta$ implies $\Delta^i \lor L^i \geq_{l} \Delta^j \lor L^j$, and so we also have $j < i$. But then, by the lemma assumption we have $R_{l}^{i} \models \Delta^j \lor L^j$, which implies $R_{l}^{i} \models \Delta$, as required.
Lemma 6. For each $\Gamma \to \Delta \in N_i$, we have $R^i_1 \models \Delta$.

Proof. For the sake of a contraction, choose $C^i = \Gamma^i \to \Delta^i \vee L^i$ as the clause in the sequence of clauses from Appendix C.1.2 with the smallest $i$ such that $R^*_i \not\models \Delta^i \vee L^i$; please recall that $L^i \gg_1 \Delta^i$ and that $L^i = L^i \circ r^i$ with $\circ \in \{\approx, \not\approx\}$. The way we choose $i$ ensures that $i$ satisfies condition 1 of Lemma 5. Now by the definition of $N_i$, a clause $\Gamma \to \Delta \vee L \in S_v$ exists such that

$$\Gamma\alpha_i = \Gamma^i \subseteq \Gamma_i, \quad \Delta\alpha_i = \Delta^i, \quad L\alpha_i = L^i, \quad \text{and} \quad \Delta \not\preceq_v L. \quad (67)$$

We next prove the claim of this lemma by considering the possible forms of $L^i$.

Assume $L^i = l^i \approx r^i$ with $l^i = r^i$. But then, we have $R^*_i \models L^i$, which contradicts our assumption that $R^*_i \not\models \Delta^i \vee L^i$.

Assume $L^i = l^i \approx r^i$ with $l^i \gg_t r^i$. By the definition of $\gg_t$, we have $l \gg_v r$. We first show that $(R^*_i)^{-1} \not\models \Delta^i \vee L^i$ holds; towards this goal, note that, for each equality $l \approx r \in \Delta^i \vee L^i$, properties $R^*_i \not\models l \approx r$ and $R^*_i \not\models i \approx r$ imply $(R^*_i)^{-1} \not\models l \approx r$; and for each inequality $l \not\approx r \in \Delta^i$, Lemma 3 and $R^*_i \not\models l \not\approx r$ imply $(R^*_i)^{-1} \not\models l \not\approx r$. Thus, clause $C^i$ satisfies conditions R1 and R2; however, since $R^*_i \not\models l \approx r$, clause $C^i$ is not generational and thus either condition R3 or condition R4 are not satisfied. We next consider each of these two possibilities.

- Condition R3 does not hold—that is, $l^i$ is reducible by $R^*_i$. By the definition of reducibility, a position $p$ and a generative clause $C^j = \Gamma^j \to \Delta^j \vee L^j$ exist such that $j < i$ and $l^i|_p = l^j$. Due to $j < i$, we have $l^i \approx r^i \gg_t l^j \approx r^j$; together with $l^j \approx r^j \gg_t \Delta^j$, we have $l^i \approx r^i \gg_t \Delta^j$. Lemma 4 ensures $R^*_i \not\models \Delta^j$, and the definition of $N_i$ ensures that a clause $\Gamma' \to \Delta' \vee L' \approx r' \in S_v$ exists such that

$$\Gamma'\sigma_i = \Gamma^j \subseteq \Gamma_i, \quad \Delta'\sigma_i = \Delta^j, \quad l'\sigma_i = l^j, \quad r'\sigma_i = r^j, \quad \Delta' \not\preceq_v \Delta' \approx r' \quad \text{and} \quad l' \gg_v r'. \quad (68)$$

By the assumption of Theorem 2, the Eq rule is not applicable to (67) and (68), and so $\Gamma \land \Gamma' \to \Delta \land \Delta' \vee L'[r^j|_p] \approx r \in N_i$. Let $\Delta'' = \Delta^i \land \Delta^j \vee L'[r^j|_p] \approx r^i$. Then clearly $\Gamma\sigma_i \cup \Gamma'\sigma_i \subseteq \Gamma_i$, so Lemma 2 ensures that $\Gamma \land \Gamma' \to \Delta'' \in N_i$ holds. Set $R^*_i$ is a congruence, so $l^i[r^j|_p] \approx r^i \not\in R^*_i$ holds, and therefore $R^*_i \not\models \Delta''$ holds. Finally, $\gg_t$ is a simplification order, which ensures $l^i \approx r^i \gg_t l^i[r^j|_p] \approx r^i$; together with $l^i \approx r^i \gg_t \Delta^j$ and $l^i \approx r^i \gg_t \Delta^j$, we have $l^i \approx r^i \gg_t \Delta''$. But then, Lemma 5 implies $R^*_i \models \Delta''$, which is a contradiction.

- Condition R4 does not hold. Then, some term $s$ exists such that $l^i \approx s \in \Delta^i$ and $s \approx r^i \in (R^*_i)^{-1}$. Due to $R^*_i \not\models \Delta^i$, we have $s \approx r^i \in R^*_i$, and so $R^*_i \not\models s \not\approx r^i$. Furthermore, $\Delta \land L$ is of the form $\Delta' \land L \approx r \land L' \approx r'$ such that

$$l\sigma_i = l^i, \quad r\sigma_i = s, \quad l'\sigma_i = l^i, \quad \text{and} \quad r'\sigma_i = r^i. \quad (69)$$

But then, we clearly have $l' = l$. By the assumption of Theorem 2, the Factor rule is not applicable to $\Gamma \to \Delta \land L$, and so we have $\Gamma \to \Delta' \land L \not\models r' \land L' \approx r' \in S_v$. Let $\Delta'' = \Delta'\land s \not\approx r^i \land l' \approx r^i$. But then, $\Gamma\sigma_i \subseteq \Gamma_i$ and Lemma 2 ensure that $\Gamma\sigma_i \to \Delta'' \in N_i$ holds. By all the previous observations, we have $R^*_i \not\models \Delta''$. Moreover, $l^i \gg_t r^i$ and $l^i \gg_t s$ imply $l^i \approx r^i \gg_t s \approx r^i$; thus, $\Delta' \land L \approx r \land L' \approx r'$ holds. But then, Lemma 5 implies $R^*_i \models \Delta''$, which is a contradiction.
Assume \( L^i = l^i \neq r^i \) with \( l^i = r^i \). Then, literal \( L \) is of the form \( l \neq r \) such that \( l_o \neq r_q \). But then, \( l^i = r^i \) implies \( l = r \). By the assumption of Theorem 2, the lneq rule is not applicable to clause \( \Gamma \Rightarrow \Delta \vee L \), and so we have \( \Gamma \Rightarrow \Delta \in \mathcal{S}_o \).

Since \( \mathcal{F}_o \subseteq \mathcal{F} \), by Lemma 2 we have \( \Gamma^i \Rightarrow \Delta^i \in \mathcal{N}_i \). Clearly, \( \Delta^i \vee l^i \neq r^i >_i \Delta^i \), and so Lemma 5 implies \( R_i^o \models \Delta^i \), which is a contradiction.

Assume \( L^i = l^i \neq r^i \) with \( l^i >_i r^i \). Lemma 3 ensures \( (R_i^{o-1})^* \models l^i \neq r^i \); hence, \( l^i \) is reducible by \( R_i^{o-1} \). By the definition of reducibility, a position \( p \) and a generative clause \( C^i = \Gamma^i \Rightarrow \Delta^i \vee l^i \approx r^i \) exist such that \( j < i \) and \( l^i|_p = l^i \). Due to \( j < i \), we have \( l^i \neq r^i >_i l^i \approx r^i >_i \Delta^i \). Lemma 4 ensures \( \Delta^i \neq l^i \), and the definition of \( N_i \) ensures that a clause \( \Gamma^i \Rightarrow \Delta^i \vee l^i \approx r^i \in \mathcal{S}_o \) exists satisfying (68), as in the first case.

By the assumption of Theorem 2, the Eq rule is not applicable to clauses (67) and (68), and so \( \Gamma \Rightarrow \Gamma^i \Rightarrow \Delta \vee \Delta^i \vee l^i[r^i] \neq r^i \in \mathcal{S}_o \). Let \( \Delta'' = \Delta^i \vee \Delta^i \vee l^i[r^i] \neq r^i \). We clearly have \( \Gamma^i \Rightarrow \Delta'' \in \mathcal{N}_i \), so by Lemma 2 we have \( \Gamma^i \Rightarrow \Gamma^i \Rightarrow \Delta'' \in \mathcal{N}_i \). Since \( R_i^o \) is a congruence, we have \( R_i^o \models \Gamma^i \Rightarrow \Delta'' \neq r^i \), and therefore \( R_i^o \models \Delta'' \). Finally, \( >_i \) is a simplification order, so \( l^i \neq r^i >_i l^i[r^i] \); together with \( l^i \approx r^i >_i \Delta^i \) and \( l^i \approx r^i >_i \Delta^i \), we have \( l^i \approx r^i >_i \Delta'' \). But then, Lemma 5 implies \( R_i^o \models \Delta'' \), which is a contradiction.

**Lemma 7.** For each clause \( \Gamma \Rightarrow \Delta \) with \( \Gamma \Rightarrow \Delta \in \mathcal{N}_i \), we have \( R_i^o \models \Delta \). □

**Proof.** Apply Lemma 5 for \( i = n + 1 \) and Lemma 6. □

**Lemma 8.** \( R_i^o \models \Gamma_i \Rightarrow \Delta_i \).

**Proof.** For \( R_i^o \models \Gamma_i \), note that condition L2 ensures \( \Gamma_i \Rightarrow A \in \mathcal{N}_i \), and so Lemma 7 ensures \( R_i^o \models A \) for each atom \( A \in \Gamma_i \).

For \( R_i^o \models \Delta_i \), assume for the sake of a contradiction that an atom \( A \in \Delta_i \) exists such that \( R_i^o \models A \). Then, a generative clause \( C^i = \Gamma^i \Rightarrow \Delta^i \vee l^i \approx r^i \in \mathcal{N}_i \) and a position \( p \) exist such that \( A|_p = l^i \); let \( \Delta = \Delta^i \vee l^i \approx r^i \). Since \( >_i \) is a simplification order and \( l^i >_i r^i \), we have \( A \geq l \approx r \), but then, since \( l^i \approx r^i >_i \Delta^i \), we have \( A \geq \Delta \).

We next consider an arbitrary literal \( l \circ r \in \Delta \) with \( \circ \in \{ \approx, \neq \} \) and \( l \geq r \). Literal \( l \circ r \) is obtained by grounding a context literal by \( \sigma_i \), and so \( l \) cannot be \( t \) or \( t' \). Moreover, \( A \geq l \circ r \) implies \( A \geq l \); but then, by condition O2 we have \( l \approx t \in \Delta_i \) and \( r = t \). We thus have \( \Delta \subseteq \Delta_i \); but then, \( \Gamma^i \subseteq \Gamma_i \) implies that \( \Gamma_i \Rightarrow \Delta_i \in \mathcal{N}_i \) holds, which contradicts condition L1. □

**C.2 Interpreting the Ontology \( \mathcal{O} \)**

We now show how to combine the rewrite systems \( R_i \) from Appendix C.1 into a single rewrite system \( R \) such that \( R^* \models \mathcal{O} \) and \( R^* \models \Gamma_Q \Rightarrow \Delta_Q \).

**C.2.1 Unfolding the Context Structure**

We construct \( R \) by a partial induction over the terms in \( \mathcal{T} \). We define several partial functions: function \( X \) maps a term \( t \) to a context \( X_t \in \mathcal{V} \); functions \( \Gamma \) and \( \Delta \) assign to a term \( t \) a conjunction \( \Gamma_t \) and a disjunction \( \Delta_t \), respectively, of atoms; and function \( R \) maps each term into a model fragment \( R_t \) for \( t, X_t, \Gamma_t, \) and \( \Delta_t \).
The proof is by induction on the structure of terms 

\[ X_c = q \]  

(70)

\[ \Gamma_c = \Gamma_Q \sigma_c \]  

(71)

\[ \Delta_c = \Delta_Q \sigma_c \]  

(72)

\[ R_c = \text{the model fragment for } c, \ q, \ \Gamma_c, \text{ and } \Delta_c \]  

(73)

M2. For the inductive step, assume that \( X_{\nu} \) has already been defined, and consider an arbitrary function symbol \( f \in \mathcal{F} \) such that \( f(t') \) is \( R_{\nu} \)-irreducible. Let \( u = X_{\nu} \) and \( t = f(t') \). We have two possibilities.

M2.a. Term \( t \) occurs in \( R_{\nu} \). Then, term \( t = f(t') \) was generated in \( R_{\nu} \) by some ground clause \( C = \Gamma \rightarrow \Delta \lor L \in \mathcal{N}_{\nu} \) such that \( L >_t \Delta \) and \( f(t') \) occurs in \( L \). By the definition of \( N_{\nu} \), then a clause \( C' = \Gamma' \rightarrow \Delta' \lor L' \in \mathcal{S}_u \) exists such that \( C = C' \sigma_c \) and \( L' \) contains \( f(x) \); moreover, \( L >_{\nu} \Delta \) implies \( \Delta' \not\succeq_u L' \). The Succ and Core rules are not applicable to \( \mathcal{D} \), so we can choose a context \( v \in \mathcal{V} \) such that \( \langle u, v, f \rangle \in \mathcal{E} \) and \( A \rightarrow A \in \mathcal{S}_v \) for each \( A \in K_2 \), where \( K_2 \) is as in the Succ rule. We define the following:

\[ X_t = v \]  

(74)

\[ \Gamma_t = R_{\nu} \cap \mathcal{S}_u \]  

(75)

\[ \Delta_t = \text{Pr}_t \setminus R_{\nu} \]  

(76)

\[ R_t = \text{the model fragment for } t, v, \Gamma_t, \text{ and } \Delta_t \]  

(77)

M2.b. Term \( t \) does not occur in \( R_{\nu} \). Then, let \( R_t = \{ t \Rightarrow c \} \), and we do not define any other functions for \( t \).

Finally, let \( R \) be the rewrite system defined by \( R = \bigcup R_t \).

**Lemma 9.** The model fragments \( R_c \) and \( R_t \) constructed in lines (73) and (77) satisfy conditions L1 through L3 in Appendix C.1.

**Proof.** The proof is by induction on the structure of terms \( t \in \text{dom}(X) \). For \( t = c \), conditions L1 through L3 hold directly from conditions C1 through C3 of Theorem 2. We next assume that the lemma holds for some term \( t' \in \text{dom}(X) \), and we consider an arbitrary term \( t \) of the form \( t = f(t') \); let \( u = X_t \) and \( v = X_{t'} \). Condition L2 holds because \( >_t \) is obtained by grounding a context order \( >_t \) that satisfies condition 5 of Definition 3. Condition L3 holds by the way \( u \) is chosen in condition M2 and the fact that \( K_2 \sigma_v \subseteq \mathcal{S}_u \). We next show that condition L1 holds.

For the sake of a contradiction, assume that \( \Gamma_t \rightarrow \Delta_t \notin \mathcal{N}_t \) holds. Set \( \mathcal{S}_u \) contains only atoms and we have \( \Gamma_t \subseteq \mathcal{S}_u \) due to (76); therefore, \( \Gamma_t \rightarrow \Delta_t \notin \mathcal{N}_t \) necessarily holds due to condition 2 of Definition 4, and therefore set \( \mathcal{N}_t \) contains a clause

\[ \bigwedge_{i=1}^m A_i \rightarrow \bigvee_{i=m+1}^{m+n} A_i \quad \text{with} \quad \{ A_i \mid 1 \leq i \leq m \} \subseteq \Gamma_t \quad \text{and} \quad \{ A_i \mid m+1 \leq i \leq m+n \} \subseteq \Delta_t \subseteq \text{Pr}_t. \]  

(78)

By the definition of \( \mathcal{N}_t \), set \( \mathcal{S}_v \) contains a clause

\[ \bigwedge_{i=1}^m A'_i \rightarrow \bigvee_{i=m+1}^{m+n} A'_i \quad \text{where} \quad A_i = A'_i \sigma_t \text{ and } A'_i \in \text{Pr}(\mathcal{O}) \text{ for each } i. \]  

(79)
Consider an arbitrary atom \( A_i \) with \( 1 \leq i \leq m \). By condition M2, terms \( t \) and \( t' \) are irreducible by \( R_{t'} \); but then, since \( \Gamma_i \subseteq R_{t'} \) holds by (75), atom \( A_i \) was generated in \( R_{t'} \) by a generative clause (where subscript \( i \) does not necessarily indicate the position of the clause in sequence of clauses from Appendix C.1.2)

\[
\Gamma_i \rightarrow \Delta_i \lor A_i \quad \text{with} \quad A_i \succ \Delta_i.
\] (80)

By the definition of \( N_{t'} \), set \( S_u \) contains a clause

\[
\Gamma_i \rightarrow \Delta_i' \lor A_i' \quad \text{where} \quad \Gamma_i = \Gamma_i \sigma_i, \quad \Delta_i = \Delta_i' \sigma_i, \quad \text{and} \quad \Delta_i' \notin vA_i' \quad \text{for} \quad 1 \leq i \leq m.
\] (81)

The Pred rule is not applicable to (79) and (81) so (82) holds; together with Lemma 2, this ensures

\[
\bigwedge_{i=1}^{m} \Gamma_i \rightarrow \bigvee_{i=1}^{m} \Delta_i' \lor \bigvee_{i=m+1}^{m+n} A_i' \sigma \in S_u \quad \text{for} \quad \sigma = \{ x \mapsto f(x), \ y \mapsto x \}
\] (82)

\[
\bigwedge_{i=1}^{m} \Gamma_i \rightarrow \bigvee_{i=1}^{m} \Delta_i \lor \bigvee_{i=m+1}^{m+n} A_i \hat{\sigma} \in N_{t'}
\] (83)

By Lemma 4, we have \( R_{t'} \not\models \Delta_i \); and (76) ensures that \( R_{t'} \not\models A_i \) for each \( m + 1 \leq i \leq m + n \); however, this contradicts (83) and Lemma 7. \( \square \)

**C.2.2 Termination, Confluence, and Compatibility**

**Lemma 10.** The rewrite system \( R \) is Church-Rosser.

**Proof.** We show that \( R \) is terminating and locally confluent, and thus Church-Rosser. In the proof of the former, we use a total simplification order \( \succ \) on all ground \( \mathcal{F} \)- and \( \mathcal{P} \)-terms defined as follows. We extend the precedence \( \succ \) from Definition 3 to all \( \mathcal{F} \)- and \( \mathcal{P} \)-symbols in an arbitrary way, but ensuring that constant \( t \) is smallest in the order; then, let \( \succ \) be a lexicographic path order [4] over such \( \succ \). It is well known that such \( \succ \) is a simplification order, and that it satisfies the following properties for each \( \mathcal{F} \)-term \( t \) with predecessor \( t' \) (if one exists), all function symbols \( f, g \in \mathcal{F} \), and each \( \mathcal{P} \)-term \( A \):

- \( f(t) \succ t \succ t' \),
- \( f \succ g \) implies \( f(t) \succ g(t) \), and
- \( A \succ t \).

Thus, conditions 1 and 2 of Definition 3 and the manner in which context orders are grounded in Appendix C.1.1 clearly ensure that, for each \( \mathcal{F} \)-term \( t \in \text{dom}(X) \) and for all terms \( s_1 \) and \( s_2 \) from the \( \mathcal{F} \)-neighbourhood of \( t \) with \( s_1 \succ s_2 \), we have \( s_1 \succ s_2 \).

We next show that \( R \) is terminating by arguing that each rule in \( R \) is embedded in \( \succ \). To this end, consider an arbitrary rule \( l \Rightarrow r \in R \). Clearly, a term \( t \in \text{dom}(R) \) exists such that \( l \Rightarrow r \in R \). This rule is obtained from a head \( l \approx r \) of a clause in \( N_t \), and condition R2 of the definition of \( R_t \) ensures that \( l \succ r \). Moreover, \( l \approx r \) is obtained by grounding a context literal with \( \sigma_i \), so we have the following possible forms of \( l \approx r \):

- Terms \( l \) and \( r \) are both from the \( \mathcal{F} \)-neighbourhood of \( t \). Then, \( l \succ r \) implies \( l \succ r \).
- Terms \( l \approx r = A \approx t \) for \( A \) a \( \mathcal{P} \)-term. Then, \( A \succ t \) since \( t \) is smallest in \( \succ \).
We next show that \( R \) is left-reduced. For the sake of a contradiction, assume that a rule \( l \Rightarrow r \in R \) exists such that \( l \) is reducible by \( R' = R \setminus \{ l \Rightarrow r \} \). Let \( p \) be the ‘deepest’ position at which some rule in \( R' \) reduces \( l \) (i.e., no rule in \( R' \) reduces \( l \) at position below \( p \)), and let \( l' \Rightarrow r' \in R' \) be the rule that reduces \( l \) at position \( p \); thus, \( l' = l|_p \). By the definition of \( R \), we have \( l' \Rightarrow r' \in R_t \) where \( t \) can be as follows.

- Term \( t \) is handled in condition M2.a. Then \( l' \Rightarrow r' \) is generated by an equality \( l' \approx r' \) in the head of a generative clause, and so \( l' \) is of the form \( f(t) \). Thus, \( f(t) \) is reducible by \( R_t \), which contradicts condition M2 from the construction of \( R \).

- Term \( t \) is handled in condition M2.b. Then \( l' = t \); moreover, \( R' \) does not contain \( t \) by the construction of \( R \), which contradicts the assumption that \( l' \Rightarrow r' \in R' \). \( \square \)

**Lemma 11.** For each term \( t \), each \( f \in \mathcal{F} \), and each atom \( A \in Su_t \cup Pr_{f(t)} \) such that \( A \in R^* \) and all \( \mathcal{F} \)-terms in \( A \) are irreducible by \( R \), we have \( A \in R_t^* \).

**Proof.** Let \( t \) be a term, let \( f \in \mathcal{F} \) be a function symbol, and let \( A \in Su_t \cup Pr_{f(t)} \) be an atom such that all \( \mathcal{F} \)-terms in \( A \) are irreducible by \( R \); the latter ensures \( A \not\Rightarrow t \in R \). We next consider the possible forms of \( A \).

Assume \( A \in Su_t \). By the definition of \( Su_t \) in (65) and the fact that \( Su(O) \) contains only atoms of the form \( A(x) \), \( S(x, y) \), and \( S(y, x) \), atom \( A \) can be of the form \( A(t) \), \( S(t, t') \), or \( S(t', t) \), for \( t' \) the predecessor of \( t \) (if one exists). By the form of the generative clauses, we clearly have \( A \in R_t^* \) or \( A \in R_t^* \). Now assume \( A \in R_t^* \). Due to \( A \in Su_t \) and the definition of \( I_t \) in (75), we have \( A \in I_t \). Lemma 8 ensures that \( R_t^* \not\Rightarrow I_t \rightarrow \Delta_t \). But then, we have \( A \in R_t^* \), as required.

Assume \( A \in Pr_{f(t)} \). By the definition of \( Pr_{f(t)} \) in (66) and the fact that \( Pr(O) \) contains only atoms of the form \( A(y) \), \( S(y, x) \), and \( S(x, y) \), atom \( A \) can be of the form \( A(t) \), \( S(t, f(t)) \), or \( S(f(t), t) \). By the form of the generative clauses, we clearly have \( A \in R_t^* \) or \( A \in R_t^* \). Assume for the sake of a contradiction that \( A \not\in R_t^* \), but \( A \in R_t^* \). Due to \( A \in Pr_{f(t)} \) and the definition of \( \Delta_{f(t)} \) in (76), we have \( A \in \Delta_{f(t)} \); due to Lemma 8, we have \( R_t^* \not\Rightarrow \Delta_{f(t)} \rightarrow \Delta_{f(t)} \); therefore, we have \( A \not\in R_t^* \), which is a contradiction. \( \square \)

**Lemma 12.** Let \( s_1 \) and \( s_2 \) be DL-terms, and let \( \tau \) be a substitution irreducible by \( R \) such that \( s_1 \tau \) and \( s_2 \tau \) are ground and each \( \tau(z_i) \) (if defined) is in the \( \mathcal{F} \)-neighbourhood of \( \tau(x) \). Then, for \( \phi \in \{ \approx, \neq \} \), if \( R_{\tau(x)}^* \models s_1 \tau \phi s_2 \tau \), then \( R^* \models s_1 \tau \phi s_2 \tau \).

**Proof.** Let \( s_1 \) and \( s_2 \) be as stated above, let \( t = \tau(x) \), and let \( t' \) be the predecessor of \( t \) (if one exists). Since \( t \) is irreducible by \( R \), rewrite system \( R_t \) has been defined in Appendix C.2.1. We next consider the possible forms of \( \phi \).

- Assume \( \phi = \approx. \) But then, \( R_t \subseteq R \) and \( R_t^* \models s_1 \tau \approx s_2 \tau \) imply \( R^* \models s_1 \tau \approx s_2 \tau \).

- Assume \( \phi = \neq. \) Let \( s_1' \) and \( s_2' \) be the normal forms of \( s_1 \tau \) and \( s_2 \tau \), respectively, w.r.t. \( R_t \). Due to the shape of DL-literals, \( s_1 \) and \( s_2 \) can be of the form \( f(x) \) or \( z_i \); therefore, \( s_1 \tau \) and \( s_2 \tau \) are of the form \( f(t) \) or \( t' \). Term \( t \) is irreducible by \( R \), and therefore \( t' \) is irreducible by \( R \) as well. Furthermore, due to the shape of context terms, the only rewrite system where \( f(t) \) could occur on the left-hand side of a rewrite rule is \( R_t \). Consequently, \( f(t) \) is irreducible by \( R \) as well. But then, \( s_1' \) and \( s_2' \) are the normal forms of \( s_1 \tau \) and \( s_2 \tau \), respectively, w.r.t. \( R \); thus, \( R^* \models s_1' \neq s_2' \), and therefore we have \( R^* \models s_1 \tau \neq s_2 \tau \), as required. \( \square \)
Lemma 13. For each DL-clause $\Gamma \rightarrow \Delta \in \mathcal{O}$, we have $R^* \models \Gamma \rightarrow \Delta$.

Proof. Consider an arbitrary DL-clause $\Gamma \rightarrow \Delta \in \mathcal{O}$ of the following form:

$$\bigwedge_{i=1}^{n} A_i \rightarrow \Delta \quad (84)$$

Let $\tau'$ be an arbitrary substitution such that $\Gamma \tau' \rightarrow \Delta \tau'$ is ground, and let $\tau$ be the substitution obtained from $\tau'$ by replacing each ground term with its normal form w.r.t. $R$. Since $R^*$ is a congruence, we have $R^* \models \Gamma \tau' \rightarrow \Delta \tau'$ if and only if $R^* \models \Gamma \tau \rightarrow \Delta \tau$. We next assume that $R^* \models \Gamma \tau$, and we show that $R^* \models \Delta \tau$ holds as well.

Consider an arbitrary atom $A_i \in \Gamma$. The definition of DL-clauses, $A_i$ is of the form $B(x), S(x, z_j)$, or $S(z_j, x)$. Substitution $\tau$ is irreducible by $R$, and so all $F$-terms in $A\tau$ are irreducible by $R$; but then, $A_i \tau \in R^*$ clearly implies $A_i \tau \Rightarrow t \in R$. Each such rule is obtained from a generative clause so $A_i \tau$ is of the form $B(t_x), S(t_x, f(t))$, $S(f(t), t)$, or $S(t_x, t')$, where $t = \tau(x)$ and $t'$ is the predecessor of $t$ (if it exists). We next prove that $A_i \tau \in Su_i \cup Pr_{f(t)}$ holds by considering the possible forms of $A_i$.

- $A_i = B(x)$, so $A_i \tau = B(t)$. Thus, $B(x) \in Su(\mathcal{O})$ and so $B(t) \in Su_2$.
- $A_i = S(x, z_j)$, so $A_i \tau$ is of the form $S(t_x, t')$ or $S(t_x, f(t))$. Thus, $S(x, y) \in Su(\mathcal{O})$ and so $S(t_x, t') \in Su_1$, and $S(t_x, f(t)) \in Pr(\mathcal{O})$ and so $S(t_x, f(t)) \in Pr_{f(t)}$.
- $A_i = S(z_j, x)$, so $A_i \tau$ is of the form $S(t_x, t)$ or $S(f(t), t)$. Thus, $S(y, x) \in Su(\mathcal{O})$ and so $S(t_x, t') \in Su_1$, and $R(x, y) \in Pr(\mathcal{O})$ and so $S(f(t), t) \in Pr_{f(t)}$.

Lemma 11 then implies $A_i \tau \in R_i$, and so $N_i$ contains a generative clause of the form (85). Now let $v = X_i$; by the definition of $N_i$, set $S_i$ contains a clause of the form (86).

$$\Gamma_i \rightarrow \Delta_i \lor A_i \text{ with } A_i \triangleright_i \Delta_i \text{ and } \Gamma_i \subseteq \Gamma_i \quad (85)$$

$$\Gamma_i \rightarrow \Delta'_i \lor A'_i \text{ with } \Delta'_i \notin v A_i \text{ and } \Gamma_i \sigma_i = \Gamma_i, \Delta'_i \sigma_i = \Delta_i, \text{ and } A'_i \sigma_i = A_i \quad (86)$$

The Hyper rule is not applicable to (84) and (86), and therefore (87) holds, where $\sigma$ is the substitution obtained from $\tau$ by replacing each occurrence of $t$ (possibly nested in another term) with $x$. Finally, Lemma 2 ensures that (88) holds as well.

$$\bigwedge_{i=1}^{n} \Gamma_i \rightarrow \Delta \lor \bigvee_{i=1}^{n} \Delta'_i \in S_i \quad (87)$$

$$\bigwedge_{i=1}^{n} \Gamma_i \rightarrow \Delta \lor \bigvee_{i=1}^{n} \Delta_i \in N_i \quad (88)$$

Now (88) and Lemma 7 imply $R^*_i \models \Delta \lor \bigvee_{i=1}^{n} \Delta_i$, but Lemma 4 implies $R^*_i \not\models \Delta_i$; therefore, we have $R^*_i \not\models \Delta \tau$. Finally, Lemma 12 ensures $R^* \models \Delta \tau$, as required. □

Lemma 14. $R^* \not\models \Gamma_Q \rightarrow \Delta_Q$.

Proof. The claim clearly follows from $R^* \not\models \Gamma_c \rightarrow \Delta_c$. Note that Lemma 8 ensures $R^*_c \not\models \Gamma_c \rightarrow \Delta_c$; thus, $R^*_c \models \Gamma_c$ and $R^*_c \not\models \Delta_c$. The former observation and Lemma 12 ensure that $R^* \models \Gamma_c$ holds. Moreover, for each atom $B(x) \in \Delta_Q$, Definition 2 ensures $B(y) \in Pr(\mathcal{O})$; thus, for each $f \in \mathcal{F}$, we have $B(c) \in Pr_{f(c)}$, and so the contrapositive of Lemma 11 ensures $R^* \not\models B(c)$. Thus, $R^* \not\models \Delta_c$ holds, as required. □