Nominal Schema Absorption

Andreas Steigmiller* and **Birte Glimm**
Ulm University, Germany
<first name>.<last name>@uni-ulm.de

Thorsten Liebig
derivo GmbH, Germany
liebig@derivo.de

Abstract

Nominal schemas have recently been introduced as a new approach for the integration of DL-safe rules into the Description Logic framework. The efficient processing of knowledge bases with nominal schemas remains, however, challenging. We address this by extending the well-known optimisation of absorption as well as the standard tableau calculus to directly handle the (absorbed) nominal schema axioms. We implement the resulting extension of standard tableau calculi in a novel reasoning system and we integrate further optimisations. In our empirical evaluation, we show the effect of these optimisations and we find that the proposed approach performs well even when compared to other DL reasoners with dedicated rule support.

1 Introduction

We address the problem of an efficient handling of so-called nominal schema axioms in tableau calculi for Description Logics (DLs). Nominal schemas have been introduced recently [Krötzsch *et al.*, 2011] as a feature for expressing arbitrary DL-safe rules (as specified in the W3C standards SWRL [Horrocks *et al.*, 2004] or RIF [Kifer and Boley, 2010]) natively in DLs and, consequently, in OWL ontologies [OWL Working Group, 2009]. Hence, DLs with nominal schemas provide a unified basis for OWL and rules. Although some attempts (see, e.g., [Krisnadhi and Hitzler, 2012]) have been made to improve the performance of tableau calculi when extended with nominal schemas, handling of nominal schemas remains challenging. We tackle this problem by extending the well-know tableau optimisation of absorption [Horrocks and Tobies, 2000]. The resulting calculus extends a standard tableau calculus by additional rules to deal with the absorbed nominal schema axioms and shows a considerable performance improvement over existing techniques.

Nominal schemas extend the nominal constructor that is present in many DLs and which allows for specifying a concept as a singleton set with a named individual as member, e.g., the interpretation of the concept $\{a\}$ consists of the element that represents the named individual a. Nominal schemas introduce a new concept constructor $\{x\}$, where x is a variable that can only be bound to a named individual from the ABox of the knowledge base. This restriction ensures decidability and is common for nominal schemas as well as for SWRL rules.

We use the same running example as Krisnadhi and Hitzler (2012), which describes a conflicting review assignment for an individual who has to review a paper x that has an author y with whom that individual has a joint publication in the same venue z:

$$\exists \mathit{hasReviewAssignment}.(\{x\} \sqcap \exists \mathit{hasAuthor}.\{y\} \sqcap \exists \mathit{atVenue}.\{z\}) \sqcap \exists \mathit{hasSubmittedPaper}.(\exists \mathit{hasAuthor}.\{y\} \sqcap \exists \mathit{atVenue}.\{z\}) \sqsubseteq \exists \mathit{hasConflictingAssignedPaper}.\{x\}.$$

For brevity, we shorten *hasReviewAssignment* to r, *hasAuthor* to a, *atVenue* to v, *hasSubmittedPaper* to s, and *hasConflictingAssignedPaper* to c in the remainder. Obviously, this axiom can neither be directly expressed in a DL knowledge base nor as ordinary DL-safe rule (e.g., if we were to express the complex concepts as role atoms, we would have to introduce a variable for the submitted paper, which then would only bind to known ABox individuals). However, nominal schema axioms can be eliminated by *upfront grounding*, i.e., by replacing nominal schema axioms with all possible grounded axioms obtained by replacing nominal schemas with nominals, where the nominal schemas with the same variable are always replaced by the same nominal. Upfront grounding is, however, very inefficient. For example, a nominal schema axiom with 3 variables can be grounded for a knowledge base with 100 ABox individuals in 100^3 different ways, which is prohibitive even for small examples.

A promising approach for efficient reasoning in OWL DL ontologies extended with nominal schemas, i.e., $\mathcal{SROIQV}$ knowledge bases with $\mathcal{V}$ denoting nominal schemas, is to adapt established tableau algorithms (e.g., [Horrocks *et al.*, 2006]), which are dominantly used for sound and complete reasoning systems for expressive DLs. One such approach extends a tableau algorithm such that grounding is *delayed* until it is *required* [Krisnadhi and Hitzler, 2012]. However, this requires significant changes to the tableau algorithm and, thus, to existing optimisations, which are crucial for a reasonable performance on real-world ontologies. Furthermore, it is not clear in which way concepts have to be grounded

*The first author acknowledges the support of the doctoral scholarship under the Postgraduate Scholarships Act of the Land of Baden-Wuerttemberg (LGFG).

to achieve a well-performing implementation and some concepts even cannot be grounded efficiently.

In this paper, we present a novel approach that works by collecting possible bindings for the nominal schema variables during the application of tableau rules; then, these bindings are used to complete the processing of the nominal schema axioms. For this, we extend the widely used technique of absorption (Section 3) to handle nominal schemas (Section 4.1), and we adapt or add new rules to the tableau calculus (Section 4.2). We further sketch optimisations and empirically evaluate the proposed approach (Section 5), before we conclude (Section 6). Further details and proofs are available in a technical report [Anonymous, 2013].

2 Preliminaries

For brevity, we do not introduce DLs (see, e.g., [Baader *et al.*, 2007]) and we only present our approach for $\mathcal{ALCOIQV}$. Covering $\mathcal{SROIQV}$ is, however, easily possible: role chains (incl. transitivity) can be encoded [Horrocks *et al.*, 2006] and the remaining $\mathcal{SROIQ}$ features are easy to support.

Model construction calculi, such as tableau, decide the consistency of a knowledge base $\mathcal{K}$ by trying to construct an abstraction of a model for $\mathcal{K}$, a so-called "completion graph". A completion graph G is a tuple $(V, E, \mathcal{L}, \dot{\neq})$, where each node $x \in V$ (edge $\langle x, y\rangle \in E$) represents one or more (pairs of) individuals. Each node x (edge $\langle x, y\rangle$) is labelled with a set of concepts (roles), $\mathcal{L}(x)$ ($\mathcal{L}(\langle x, y\rangle)$), which the individuals represented by x ($\langle x, y\rangle$) are instances of. The relation $\dot{\neq}$ records inequalities between nodes.

The algorithm works by initialising the graph with one node for each ABox individual/nominal in the input knowledge base (w.l.o.g. we assume that the ABox is non-empty). Complex concepts are then decomposed using a set of expansion rules, where each rule application can add new concepts to node labels and/or new nodes and edges to the completion graph, thereby explicating the structure of a model. The rules are applied until either the graph is fully expanded (no more rules are applicable), in which case the graph can be used to construct a model that is a *witness* to the consistency of $\mathcal{K}$, or an obvious contradiction (called a *clash*) is discovered (e.g., both C and $\neg C$ in a node label), proving that the completion graph does not correspond to a model. The input knowledge base $\mathcal{K}$ is *consistent* if the rules (some of which are non-deterministic) can be applied such that they build a fully expanded, clash free completion graph.

In order to guarantee that each node of the completion graph indeed satisfies all TBox axioms, one can use a tableau rule that checks, for each general concept inclusion (GCI) $C \sqsubseteq D$, whether C is satisfied for a node v and only then adds D to $\mathcal{L}(v)$. If C is complex, it can, however, be non-trivial to decide whether C is satisfied for v. To guarantee correctness, one treats axioms where C is complex as $\top \sqsubseteq \neg C \sqcup D$. Given that $\top$ is satisfied at each node, the disjunction is then added to the label of each node. In practise, one uses elaborate transformations in a preprocessing step called absorption to avoid axioms of the form $\top \sqsubseteq \neg C \sqcup D$.

The absorption algorithm extracts those conditions of a disjunction for which it can be ensured that if one of these conditions is not satisfied for a node in a completion graph, then at least one alternative of the disjunction is trivially satisfiable. These conditions are then used for expressing the disjunction in such a way that non-determinism can be avoided as much as possible in the tableau algorithm. For example, one would like to avoid treating $\exists r.(A_1 \sqcup A_2) \sqsubseteq \exists s.A$ as $\top \sqsubseteq \forall r.(\neg A_1 \sqcap \neg A_2) \sqcup \exists s.A$. Any node that does *not* have an r-neighbour trivially satisfies $\forall r.(\neg A_1 \sqcap \neg A_2)$ and, hence, the overall disjunction. Thus, we could only add the disjunction to nodes that have at least one r-successor. We can, however, go even further by first identifying nodes that satisfy A_1 or A_2 and then make sure that their r^--neighbour satisfies $\exists s.A$. Hence, the disjunctive axiom can be rewritten into $A_1 \sqsubseteq T$, $A_2 \sqsubseteq T$ and $T \sqsubseteq \forall r^-.(\exists s.A)$, where T is a fresh atomic concept. Here, A_1 and A_2 have been *absorbed* (i.e., moved to the left-hand side of the axiom) and the concept T is used to enforce the semantics of the original axiom. We call $\forall r.(\neg A_1 \sqcap \neg A_2)$ *completely absorbable* since it no longer contributes a disjunct. The goal of the absorption preprocessing step is, therefore, the extraction of such easy to verify conditions that allow for expressing a GCI by possibly several axioms that ideally do not require a disjunction.

For a more complex absorption it is often necessary to join several conditions, say A_1 to A_n. An efficient way to do this is *binary absorption* [Hudek and Weddell, 2006], where two concepts A_1 and A_2 imply a fresh atomic concept T_1 by the axiom $(A_1 \sqcap A_2) \sqsubseteq T_1$. We can then combine T_1 with the next condition A_3 and so on, until $(T_{n-2} \sqcap A_n) \sqsubseteq T_{n-1}$, where T_{n-1} can then be used for further absorption.

3 Absorption Algorithm

Since our handling of nominal schemas is based on absorption methods, we next present an improved variant of a recursive binary absorption algorithm, which we then extend to nominal schemas in the next section. The improvements allow for absorbing parts of the axioms partially without creating additional disjunctions. For example, the TBox axiom $\exists r.(A \sqcap \forall r.C) \sqsubseteq D$ is, without absorption, handled as $\top \sqsubseteq \forall r.(\neg A \sqcup \exists r.\neg C) \sqcup D$. None of the disjuncts can be absorbed completely, but it is nevertheless possible to delay the processing of the disjunction until there is an r-neighbour with the concept A in its label. In order to capture this, the absorption rewrites the axiom such that the disjunction is propagated from a node with A in its label to all r^--neighbours (if there are any), which results in $A \sqsubseteq \forall r^-.(\forall r.(\neg A \sqcup \exists r.\neg C) \sqcup D)$.

In the following, $C_{(i)}, D_{(i)}$ are (possibly complex) concepts, $A_{(i)}, T_{(i)}$ are atomic concepts with $T_{(i)}$ used for fresh concepts and S is a set of concepts. We assume that all concepts are in the well-known negation normal form (NNF) or we use $\mathsf{nnf}(C)$ to transform a concept C to an equivalent one in NNF. Our algorithm uses the following functions to absorb axioms of a TBox $\mathcal{T}$ into a new (global) TBox $\mathcal{T}'$:

- $\mathsf{isCA}(C)$ ($\mathsf{isPA}(C)$) is a recursive function that returns *true* if the concept C is *completely (partially) absorbable* and *false* otherwise. For the base case, C is completely (partially) absorbable if C has the form $\neg\{a\}$ or $\neg A$ ($\neg\{a\}$, $\neg A$, or $\forall r.C'$). For the recursion, C is completely absorbable if

Algorithm 1 collectDisjuncts(C, *absorbable*), returns the absorbable/not absorbable disjuncts of the concept C

1: $S \leftarrow \{C\}$
2: **while** $(C_1 \sqcup C_2) \in S$ **do**
3: $\quad S \leftarrow (S \setminus (C_1 \sqcup C_2)) \cup \{C_1, C_2\}$
4: **end while**
5: **if** *absorbable* $=$ *true* **then return** $\{\ C \in S \mid \mathsf{isPA}(C)\ \}$
6: **else return** $\{\ C \in S \mid \neg\mathsf{isCA}(C)\ \}$
7: **end if**

Algorithm 2 absorbJoined(S), returns the atomic concept that is implied by the join of the absorptions of S

1: $S' \leftarrow \emptyset$
2: **for all** $C \in S$ **do**
3: $\quad A' \leftarrow \mathsf{absorbConcept}(C)$
4: $\quad S' \leftarrow S' \cup \{A'\}$
5: **end for**
6: **while** $A_1 \in S'$ **and** $A_2 \in S'$ **and** $A_1 \neq A_2$ **do**
7: $\quad T \leftarrow$ fresh atomic concept
8: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \cup \{(A_1 \sqcap A_2) \sqsubseteq T\}$
9: $\quad S' \leftarrow (S' \cup \{T\}) \setminus \{A_1, A_2\}$
10: **end while**
11: **if** $S' = \emptyset$ **then return** $\top$
12: **else return** the element $A' \in S'$ $\quad \triangleright\ S'$ is a singleton
13: **end if**

  - $C = \forall r.C'$ and C' is completely absorbable,
  - either $C = C_1 \sqcup C_2$ or $C = C_1 \sqcap C_2$ and both C_1 and C_2 are completely absorbable;

  and C is partially absorbable if

  - $C = C_1 \sqcup C_2$ and C_1 or C_2 are partially absorbable,
  - $C = C_1 \sqcap C_2$ and both C_1 and C_2 are partially absorbable.

  In all other cases, we say that C is neither completely nor partially absorbable, however, the absorption can further be extended to other constructors, e.g., to constructors of more expressive DLs [Anonymous, 2013].
- collectDisjuncts(C, *absorbable*), shown in Algorithm 1, returns the set of (completely or partially) absorbable disjuncts for C if *absorbable* $=$ *true* and the set of not completely absorbable disjuncts otherwise. If C is not a disjunction, then $\{C\}$ itself is returned, in case it conforms to the specified *absorbable* condition.

For simplicity, we assume here that axioms of the form $C \equiv D$ are rewritten into $C \sqsubseteq D$ and $D \sqsubseteq C$. An extension that directly and, hence, more efficiently handles axioms of the form $A \equiv C$ is also possible [Anonymous, 2013].

To absorb the TBox $\mathcal{T}$, we call for each axiom $C \sqsubseteq D$ the function absorbJoined for the set of absorbable disjuncts, i.e., collectDisjuncts(nnf$(\neg C \sqcup D)$, *true*), which returns a fresh atomic concept that is used to imply a disjunction of the non-absorbable disjuncts, i.e., collectDisjuncts(nnf$(\neg C \sqcup D)$, *false*). The methods absorbJoined (Algorithm 2) and absorbConcept (Algorithm 3) are recursively calling each other, whereby absorbJoined is joining several atomic concepts with binary absorption axioms and absorbConcept creates the absorption for a specific concept. For instance, a concept of the form $\forall r.C$ can be absorbed (lines 7–11 of Algorithm 3) by creating a propagation from the atomic concept A_{nb}, which is obtained by the absorption of the concept C, back over the r-edge, to trigger a fresh atomic concept T. Note, if C cannot be absorbed, then absorbJoined returns $\top$ and the axiom $\top \sqsubseteq \forall r^-.T$ is created, which corresponds to $\exists r.\top \sqsubseteq T$ and, thus, is similar to the well known *role absorption* [Tsarkov and Horrocks, 2004].

Algorithm 3 absorbConcept(C), returns the atomic concept for the absorption of the concept C

1: **if** $C = C_1 \sqcap C_2$ **then**
2: $\quad A_1 \leftarrow \mathsf{absorbJoined}(\mathsf{collectDisjuncts}(C_1, \mathit{true}))$
3: $\quad A_2 \leftarrow \mathsf{absorbJoined}(\mathsf{collectDisjuncts}(C_2, \mathit{true}))$
4: $\quad T \leftarrow$ fresh atomic concept
5: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_1 \sqsubseteq T, A_2 \sqsubseteq T\}$
6: $\quad$ **return** T
7: **else if** $C = \forall r.C'$ **then**
8: $\quad A_{nb} \leftarrow \mathsf{absorbJoined}(\mathsf{collectDisjuncts}(C', \mathit{true}))$
9: $\quad T \leftarrow$ fresh atomic concept
10: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall r^-.T\}$
11: $\quad$ **return** T
12: **else if** $C = \neg\{a\}$ **then**
13: $\quad T \leftarrow$ fresh atomic concept
14: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \cup \{\{a\} \sqsubseteq T\}$
15: $\quad$ **return** T
$\quad \ldots$
16: **else return** A $\quad \triangleright\ C$ is of the form $\neg A$
17: **end if**

The absorbJoined function creates binary absorption axioms (Algorithm 2, lines 6-10) for the atomic concepts returned by absorbConcept. Thus, absorbJoined is joining several conditions into one fresh atomic concept, which can be used for further absorption or to initiate the addition of the remaining and non-absorbable part of the axiom. One can further reduce the number of produced axioms by reusing absorption axioms for concepts that occur more then once.

One can show that concept satisfiability is indeed preserved for the absorbed TBox:

Theorem 1 *Let $\mathcal{T}$ denote a TBox, $\mathcal{T}'$ the TBox obtained by absorbing $\mathcal{T}$, and C a concept, then C is satisfiable with respect to $\mathcal{T}$ iff it is satisfiable with respect to $\mathcal{T}'$.*

4 Nominal Schema Absorption

In contrast to DL-safe SWRL rules, the left-hand side of axioms with nominal schemas can be satisfied on arbitrary nodes in the completion graph (even though variables can only bind to nodes that represent individuals/nominals). As a consequence, axioms with nominal schemas can influence arbitrary nodes in the completion graph and, thus, blocking, which ensures termination, easily becomes unsound when typical approaches for rule processing, such as Rete [Forgy, 1982], are used naively. Our approach to overcome this issue is to emulate such rule processing mechanisms by adapted tableau rules, which propagate bindings of variables for concepts through the completion graph. As a nice side-effect,

this propagation means that complex roles can be supported without further adjustments. Analogously to ordinary GCIs, our approach works well if the axioms with nominal schemas have a large absorbable part and, furthermore, most nominal schema variables appear at least once in the absorbable part.

4.1 Absorption of Axioms with Nominal Schemas

The absorption of axioms with nominal schema variables works very similar to the absorption of ordinary axioms. We could directly extend the absorption algorithm to handle the new concept construct, however, to avoid some special cases for conjunctions $C_1 \sqcap C_2$ in an absorbable disjunct, where different nominal schema variables are used in C_1 and C_2, we require that conjunctions in absorbable positions are eliminated. This can be done by duplicating the disjunction that is absorbed and by replacing $C_1 \sqcap C_2$ once with C_1 and once with C_2. For example, the axiom $\{x\} \sqcup A \sqsubseteq \exists r.\{x\}$ is handled as the disjunction $(\neg\{x\} \sqcap \neg A) \sqcup \exists r.\{x\}$ in the absorption and to eliminate $\neg\{x\} \sqcap \neg A$ we replace the original axiom with $\{x\} \sqsubseteq \exists r.\{x\}$ and $A \sqsubseteq \exists r.\{x\}$.

For our absorption algorithm of Section 3, the following two modifications are necessary in order to handle nominal schemas in the remaining axioms:

- $\mathsf{isCA}(C)$ ($\mathsf{isPA}(C)$) is extended to return that a negated occurrence of a nominal schema $\neg\{x\}$ is completely (partially) absorbable.
- $\mathsf{absorbConcept}(C)$ of Algorithm 3 must now also handle a negated occurrence of a nominal schema $\neg\{x\}$ by absorbing it to $O \sqsubseteq \downarrow x.T_x$ for which the fresh atomic concept T_x is returned and O is a special concept that is added to the label of all ABox individuals.

The $\downarrow$ binder operator, as known from Hybrid Logics [Blackburn and Tzakova, 1998], is introduced to actually bind variables to individuals (or nodes in a completion graph). It is handled by a new tableau rule, which adds, for a node a with $\downarrow x.T_x \in \mathcal{L}(a)$, T_x to the label and records that x is bound to a. In the remainder, we assume that knowledge bases contain, for each individual a, an axiom of the form $\{a\} \sqsubseteq O$, where O is a fresh atomic concept. Since the binders are, therefore, only added to ABox individuals (due to axioms of the form $O \sqsubseteq \downarrow x.T_x$), the decidability is retained, whereas the unrestricted extension of a Description Logic with binders easily leads to undecidability of the standard reasoning problems.

Other concepts can be absorbed as before, however, the remaining, non-absorbed part of the axiom, say the disjuncts $D_1, \ldots, D_n$, have to be added with a disjunction that is grounded with those bindings of variables that have been propagated to the last concept from the absorption, say A. In the tableau algorithm this can be done dynamically, e.g., with a new "grounding concept" and a corresponding rule. Therefore, if $D_1, \ldots, D_n$ still contain concepts with nominal schemas, then $A \sqsubseteq gr(D_1 \sqcup \ldots \sqcup D_n)$ has to be added to the TBox, where $gr(\cdot)$ is the new grounding concept. For simplicity, let us assume that $gr(C)$ is always used to add the remaining, non-absorbed part of the axiom, even if C or the axiom does not contain any nominal schemas.

Example 1 *Our running example* $\exists r.(\{x\} \sqcap \exists a.\{y\} \sqcap \exists v.\{z\}) \sqcap \exists s.(\exists a.\{y\} \sqcap \exists v.\{z\}) \sqsubseteq \exists c.\{x\}$ *can be almost completely absorbed into the following axioms:*

$$
\begin{array}{lll}
O \sqsubseteq \downarrow x.T_x & T_z \sqsubseteq \forall v^-.T_2 & (T_1 \sqcap T_2) \sqsubseteq T_3 \\
O \sqsubseteq \downarrow y.T_y & T_3 \sqsubseteq \forall s^-.T_4 & (T_3 \sqcap T_x) \sqsubseteq T_5 \\
O \sqsubseteq \downarrow z.T_z & T_5 \sqsubseteq \forall r^-.T_6 & (T_4 \sqcap T_6) \sqsubseteq T_7. \\
T_y \sqsubseteq \forall a^-.T_1 & T_7 \sqsubseteq gr(\exists c.\{x\}), &
\end{array}
$$

where T_x, T_y, T_z, $T_1, \ldots, T_7$ are fresh atomic concepts. Only $\exists c.\{x\}$ cannot be absorbed and has to be grounded on demand. To keep the example small, we have reused axioms for the absorption of the same concepts, whereas the algorithm of Section 3 would generate for each occurrence of $\neg\{y\}$ and $\neg\{z\}$ a separate binder concept.

Table 1: Tableau rule extensions to propagate mappings

$\forall$-rule:	if	$\forall r.C \in \mathcal{L}(v)$, v not indirectly blocked, there is an r-neighbour w of v with $C \notin \mathcal{L}(w)$ or $\mathcal{B}(\forall r.C, v) \not\subseteq \mathcal{B}(C, w)$
	then	$\mathcal{L}(w) \longrightarrow \mathcal{L}(w) \cup \{C\}$ and $\mathcal{B}(C, w) \longrightarrow \mathcal{B}(C, w) \cup \mathcal{B}(\forall r.C, v)$
$\sqsubseteq_1$-rule:	if	$A \sqsubseteq C \in \mathcal{K}$, $A \in \mathcal{L}(v)$, v not indirectly blocked, and $C \notin \mathcal{L}(v)$ or $\mathcal{B}(A, v) \not\subseteq \mathcal{B}(C, v)$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$ and $\mathcal{B}(C, v) \longrightarrow \mathcal{B}(C, v) \cup \mathcal{B}(A, v)$
$\sqsubseteq_2$-rule:	if	$(A_1 \sqcap A_2) \sqsubseteq C \in \mathcal{K}$, $\{A_1, A_2\} \subseteq \mathcal{L}(v)$, v not indirectly blocked, and 1. $\mathcal{B}(A_1, v) \cup \mathcal{B}(A_2, v) = \emptyset$ and $C \notin \mathcal{L}(v)$, or 2. $(\mathcal{B}(A_1, v) \bowtie^{\epsilon} \mathcal{B}(A_2, v)) \neq \emptyset$ and $C \notin \mathcal{L}(v)$ or $(\mathcal{B}(A_1, v) \bowtie^{\epsilon} \mathcal{B}(A_2, v)) \not\subseteq \mathcal{B}(C, v)$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$ and $\mathcal{B}(C, v) \longrightarrow \mathcal{B}(C, v) \cup (\mathcal{B}(A_1, v) \bowtie^{\epsilon} \mathcal{B}(A_2, v))$
$\downarrow$-rule:	if	$\downarrow x.C \in \mathcal{L}(v)$, v not indirectly blocked, and $C \notin \mathcal{L}(v)$ or $\{x \mapsto v\} \notin \mathcal{B}(C, v)$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$ and $\mathcal{B}(C, v) \longrightarrow \{\{x \mapsto v\}\}$
gr-rule:	if	$gr(C) \in \mathcal{L}(v)$, v not indirectly blocked, there exists a variable mapping $\mu \in \mathsf{comp}^{\mathcal{K}}_{\mathsf{Vars}(C)}(\mathcal{B}(gr(C), v))$ with $C_{[\mu]} \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C_{[\mu]}\}$

4.2 Tableau Algorithm Extensions

We can now extend a standard tableau decision procedure to support (absorbed) nominal schema axioms. Note, we assume that GCIs are handled by two rules: the $\sqsubseteq_1$-rule handles GCIs of the form $A \sqsubseteq C$ (i.e., a non-absorbable axiom $C \sqsubseteq D$ is handled as $\top \sqsubseteq \mathsf{nnf}(\neg C \sqcup D)$) and the $\sqsubseteq_2$-rule handles binary absorption axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$. These rules and the $\forall$-rule (for transitivity support also the $\forall^+$-rule) have to be adapted. The $\downarrow$ binders and $gr(\cdot)$ concepts are handled by new rules. In order to propagate variable bindings, we keep a set of mappings that records bindings for variables, for each concept in a node label.

Definition 1 (Variable Mapping) *A* variable mapping μ *is a (partial) function from variable names to individual names. The set of elements on which μ is defined is the* domain*, written $\mathsf{dom}(\mu)$, of μ. We use ϵ for the empty variable mapping, i.e., $\mathsf{dom}(\epsilon) = \emptyset$. We associate a concept C in the label of a node v with a set of variable mappings, denoted by $\mathcal{B}(C, v)$.*

When clear from the context, we simply write mapping instead of variable mapping in the remainder.

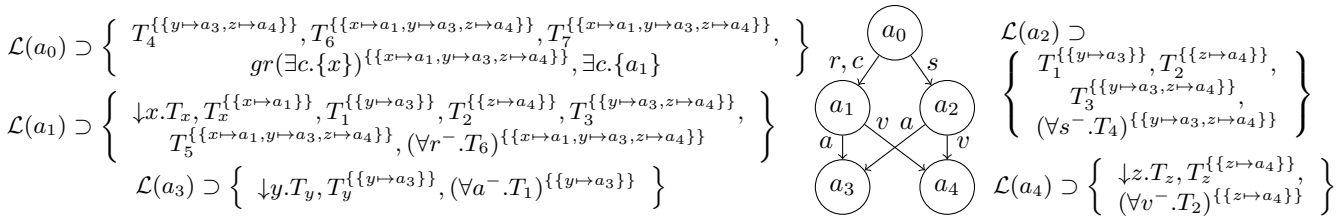

Figure 1: Variable mapping propagation example

Table 1 shows the adapted and new tableau rules and we describe the not so straightforward extensions in more detail below. The mappings have to be propagated by the tableau rules for the concepts and axioms that are used in the absorption. For example, if we apply the adapted $\sqsubseteq_1$-rule to an axiom of the form $A \sqsubseteq C$, we keep the mappings also for the concept C. Note that it is only necessary to extend those rules, which are related to concepts and axioms that are used in the absorption, because if the mappings are propagated to a $gr(\cdot)$ concept, the remaining, non-absorbed part of the axiom is grounded and thus corresponds to an ordinary concept.

Some major adjustments are necessary for the $\sqsubseteq_2$-rule that handles binary absorption axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$. First of all, we want to keep the default behaviour if there are no variable mappings associated to the concept facts for which the rule is applied, i.e., if $\mathcal{B}(A_1, v) \cup \mathcal{B}(A_2, v) = \emptyset$, then we add C to the label of v. In contrast, if $\mathcal{B}(A_1, v) \neq \emptyset$ or $\mathcal{B}(A_2, v) \neq \emptyset$, we propagate the join of the mapping sets to the implied concept. In the case $\mathcal{B}(A_1, v) = \emptyset$ and $\mathcal{B}(A_2, v) \neq \emptyset$, we extend $\mathcal{B}(A_1, v)$ by the empty mapping ϵ so that the join of $\mathcal{B}(A_1, v)$ and $\mathcal{B}(A_2, v)$ results in $\mathcal{B}(A_2, v)$, which is then propagated to C. We proceed analogously for $\mathcal{B}(A_2, v) = \emptyset$ and $\mathcal{B}(A_1, v) \neq \emptyset$. In principle, the join combines variable mappings that map common variables to the same individual name and to point out that the empty sets of mappings are specially handled, we have extended the join operator $\bowtie$ with the superscript ϵ.

Definition 2 (Variable Mapping Join) *Two variable mappings μ_1 and μ_2 are* compatible *if $\mu_1(x) = \mu_2(x)$ for all $x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$. For compatible mappings μ_1 and μ_2, $\mu_1 \cup \mu_2$ is defined as $(\mu_1 \cup \mu_2)(x) = \mu_1(x)$ if $x \in \mathsf{dom}(\mu_1)$, and $(\mu_1 \cup \mu_2)(x) = \mu_2(x)$ otherwise. Given two (possibly empty) sets of variable mappings M_1, M_2, let $M_1^\epsilon = \{\epsilon\}$ ($M_2^\epsilon = \{\epsilon\}$) if $M_1 = \emptyset$ ($M_2 = \emptyset$) and $M_1^\epsilon = M_1$ ($M_2^\epsilon = M_2$) otherwise. The* join *$M_1 \bowtie^\epsilon M_2$ is defined as $\{\mu_1 \cup \mu_2 \mid \mu_1 \in M_1^\epsilon, \mu_2 \in M_2^\epsilon$ and μ_1 is compatible with $\mu_2\} \setminus \{\epsilon\}$.*

For a concept $gr(C)$ the gr-rule grounds C based on the variable mappings associated to $gr(C)$. Since these mappings might not cover all nominal schema variables that occur in C, it is necessary to extend the mappings with every combination of named individuals for the remaining variables. This so-called completion ensures that only fully grounded concepts are added, which can then be handled as ordinary concepts in the completion graph. Therefore, it is also not necessary to further propagate mappings to such newly added concepts.

Definition 3 (Grounding, Completion) *For a concept C, $\mathsf{Vars}(C)$ is the set of nominal schema variables that syntactically occur in C. A concept C is* grounded *if $\mathsf{Vars}(C) = \emptyset$. Let μ be a variable mapping. We write $C_{[\mu]}$ to denote the concept obtained by replacing each nominal schema $\{x\}$ that occurs in C and $x \in \mathsf{dom}(\mu)$ with the nominal $\{\mu(x)\}$.*

Given a set of variables Y and a variable mapping set M with M^ϵ as the extension by the empty mapping ϵ if $M = \emptyset$, the completion $\mathsf{comp}_Y^{\mathcal{K}}(M)$ of M w.r.t. Y and a knowledge base $\mathcal{K}$ *containing the individuals* $\mathsf{Inds}(\mathcal{K})$ *is*

$$\mathsf{comp}_Y^{\mathcal{K}}(M) = \{\mu \cup \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\} \mid \mu \in M^\epsilon, x_1, \ldots, x_n \in (Y \setminus \mathsf{dom}(\mu)), v_1, \ldots, v_n \in \mathsf{Inds}(\mathcal{K})\}.$$

The unrestricted application of generating rules such as the $\exists$-rule can lead to the introduction of infinitely many new tableau nodes. To guarantee termination, one uses a cycle detection technique called *(pairwise) blocking* [Horrocks and Sattler, 1999] that restricts the application of such rules. To apply blocking, we distinguish *blockable nodes* from *nominal nodes*, which have a nominal from the knowledge base in their label. A node v with predecessor v' is *blocked* by a node w with predecessor w', if v, v', w, w' are all blockable and the labels of (i) v and w (ii) v' and w' and (iii) $\langle v', v\rangle$ and $\langle w', w\rangle$ coincide. We extend the standard blocking conditions to also require that the bindings for the concepts in the labels of these nodes coincide.

The completion graph in Figure 1 is obtained in the course of testing the consistency of a knowledge base containing the axioms of Example 1 and the assertions: $r(a_0, a_1)$, $s(a_0, a_2)$, $a(a_1, a_3)$, $v(a_1, a_4)$, $a(a_2, a_3)$, $v(a_2, a_4)$. Note, Figure 1 shows only those concepts and variable mappings (in superscripts) that are relevant for the grounding of new concepts. However, since O and thereby also the binder concepts are added to all ABox individuals, additional variable mappings are automatically created for every ABox individual. The joins of the mapping sets are created in the nodes a_1 and a_2 for the concepts T_3 and T_5 and finally in node a_0 for the concept T_7. Only the variable mapping $\{x \mapsto a_1, y \mapsto a_3, z \mapsto a_4\}$ is propagated to the grounding concept $gr(\exists c.\{x\}) \in \mathcal{L}(a_0)$ and, thus, by replacing the nominal schema $\{x\}$ with the nominal $\{a_1\}$, we have $\exists c.\{a_1\}$ as the only grounded concept. Hence, the individual a_0 is found to have a conflicting review assignment with the paper a_1.

Roughly speaking, it is possible to prove the correctness of our nominal schema absorption technique by a reduction between a completion graph for a TBox with nominal schemas and a standard completion graph for the upfront grounded TBox. Blocking still guarantees termination since only a limited number of variable mappings are introduced.

Theorem 2 *Let $\mathcal{T}$ denote an absorbed TBox (possibly with nominal schema axioms), then a tableau decision procedure (as described above) extended by the rules in Table 1 is a decision procedure for the satisfiability of $\mathcal{T}$.*

Table 2: DL-safe Rules for UOBM-Benchmarks

Name	DL-safe Rule	Matches
R1	*isFirendOf*$(?x,?y)$, *like*$(?x,?z)$, *like*$(?y,?z) \rightarrow$ *hasLink1*$(?x,?y)$	4,037
R2	*isFirendOf*$(?x,?y)$, *takesCourse*$(?x,?z)$, *takesCourse*$(?y,?z) \rightarrow$ *hasLink2*$(?x,?y)$	82
R3	*takesCourse*$(?x,?z)$, *takesCourse*$(?y,?z)$, *hasSameHomeTownWith*$(?x,?y) \rightarrow$ *hasLink3*$(?x,?y)$	940
R4	*hasDoctoralDegreeFrom*$(?x,?z)$, *hasMasterDegreeFrom*$(?x,?w)$, *hasDoctoralDegreeFrom*$(?y,?z)$, *hasMasterDegreeFrom*$(?y,?w)$, *worksFor*$(?x,?v)$, *worksFor*$(?y,?v), \rightarrow$ *hasLink4*$(?x,?y)$	369
R5	*isAdvisedBy*$(?x,?z)$, *isAdvisedBy*$(?y,?z)$, *like*$(?x,?w)$, *like*$(?y,?w)$, *like*$(?z,?w) \rightarrow$ *hasLink5*$(?x,?y)$	286

Table 3: Comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by rules in seconds

Rule	upfront grounding		direct propagation		representative propagation		HermiT	Pellet
			without BC	with BC	without BC	with BC	1.3.7	2.3.0
R1	(10.99)	mem	9.12	7.10	5.06	3.38	31.46	6.33
R2	(10.92)	4.05	3.33	2.33	2.13	2.11	4.79	7.4
R3	(13.33)	3.55	1.98	0.62	2.20	0.76	1.67	142.25
R4	(16.44)	0.30	1.08	0.09	1.06	0.07	1.42	122.85
R5	(time)	–	1.87	0.50	1.80	0.43	28.41	mem

5 Implementation and Evaluation

The techniques are implemented in the novel reasoning system Konclude that supports $\mathcal{SROIQV}$ by (i) upfront grounding and (ii) tableau extensions with different optimisations.

A detailed evaluation can be found in the technical report [Anonymous, 2013]. For brevity, we exemplarily show here some results for the University Ontology Benchmark (UOBM) [Ma *et al.*, 2006] extended by DL-safe rules, which can straightforwardly be expressed as nominal schema axioms. This allows for comparing our system to the DL reasoners HermiT 1.3.7[1] and Pellet 2.3.0 [Sirin *et al.*, 2007]. To the best of our knowledge, these are the only reasoning systems that support DL-safe rules for such expressive ontologies. The used ontology ($\text{UOBM}_1\backslash\text{D}$, data properties removed) has $\mathcal{SHOIN}$ expressivity and consists of $190,093$ axioms, 69 classes, 36 properties, and $25,453$ individuals. All experiments were performed on an Intel Core i7 940 quad core processor running at 2.93 GHz. The reasoners are restricted to use one core and all results are averaged over three runs. Exceeding the time limit of 24 hours is shown as *time* and the memory limit of 10 GB as *mem* in the results.

Table 2 shows the rules and the number of matches for each rule in the consistency check. However, since reasoning with UOBM-1 is non-deterministic, these numbers might vary between different executions and reasoners. Our system requires 1.03 s for preprocessing and 1.09 s for the consistency test for the ontology without rules. Table 3 then shows *the increase* in reasoning time for the ontology with nominal schema axioms. In parenthesis we show the additional preprocessing time for the upfront grounding, which is mostly spend on absorption, lexical normalisation, etc. Upfront grounding fails for R5 since although two variables can be eliminated (see safety condition in [Krötzsch *et al.*, 2011]) it requires $647,855,209$ new axioms. We have also implemented an optimisation where we create a *representative* for a set of variable mappings. Only these representatives are then propagated and considered in the dependency directed backtracking, which saves memory. The direct propagation and the propagation of representatives are depicted (i) with and (ii) without the backward chaining (BC) optimisation, which is used to restrict the creation and propagation of variable mappings, i.e., variable mappings are only created if there is an opportunity to propagate them to a grounding concept. This is realised by additionally absorbing nominal schema axioms, where all nominal schemas are replaced by O, and by using the created atomic concept from the absorption to identify "interesting" individuals with possibly the grounding concept in the label. We then use a back propagation, whereby only binder concepts are activated that are in the scope of these "interesting" individuals. However, for example for rule R2, still nearly all variable mappings have to be created and propagated, and thus, the backward chaining only slightly improves the reasoning time.

Table 3 further shows the reasoning time *increase* for HermiT and Pellet when a rule from Table 2 is added. Without rules our system requires 1.09 s, HermiT 23.24 s, and Pellet 2.22 s for a consistency test (ignoring loading and preprocessing time). With backwards chaining and the propagation of representatives, the reasoning times for our system are significantly faster than HermiT's or Pellet's. HermiT uses, however, significantly less memory than the other systems. This might be because HermiT does not support complex roles, such as *hasSameHomeTownWith* in R3, in the body of rules and its results might be incomplete.

6 Conclusions

We have addressed the problem of practical reasoning with nominal schemas through an extended absorption algorithm and with slight modifications of standard tableau calculi. Our approach "collects" the bindings for nominal schema axioms that have to be grounded and considered for a specific node in the completion graph. The presented techniques have been implemented and our empirical evaluation, which focusses on DL-safe rules, shows that our approach works well even compared to reasoners with dedicated rule support.

[1] `http://www.hermit-reasoner.com`

References

[Anonymous, 2013] Anonymous. Nominal schema absorption. Technical report, 2013. http://ijcai2013.tripod.com/webonmediacontents/NSA.pdf.

[Baader *et al.*, 2007] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[Blackburn and Tzakova, 1998] Patrick Blackburn and Miroslava Tzakova. Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence*, 24(1–4):23–49, 1998.

[Forgy, 1982] Charles L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[Horrocks and Sattler, 1999] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of of Logic and Computation*, 9(3):385–410, 1999.

[Horrocks and Tobies, 2000] Ian Horrocks and Stephan Tobies. Reasoning with axioms: Theory and practice. In *Proc. 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'00)*, pages 285–296. Morgan Kaufmann, 2000.

[Horrocks *et al.*, 2004] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin N. Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language*. W3C Member Submission, 21 May 2004. Available at http://www.w3.org/Submission/SWRL/.

[Horrocks *et al.*, 2006] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006.

[Hudek and Weddell, 2006] Alexander K. Hudek and Grant E. Weddell. Binary absorption in tableaux-based reasoning for description logics. In *Proc. 19th Int. Workshop on Description Logics (DL'06)*, volume 189. CEUR, 2006.

[Kifer and Boley, 2010] Michael Kifer and Harold Boley, editors. *RIF Overview*. W3C Working Group Note, 22 June 2010. Available at http://www.w3.org/TR/rif-overview/.

[Krisnadhi and Hitzler, 2012] Adila Krisnadhi and Pascal Hitzler. A tableau algorithm for description logics with nominal schema. In Markus Krötzsch and Umberto Straccia, editors, *Proc. 6th Int. Conf. on Web Reasoning and Rule Systems (RR'12)*, volume 7497 of *LNCS*, pages 234–237. Springer, 2012.

[Krötzsch *et al.*, 2011] Markus Krötzsch, Frederick Maier, Adila Krisnadhi, and Pascal Hitzler. A better uncle for OWL: nominal schemas for integrating rules and ontologies. In *Proc. 20th Int. Conf. on World Wide Web (WWW'11)*, pages 645–654. ACM, 2011.

[Ma *et al.*, 2006] Li Ma, Yang Yang, Zhaoming Qiu, Guotong Xie, Yue Pan, and Shengping Liu. Towards a complete OWL ontology benchmark. In *Proc. 3rd European Semantic Web Conf. (ESWC'06)*, volume 4011 of *LNCS*, pages 125–139. Springer, 2006.

[OWL Working Group, 2009] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.

[Sirin *et al.*, 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *J. of Web Semantics*, 5(2):51–53, 2007.

[Tsarkov and Horrocks, 2004] Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Proc. 17th Int. Workshop on Description Logics (DL'04)*, volume 104. CEUR, 2004.