Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Reasoning with Nominal Schemas through Absorption

Andreas Steigmiller · Birte Glimm · Thorsten Liebig

the date of receipt and acceptance should be inserted later

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 the novel reasoning system Konclude and present further optimisations. In our empirical evaluation, we show the effect of these optimisations and we find that the proposed nominal schema handling performs well even when compared to (hyper)tableau systems with dedicated rule support.

Keywords Description Logics · Nominal Schemas · Tableau Theorem Proving · Absorption · Implementation and Optimisation Techniques

1 Introduction

Description Logics (DLs) already have a long tradition as declarative knowledge representation formalisms. In recent years, they gained in popularity since they form the logical underpinning of the Web Ontology Language (OWL) [32], which is standardised by the World Wide Web Consortium (W3C). In many application contexts, it is, however, convenient to combine a knowledge base (called ontology in the context of OWL) with a set of rules. Consequently, combining OWL and rules is an active area of research and, recently, so-called

Andreas Steigmiller
Institute of Artificial Intelligence, University of Ulm, Ulm, Germany
E-mail: andreas.steigmiller@uni-ulm.de

Birte Glimm
Institute of Artificial Intelligence, University of Ulm, Ulm, Germany
E-mail: birte.glimm@uni-ulm.de

Thorsten Liebig
derivo GmbH, Ulm, Germany
E-mail: liebig@derivo.de

nominal schemas have been proposed [17] for expressing arbitrary DL-safe rules (as specified in the W3C standards SWRL [8] or RIF [15]) natively in DLs and, consequently, in OWL ontologies.

We address the problem of an efficient handling of nominal schema axioms in tableau calculi for Description Logics. Although some attempts (see, e.g., [16]) 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 [12]. 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 binds to individuals of the knowledge base. To ensure decidability and similarly to the safeness restriction for rules, one usually assumes that x binds only to individuals that are named in the knowledge base and not to anonymous individuals whose existence can only be inferred, e.g., due to existential restrictions in the knowledge base.

We use the same running example (or parts thereof) as Krisnadhi and Hitzler [16], which describes a conflicting review assignment between a person and a paper if the individual has to review a paper x that has an author (y) with whom that individual has a joint publication in the same venue (z):

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

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 named individuals). However, such nominal schema axioms can easily be eliminated by replacing them with all corresponding grounded axioms, i.e., the axioms that are obtained by replacing each nominal schema by a nominal, in all possible combinations, where all nominal schemas with the same variable are replaced by the same nominal. Thus, a knowledge base for a DL with nominal schema constructs, which is denoted by an additional $\mathcal{V}$ in the DL nomenclature, can be reduced, with this *upfront grounding* approach, to a knowledge base without nominal schema axioms. The upfront grounding is, however, very inefficient. For example, a nominal schema axiom with 3 variables can be grounded for a knowledge base with 100 named individuals in 100^3 different ways, which is prohibitive even for small examples. One way to restrict the effort of reasoning with nominal schemas is to restrict the expressiveness of the nominal schema axioms, whereby it is possible to achieve that the grounding adds only linearly or polynomially many new axioms [17]. For less expressive Description Logics, it is in principle also possible to convert the knowledge base into Datalog [21], or to use resolution-based decision procedures for reasoning [33]. However, it is not clear how efficient these approaches are in comparison to established DL reasoners and how to extend these approaches to more expressive Description Logics.

For efficient reasoning in OWL ontologies extended with nominal schemas, i.e., knowledge bases in the DL $\mathcal{SROIQV}$, it is more promising to adapt the established tableau algorithms, which are dominantly used for sound and complete reasoning systems. One such approach extends a tableau algorithm such that grounding is *delayed* until it is *required* [16]. The standard rules are blocked until the new grounding rules ensure that a concept with nominal schemas can be processed safely, e.g., the concept $\exists r.(\{x\} \sqcap C)$ has to be grounded before the $\exists$-rule can be applied. However, this requires significant changes to the tableau algorithm and, thus, existing optimisations, which are crucial for a reasonable performance on real-world ontologies, have to be adapted as well. Furthermore, it is not clear in which way concepts have to be grounded to achieve a well-performing implementation and some concepts even cannot be grounded efficiently, e.g., disjunctions that have the same nominal schema variable in several disjuncts have to be grounded before the disjunction can be processed.

In this paper, we present a new approach that works more from the opposite direction by collecting possible bindings for the nominal schema variables during the application of rules and, then, these bindings are used to complete the processing of the nominal schema axioms. To implement this idea, we extend the absorption, which is a widely used preprocessing step (Section 2.3 and 3), to handle nominal schemas (Section 4.1), and we adapt or add new rules to the tableau calculus, which create and propagate bindings of variables through the completion graph constructed by the tableau algorithm (Section 4.2). These bindings are then used to ground the remaining, non-absorbable part of the nominal schema axioms.

Our rules can be completely separated from other standard rules and, thus, can be integrated well into existing implementations without any adaptation of other optimisations. We have implemented our nominal schema absorption technique into the novel $\mathcal{SROIQ}$ reasoning system Konclude and the empirical evaluation shows that our approach works well, even if we convert ordinary DL-safe rules to nominal schema axioms and compare our approach to other DL reasoners with dedicated rule support. Our evaluation focuses on consistency checking, which is the elementary reasoning task in tableau-based reasoning system. In particular, higher level reasoning tasks, such as classification and realisation, are usually reduced to a multitude of consistency tests and, thus, the improvements shown for consistency checking by the presented approach also pay off for other reasoning tasks.

Note that this paper is based on a previous workshop [25] and conference publication [26]. Compared to these previous publications, this paper contains significantly extended examples and explanations, full proofs, descriptions of further optimisations, and an extended evaluation. In particular, the absorption technique presented in this paper is extended to more concept constructors and allows for generating so-called candidate concepts, which can be used for further absorption and to prune possible subsumers (Section 3). In addition, this paper contains proofs for the correctness of the absorption technique (Section 3.2) and the propagation of the bindings for nominal schema variables with the tableau algorithm (Section 4.3). Whereas the previous publications contain only sketches of ideas of further optimisation techniques (e.g., backward chaining, propagation of representatives), this paper presents these optimisations in detail (Section 5 and 7). Moreover, this paper describes new optimisation techniques and relevant implementation improvements (Section 6 and 7). Last but not least, the evaluation presented in this paper is significantly extended and covers more (benchmark-)ontologies and a greater diversity of DL-safe rules (Section 7).

The paper is organised as follows: Section 2 summarises some preliminaries about Description Logics and model construction calculi. Section 3 introduces the absorption mechanism that forms the basis for integrating nominal schemas. In Section 4 we present our new nominal schema absorption technique, which is then further optimised in Section 5 and Section 6. The empirical evaluation is shown in Section 7 and we conclude in Section 8.

2 Preliminaries

In this section, we first give a brief introduction into Description Logics extended with nominal schemas. For ease of presentation, we introduce only the DL $\mathcal{ALCOIQ}$ with its extension to nominal schemas here instead of $\mathcal{SROIQ}$ [7], which underpins OWL 2, but our technique to handle nominal schemas can easily be used with $\mathcal{SROIQV}$ too, since the additional features can either be encoded in $\mathcal{ALCOIQ}$ (possibly via an exponential encoding) [7, 14, 23] or they can be handled by simple additional tableau rules.

2.1 The Description Logic $\mathcal{ALCOIQV}$

We first define the syntax and semantics of roles, and then go on to $\mathcal{ALCOIQV}$-concepts, individuals, and ontologies/knowledge bases.

Definition 1 (Syntax of $\mathcal{ALCOIQV}$) Let N_C, N_R, N_I, and N_V be countable, infinite, and pairwise disjoint sets of *concept names*, *role names*, *individual names*, and *variable names*, respectively. We call $\mathcal{S} = (N_C, N_R, N_I, N_V)$ a *signature*. The set $\mathsf{rol}(\mathcal{S})$ of *$\mathcal{ALCOIQV}$-roles* over $\mathcal{S}$ (or roles for short) is $N_R \cup \{r^- \mid r \in N_R\}$, where roles of the form r^- are called *inverse roles*. Since the inverse relation on roles is symmetric, we can define a function inv, which returns the inverse of a role and, therefore, we do not have to consider roles of the from r^{--}. For $r \in N_R$, let be $\mathsf{inv}(r) = r^-$ and $\mathsf{inv}(r^-) = r$.

The set of *$\mathcal{ALCOIQV}$-concepts* (or concepts for short) over $\mathcal{S}$ is the smallest set built inductively over symbols from $\mathcal{S}$ using the following grammar, where $a \in N_I, x \in N_V, n \in \mathbb{N}_0, A \in N_C$, and $r \in \mathsf{rol}(\mathcal{S})$:

$$C ::= \top \mid \bot \mid \{a\} \mid \{x\} \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall r.C \mid \exists r.C \mid \leqslant n\ r.C \mid \geqslant n\ r.C.$$

Definition 2 (Semantics of $\mathcal{ALCOIQV}$-concepts) An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$, the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$, which maps every concept name $A \in N_C$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, every role name $r \in N_R$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and every individual name $a \in N_I$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$. For each role name $r \in N_R$, the interpretation of its inverse role $(r^-)^\mathcal{I}$ consists of all pairs $\langle \delta, \delta' \rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for which $\langle \delta', \delta \rangle \in r^\mathcal{I}$. A *variable assignment* for $\mathcal{I}$ is a function $\mu \colon N_V \to \Delta^\mathcal{I}$ such that, for each $x \in N_V$, $\mu(x) = a^\mathcal{I}$ for some $a \in N_I$.

For any interpretation $\mathcal{I}$ and assignment μ, the semantics of $\mathcal{ALCOIQV}$-concepts over a signature $\mathcal{S}$ is defined by the function $\cdot^{\mathcal{I},\mu}$ as follows:

$$\begin{aligned}
\top^{\mathcal{I},\mu} &= \Delta^\mathcal{I} & A^{\mathcal{I},\mu} &= A^\mathcal{I} & (\{a\})^{\mathcal{I},\mu} &= \{a^\mathcal{I}\} \\
\bot^{\mathcal{I},\mu} &= \emptyset & r^{\mathcal{I},\mu} &= r^\mathcal{I} & (\{x\})^{\mathcal{I},\mu} &= \{\mu(x)\} \\
(\neg C)^{\mathcal{I},\mu} &= \Delta^\mathcal{I} \setminus C^{\mathcal{I},\mu} & (C \sqcap D)^{\mathcal{I},\mu} &= C^{\mathcal{I},\mu} \cap D^{\mathcal{I},\mu} & (C \sqcup D)^{\mathcal{I},\mu} &= C^{\mathcal{I},\mu} \cup D^{\mathcal{I},\mu} \\
(\forall r.C)^{\mathcal{I},\mu} &= \{\delta \in \Delta^\mathcal{I} \mid \text{if } \langle \delta, \delta' \rangle \in r^\mathcal{I}, \text{ then } \delta' \in C^{\mathcal{I},\mu}\} \\
(\exists r.C)^{\mathcal{I},\mu} &= \{\delta \in \Delta^\mathcal{I} \mid \text{there is a } \langle \delta, \delta' \rangle \in r^\mathcal{I} \text{ with } \delta' \in C^{\mathcal{I},\mu}\} \\
(\leqslant n\ r.C)^{\mathcal{I},\mu} &= \{\delta \in \Delta^\mathcal{I} \mid \sharp\{\delta' \in \Delta^\mathcal{I} \mid \langle \delta, \delta' \rangle \in r^\mathcal{I} \text{ and } \delta' \in C^{\mathcal{I},\mu}\} \leq n\} \\
(\geqslant n\ r.C)^{\mathcal{I},\mu} &= \{\delta \in \Delta^\mathcal{I} \mid \sharp\{\delta' \in \Delta^\mathcal{I} \mid \langle \delta, \delta' \rangle \in r^\mathcal{I} \text{ and } \delta' \in C^{\mathcal{I},\mu}\} \geq n\},
\end{aligned}$$

where $\sharp M$ denotes the cardinality of the set M.

Definition 3 (Syntax and Semantics of Axioms and Ontologies) For C, D concepts, a *general concept inclusion* (GCI) is an expression $C \sqsubseteq D$. We introduce $C \equiv D$ as an abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. A finite set of GCIs is called a *TBox*. An (ABox) *assertion* is an expression of the form $C(a)$ or $r(a, b)$,[1] where C is a concept, r is a role, and $a, b \in N_I$ are individual names. An *ABox* is a finite set of assertions. We call $\mathcal{T} \cup \mathcal{A}$ a *knowledge base* with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation and μ an assignment, then $\mathcal{I}$ and μ *satisfy* an axiom or assertion α, written $\mathcal{I}, \mu \models \alpha$ if (i) α is a GCI $C \sqsubseteq D$ and $C^{\mathcal{I},\mu} \subseteq D^{\mathcal{I},\mu}$, (ii) α is an assertion $C(a)$ and $a^{\mathcal{I}} \in C^{\mathcal{I},\mu}$ or (iii) α is an assertion $r(a, b)$ and $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$. The interpretation $\mathcal{I}$ *satisfies* α if $\mathcal{I}, \mu \models \alpha$ for every assignment μ. $\mathcal{I}$ *satisfies* a TBox $\mathcal{T}$ (ABox $\mathcal{A}$) if it satisfies each GCI in $\mathcal{T}$ (each assertion in $\mathcal{A}$). We say that $\mathcal{I}$ *satisfies* $\mathcal{K}$ if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathcal{A}$. In this case, we say that $\mathcal{I}$ is a *model* of $\mathcal{K}$ and write $\mathcal{I} \models \mathcal{K}$. We say that $\mathcal{K}$ *is consistent* if $\mathcal{K}$ has a model.

In the remainder of the paper we assume w.l.o.g. that all axioms in a knowledge base use different variable names. Furthermore, one usually imposes a "safeness" condition on variable assignments. A variable assignment μ for an interpretation $\mathcal{I}$ of a knowledge base $\mathcal{K}$ is *safe* if, for each nominal schema variable x occurring in $\mathcal{K}$ with $\mu(x) = d$, there is an individual $a \in N_I$ that occurs in $\mathcal{K}$ such that $a^{\mathcal{I}} = d$. We also make this assumption in the remainder of this paper. Hence, from now on, when we say that an interpretation $\mathcal{I}$ for a knowledge base $\mathcal{K}$ *satisfies* an axiom or an assertion α, then $\mathcal{I}, \mu \models \alpha$ for every *safe* assignment μ.

In the following proposition, we write $\mathcal{I} \models_S \alpha$ if $\mathcal{I}, \mu \models \alpha$ holds for every variable assignment μ whose range is restricted to the (finite) set S and we use $\mathsf{Inds}(\mathcal{K})$ to denote the set of individual names occurring in a knowledge base $\mathcal{K}$.Without the unique name assumption, the following proposition is a straightforward consequence of the defined semantics and the safeness condition (cf. [18]):

Proposition 1 *Let $\mathcal{K}$ be a knowledge base and $\mathcal{I}$ an interpretation, then $\mathcal{I} \models \mathcal{K}$ iff $\mathcal{I} \models_S \mathcal{K}$, where $S = \{a^{\mathcal{I}} \mid a \in \mathsf{Inds}(\mathcal{K})\}$.*

We assume that the safeness condition is similarly imposed for satisfiability of ABoxes, TBoxes, and for the consistency of knowledge bases.

Note, the safeness condition allows us to focus on the signature symbols that actually occur in a knowledge base for solving the reasoning tasks. In particular, with the safeness condition, the upfront grounding can be restricted to the named individuals that occur in a knowledge base and, thus, the upfront grounding of an $\mathcal{ALCOIQV}$ knowledge base results in a finite set of axioms.

Also note that if the knowledge base does not contain any nominal schemas, then we do not have to consider the variable assignments for the satisfiability of axioms, TBoxes, ABoxes and knowledge bases. We exploit this in the following to improve readability, i.e., we omit the variable assignments where possible.

2.2 Tableau Calculus

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".

[1] Many DLs allow for expressing other types of ABox assertions, e.g., $a \neq b$. We omit this here for ease of presentation since such assertions can easily be expressed with GCIs in the presence of nominals, e.g., $a \neq b$ is equivalent to the GCI $\{a\} \sqsubseteq \neg\{b\}$.

The concepts that are possibly used in the completion graph by a tableau calculus can be described with the concept closure.

Definition 4 (Closure) The *closure* $\mathsf{clos}(C)$ of a concept C is a set of concepts that is closed under sub-concepts of C and also contains C. The *full closure* $\mathsf{fclos}(C)$ of a concept C is defined as:

$$\mathsf{fclos}(C) := \mathsf{clos}(C) \cup \{\leqslant m\, r.C \mid \leqslant n\, r.C \in \mathsf{clos}(C) \text{ and } m \leq n\}.$$

Given a set Z of axioms and assertions, we extend the full closure as follows:

$$\mathsf{fclos}(Z) := \{\mathsf{fclos}(\mathsf{nnf}(\neg C \sqcup D)) \mid C \sqsubseteq D \in Z\} \cup \{\mathsf{fclos}(C) \mid C(a) \in Z\}.$$

Now, the completion graph for an $\mathcal{ALCOIQ}$ knowledge base $\mathcal{K}$ is defined as follows:

Definition 5 (Completion Graph) A *completion graph* for $\mathcal{K}$ is a directed graph $G = (V, E, \mathcal{L}, \dot{\neq})$. Each node $v \in V$ is labelled with a set $\mathcal{L}(v) \subseteq \mathsf{fclos}(\mathcal{K})$. Each edge $\langle v, v'\rangle \in E$ is labelled with the set $\mathcal{L}(\langle v, v'\rangle) \subseteq \mathsf{rol}(\mathcal{K})$, where $\mathsf{rol}(\mathcal{K})$ are the roles occurring in $\mathcal{K}$. The symmetric binary relation $\dot{\neq}$ is used to keep track of inequalities between nodes in V. We say $G = (V, E, \mathcal{L}, \dot{\neq})$ contains the *concept fact* $C(v)$ (*role fact* $r(v, v')$) if $C \in \mathcal{L}(v)$ ($\langle v_1, v_2\rangle \in E$ and $r \in \mathcal{L}(\langle v_1, v_2\rangle)$).

In the following, we often use $r \in \mathcal{L}(\langle v_1, v_2\rangle)$ as an abbreviation for $\langle v_1, v_2\rangle \in E$ and $r \in \mathcal{L}(\langle v_1, v_2\rangle)$.

Definition 6 (Successor, Predecessor, Neighbour) If $\langle v_1, v_2\rangle \in E$, then v_2 is called a *successor* of v_1 and v_1 is called a *predecessor* of v_2. *Ancestor* is the transitive closure of predecessor, and *descendant* is the transitive closure of successor. A node v_2 is called an *r-successor* of a node v_1 if $r \in \mathcal{L}(\langle v_1, v_2\rangle)$. A node v_2 is called a neighbour (r-neighbour) of a node v_1 if v_2 is a successor (r-successor) of v_1 or if v_1 is a successor ($\mathsf{inv}(r)$-successor) of v_2.

For a role r and a node $v \in V$, we define the set of v's r-neighbours with the concept C in their label, written $\mathsf{neighbs}(v, r, C)$, as $\{v' \in V \mid v' \text{ is an } r\text{-neighbour of } v \text{ and } C \in \mathcal{L}(v')\}$.

The completion graph is initialised for the tableau algorithm by creating one node for each ABox individual/nominal in the input knowledge base (w.l.o.g. we assume that the ABox is non-empty, should this not be the case, we can always add an assertion $\top(a)$ for a fresh individual a) and by adding the concept and role facts for the ABox assertions of $\mathcal{K}$. If $v_1, \ldots, v_\ell$ are the nodes for the ABox individuals $a_1, \ldots, a_\ell$ of $\mathcal{K}$, then the initial completion graph $G = (\{v_1, \ldots, v_\ell\}, E, \mathcal{L}, \emptyset)$ has to contain (i) for each ABox assertion of the form $C(a_i)$ the concept fact $C(v_i)$, i.e., $C \in \mathcal{L}(v_i)$, (ii) for each ABox assertion of the form $r(a_i, a_j)$ the role fact $r(v_i, v_j)$, i.e., $\langle v_i, v_j\rangle \in E$ and $r \in \mathcal{L}(\langle v_i, v_j\rangle)$. Furthermore, we add for each ABox individual a_i the nominal $\{a_i\}$ and the concept $\top$ to the label of v_i, i.e., $\{\{a_i\}, \top\} \subseteq \mathcal{L}(v_i)$.

Additionally, we assume all concepts to be in negation normal form (NNF). Each concept can be transformed into an equivalent one in NNF by pushing negation inwards, making use of de Morgan's laws and the following equivalences that exploit the duality between existential and universal restrictions, and between at-most and at-least cardinality restrictions [11]:

$$\begin{array}{lll}
\neg(\exists r.C) \equiv \forall r.\neg C & \neg(\geqslant k\, r.C) \equiv \leqslant (k-1)\, r.C & \neg(\geqslant 0\, r.C) \equiv \bot \\
\neg(\forall r.C) \equiv \exists r.\neg C & \neg(\leqslant n\, r.C) \equiv \geqslant (n+1)\, r.C &
\end{array}$$

Table 1 Tableau expansion rules for $\mathcal{ALCOIQ}$ knowledge bases

$\top$-rule	if		$\top \sqsubseteq C \in \mathcal{K}$, $C \notin \mathcal{L}(v)$, and v is not indirectly blocked
	then		$\mathcal{L}(v) = \mathcal{L}(v) \cup \{C\}$
$\sqcap$-rule	if		$C_1 \sqcap C_2 \in \mathcal{L}(v)$, v is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$
	then		$\mathcal{L}(v) = \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule	if		$C_1 \sqcup C_2 \in \mathcal{L}(v)$, v is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then		$\mathcal{L}(v') = \mathcal{L}(v') \cup \{H\}$ for some $H \in \{C_1, C_2\}$
$\exists$-rule	if		$\exists r.C \in \mathcal{L}(v)$, v is not blocked, and v has no r-neighbour v' with $C \in \mathcal{L}(v')$
	then		create a new node v' and an edge $\langle v, v' \rangle$ with $\mathcal{L}(v') = \{\top, C\}$ and $\mathcal{L}(\langle v, v' \rangle) = \{r\}$
$\forall$-rule	if		$\forall r.C \in \mathcal{L}(v)$, v is not indirectly blocked, and there is an r-neighbour v' of v with $C \notin \mathcal{L}(v')$
	then		$\mathcal{L}(v') = \mathcal{L}(v') \cup \{C\}$
ch-rule	if		$\leqslant n\, r.C \in \mathcal{L}(v)$, v is not indirectly blocked, and there is an r-neighbour v' of v with $\{C, \mathsf{nnf}(\neg C)\} \cap \mathcal{L}(v') = \emptyset$
	then		$\mathcal{L}(v') = \mathcal{L}(v') \cup \{H\}$ for some $H \in \{C, \mathsf{nnf}(\neg C)\}$
$\geqslant$-rule	if	1.	$\geqslant n\, r.C \in \mathcal{L}(v)$, v is not blocked, and
		2.	there are not n r-neighbours $v_1, \ldots, v_n$ of v with $C \in \mathcal{L}(v_i)$ and $v_i \dot{\neq} v_j$ for $1 \leq i < j \leq n$
	then		create n new nodes $v_1, \ldots, v_n$ with $\mathcal{L}(\langle v, v_i \rangle) = \{r\}$, $\mathcal{L}(v_i) = \{\top, C\}$ and $v_i \dot{\neq} v_j$ for $1 \leq i < j \leq n$.
$\leqslant$-rule	if	1.	$\leqslant n\, r.C \in \mathcal{L}(v)$, v is not indirectly blocked,
		2.	$\sharp\mathsf{neighbs}(v, r, C) > n$ and there are two r-neighbours v_1, v_2 of v with $C \in (\mathcal{L}(v_1) \cap \mathcal{L}(v_2))$ and not $v_1 \dot{\neq} v_2$
	then	a.	if v_1 is a nominal node, then $\mathsf{merge}(v_2, v_1)$
		b.	else if v_2 is a nominal node or an ancestor of v_1, then $\mathsf{merge}(v_1, v_2)$
		c.	else $\mathsf{merge}(v_2, v_1)$
o-rule	if		there are two nodes v, v' with $\{a\} \in (\mathcal{L}(v) \cap \mathcal{L}(v'))$ and not $v \dot{\neq} v'$
	then		$\mathsf{merge}(v, v')$
NN-rule	if	1.	$\leqslant n\, r.C \in \mathcal{L}(v)$, v is a nominal node, and there is a blockable r-neighbour v' of v such that $C \in \mathcal{L}(v')$ and v is a successor of v',
		2.	there is no m such that $1 \leq m \leq n$, $(\leqslant m\, r.C) \in \mathcal{L}(v)$, and there exist m nominal r-neighbours $v_1, \ldots, v_m$ of v with $C \in \mathcal{L}(v_i)$ and $v_i \dot{\neq} v_j$ for all $1 \leq i < j \leq m$
	then	1.	guess m with $1 \leq m \leq n$ and $\mathcal{L}(v) = \mathcal{L}(v) \cup \{\leqslant m\, r.C\}$
		2.	create m new nodes $v'_1, \ldots, v'_m$ with $\mathcal{L}(\langle v, v'_i \rangle) = \{r\}$, $\mathcal{L}(v'_i) = \{\top, C, \{a_i\}\}$ with each $a_i \in N_I$ new in G and $\mathcal{K}$, and $v'_i \dot{\neq} v'_j$ for $1 \leq i < j \leq m$.

where $k \in \mathbf{N}$ and $n \in \mathbf{N}_0$. For C a concept possibly not in NNF, let $\mathsf{nnf}(C)$ be the equivalent concept to C in NNF.

The tableau algorithm works by decomposing concepts in the completion graph with a set of expansion rules (see Table 1). 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 for the input knowledge base. The rules are repeatedly 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 and clash-free completion graph.

During the expansion it is sometimes necessary to merge two nodes or to delete (prune) a part of the completion graph. When a node w is merged into a node v (e.g., by an application of the $\leqslant$-rule), we "prune" the completion graph by removing w and, recursively, all blockable successors of w to prevent a further rule application on these nodes.

Intuitively, when we merge a node w into a node v, we add $\mathcal{L}(w)$ to $\mathcal{L}(v)$, "move" all the edges leading to w so that they lead to v and "move" all the edges leading from w to nominal nodes so that they lead from v to the same nominal nodes; we then remove w (and blockable sub-trees below w) from the completion graph.

Definition 7 (Pruning, Merging) Pruning a node w in the completion graph $G = (V, E, \mathcal{L}, \dot{\neq})$, written $\mathsf{prune}(w)$, yields a graph that is obtained from G as follows:

1. for all successors v' of w, remove $\langle w, v' \rangle$ from E and, if v' is blockable, $\mathsf{prune}(v')$;
2. remove w from V.

Merging a node w into a node v in $G = (V, E, \mathcal{L}, \dot{\neq})$, written $\mathsf{merge}(w, v)$, yields a graph that is obtained from G as follows:

1. for all nodes v' such that $\langle v', w \rangle \in E$
   (a) if $\{\langle v, v' \rangle, \langle v', v \rangle\} \cap E = \emptyset$, then add $\langle v', v \rangle$ to E and set $\mathcal{L}(\langle v', v \rangle) = \mathcal{L}(\langle v', w \rangle)$,
   (b) if $\langle v', v \rangle \in E$, then set $\mathcal{L}(\langle v', v \rangle) = \mathcal{L}(\langle v', v \rangle) \cup \mathcal{L}(\langle v', w \rangle)$,
   (c) if $\langle v, v' \rangle \in E$, then set $\mathcal{L}(\langle v, v' \rangle) = \mathcal{L}(\langle v, v' \rangle) \cup \{\mathsf{inv}(r) \mid r \in \mathcal{L}(\langle v', w \rangle)\}$, and
   (d) remove $\langle v', w \rangle$ from E;
2. for all nominal nodes v' such that $\langle w, v' \rangle \in E$
   (a) if $\{\langle v, v' \rangle, \langle v', v \rangle\} \cap E = \emptyset$, then add $\langle v, v' \rangle$ to E and set $\mathcal{L}(\langle v, v' \rangle) = \mathcal{L}(\langle w, v' \rangle)$,
   (b) if $\langle v, v' \rangle \in E$, then set $\mathcal{L}(\langle v, v' \rangle) = \mathcal{L}(\langle v, v' \rangle) \cup \mathcal{L}(\langle w, v' \rangle)$,
   (c) if $\langle v', v \rangle \in E$, then set $\mathcal{L}(\langle v', v \rangle) = \mathcal{L}(\langle v', v \rangle) \cup \{\mathsf{inv}(r) \mid r \in \mathcal{L}(\langle w, v' \rangle)\}$, and
   (d) remove $\langle w, v' \rangle$ from E;
3. $\mathcal{L}(v) = \mathcal{L}(v) \cup \mathcal{L}(w)$;
4. add $v \dot{\neq} v'$ for all v' such that $w \dot{\neq} v'$; and
5. $\mathsf{prune}(w)$.

Unrestricted application of the $\exists$-rule and $\geqslant$-rule can lead to the introduction of infinitely many new tableau nodes and, thus, prevent the calculus from terminating. To counteract that, a cycle detection technique called *(pairwise) blocking* [9] is used that restricts the application of these rules. To apply blocking, we distinguish *blockable nodes* from *nominal nodes*, which have either an original nominal from the knowledge base or a new nominal introduced by the calculus in their label.

Definition 8 (Pairwise Blocking) A node is *blocked* if either it is directly or indirectly blocked. A node v is *indirectly blocked* if an ancestor of v is blocked; and v with predecessor v' is *directly blocked* if there exists an ancestor node w of v with predecessor w' such that

1. v, v', w, w' are all blockable,
2. w, w' are not blocked,
3. $\mathcal{L}(v) = \mathcal{L}(w)$ and $\mathcal{L}(v') = \mathcal{L}(w')$,
4. $\mathcal{L}(\langle v', v \rangle) = \mathcal{L}(\langle w', w \rangle)$.

In this case, we say that w directly blocks v and w is the blocker of v.

In order to guarantee that each node of the completion graph indeed satisfies all axioms of the TBox, one can "internalise" the TBox into a concept that is added to each node label. For example, if the TBox contains the axioms

$$A_1 \sqsubseteq \exists r.(B_1 \sqcap B_2) \tag{1}$$

$$\exists r.(B_1 \sqcup B_2) \sqsubseteq \exists s.A_2 \tag{2}$$

Table 2 Lazy unfolding rules for tableau algorithms

$\sqsubseteq_1$-rule	if	$A \in \mathcal{L}(v)$, $A \sqsubseteq C \in \mathcal{K}$, $C \notin \mathcal{L}(v)$, and v is not indirectly blocked
	then	$\mathcal{L}(v) = \mathcal{L}(v) \cup \{C\}$
$\sqsubseteq_2$-rule	if	$\{A_1, A_2\} \subseteq \mathcal{L}(v)$, $(A_1 \sqcap A_2) \sqsubseteq C \in \mathcal{K}$, $C \notin \mathcal{L}(v)$, and v is not indirectly blocked
	then	$\mathcal{L}(v) = \mathcal{L}(v) \cup \{C\}$

the internalised concept C_I contains one conjunct for each axiom that is a disjunction with the negated left-hand side of the axiom and the right-hand side:

$$C_I = \mathsf{nnf}(((\neg A_1 \sqcup \exists r.(B_1 \sqcap B_2)) \sqcap (\neg(\exists r.(B_1 \sqcup B_2)) \sqcup \exists s.A_2))).$$

A tableau algorithm based on the expansion rules of Table 1 adds the internalised concept with an auxiliary axiom $\top \sqsubseteq C_I$ by the $\top$-rule. It is well-known that such a tableau algorithm decides the consistency of knowledge bases [7, 10]:

Theorem 1 *A tableau algorithm based on the rules of Table 1 builds a clash-free and fully expanded completion graph for an internalised $\mathcal{ALCIOQ}$ knowledge base $\mathcal{K}$ iff $\mathcal{K}$ is consistent.*

However, this internalisation technique introduces a large number of disjunctions in each node label, which possibly require several non-deterministic choices and backtracking if the choice resulted in a clash.

It would be more efficient to integrate a rule into the tableau calculus that checks for each GCI $C \sqsubseteq D$ whether C is satisfied for a node and if this is the case, then D is also added to the node label. For Axiom (1), for example, it is easy to check whether the atomic concept A_1 is satisfied at a node and if this is the case, then $\exists r.(B_1 \sqcap B_2)$ has to be added to the label of the node. This *lazy unfolding* for atomic concepts can be realised by additionally using the $\sqsubseteq_1$-rule of Table 2 in the tableau algorithm, whereby we do not have to internalize axioms of the form $A \sqsubseteq C$.

Checking whether a complex left-hand side of an axiom is satisfied can, however, be non-trivial. For example, it can often not be verified syntactically, whether a node satisfies $\exists r.(B_1 \sqcup B_2)$, which would be required for Axiom (2). For example, any instance of A_1 has an r-successor that satisfies $B_1 \sqcap B_2$ and, therefore, this A_1 instance (semantically) also satisfies $\exists r.(B_1 \sqcup B_2)$. In order to avoid the internalisation of such GCIs with axioms of the form $\top \sqsubseteq \mathsf{nnf}(\neg C \sqcup D)$ (and the resulting processing of disjunctions), practical reasoning systems use elaborate transformations in a preprocessing step called absorption, which we describe in more detail in Section 2.3 and 3.

2.3 Absorption

An absorption algorithm extracts those conditions of an axiom for which it can be ensured that if one of these conditions is not satisfied for a node in a completion graph, then the axiom is trivially satisfiable. The extracted conditions are then used for expressing the axiom in such a way that internalisation and the processing of disjunctions in the tableau algorithm can be avoided as much as possible, which significantly reduces non-determinism.

For example, one would like to avoid treating Axiom (2) as $\top \sqsubseteq \forall r.(\neg B_1 \sqcap \neg B_2) \sqcup \exists s.A_2$ as motivated in the previous section. Any node that does *not* have an r-neighbour trivially satisfies $\forall r.(\neg B_1 \sqcap \neg B_2)$ and, hence, the overall disjunction that corresponds to the internalised axiom. Thus, the existence of an r-neighbour node can be used as condition to rewrite the axioms such that the disjunction is only added to nodes that satisfy this condition, e.g., $\top \sqsubseteq \forall r^{-}.(\forall r.(\neg B_1 \sqcap \neg B_2) \sqcup \exists s.A_2)$. We can, however, go even further by first identifying nodes that satisfy B_1 or B_2. If we find such a node, we can make sure that its r^{-}-neighbour has to satisfy $\exists s.A_2$. This is captured by the following axioms:

$$B_1 \sqsubseteq T_1 \qquad B_2 \sqsubseteq T_1 \qquad T_1 \sqsubseteq \forall r^{-}.T_2 \qquad T_2 \sqsubseteq \exists s.A_2,$$

where T_1 and T_2 are fresh atomic concepts that are used to enforce the semantics of the original axiom. We call the disjunct $\forall r.(\neg B_1 \sqcap \neg B_2)$ completely absorbable since it can be moved to the left-hand sides of simple inclusion axioms which trigger the remaining, non-absorbable part of the original axiom (i.e., $\exists s.A_2$ is automatically added for corresponding nodes in the completion graph by the $\sqsubseteq_1$-rule). Accordingly, we also use the term of absorption in the context of concepts, i.e., the absorption of a concept C generates all those axioms that are required to "automatically" detect whether C is satisfied for a node in the completion graph. For instance, the atomic concept T_2 from the absorption above implies that $\forall r.(\neg B_1 \sqcap \neg B_2)$ is satisfied and T_2 is automatically added by lazy unfolding. Hence, the axioms $B_1 \sqsubseteq T_1$, $B_2 \sqsubseteq T_1$, and $T_1 \sqsubseteq \forall r^{-}.T_2$ are the absorption of the concept $\forall r.(\neg B_1 \sqcap \neg B_2)$. Generally, the goal of the absorption preprocessing step is the extraction of such easy to verify conditions that allow for expressing a GCI by possibly several simpler inclusion axioms that ideally do not require a disjunction, e.g., as in the case of Axiom (2) above.

For the absorption of more complex concepts it is often necessary to join several conditions, say A_1 to A_n. A possibility to do this in an efficient way is *binary absorption* [13], where two concepts A_1 and A_2 imply a new 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 or to initiate the addition of the remaining and non-absorbed part of the disjunction. By joining the conditions binarily, it is possible to reuse more of these joins for several axioms if the axioms have some common conditions. Note, a binary absorption axiom $(A_1 \sqcap A_2) \sqsubseteq C$ is usually handled by a separate $\sqsubseteq_2$-rule (cf. Table 2), which adds the concept C to a label only if A_1 and A_2 are already present.

3 A Recursive Algorithm for Partial Absorption

Since our handling of nominal schemas is based on absorption methods and the quality of the absorption technique is crucial for this approach to work well, we next present an improved variant of a recursive binary absorption algorithm, which we then extend to nominal schemas in the next section.

The recursion facilitates further optimisations such as backward chaining, which are required for the handling of nominal schemas (see Section 5). In order to avoid infinite recursion, the presented absorption only handles acyclic concepts, where acyclicity is defined as follows: A_1 *directly uses* A_2 w.r.t. a TBox $\mathcal{T}$ if $A_1 \equiv C \in \mathcal{T}$ or $A_1 \sqsubseteq C \in \mathcal{T}$ and A_2 occurs in C; *uses* is the transitive closure of "directly uses". Then, a concept D *is acyclic w.r.t. a TBox* $\mathcal{T}$ if it contains no concept A that uses itself. We use the acyclicity restriction to keep the absorption algorithm simple, however, this restriction is not relevant in practice, because a cyclic concept A with the definition $A \sqsubseteq C$ can simply be made acyclic by representing $A \sqsubseteq C$ as $\top \sqsubseteq \neg A \sqcup C$.

Due to the recursion, the presented algorithm works quite differently in comparison to the original binary absorption. We exploit this for the integration of further improvements such as partial absorption and candidate concepts. In particular, the presented algorithm

Algorithm 1 isCA(C) and isPA(C)

```
Output: Returns whether the concept $C$ is completely absorbable
 1: procedure isCA($C$)
 2:     if $C = C_1 \sqcup C_2$ then
 3:         return isCA($C_1$) $\wedge$ isCA($C_2$)          ▷
 4:     else if $C = C_1 \sqcap C_2$ then
 5:         return isCA($C_1$) $\wedge$ isCA($C_2$)
 6:     else if $C = \forall r.C'$ then
 7:         return isCA($C'$)          ▷
 8:     else if $C = \leqslant n\, r.C'$ then
 9:         return false          ▷
10:     else if $C = \neg\{a\}$ then
11:         return true
        ...
12:     else if $C = \neg A$ then
13:         if $A$ is not acyclic then
14:             return false
15:         end if
16:         for all $A \equiv C' \in \mathcal{T}$ do
17:             if $\neg$isCA(nnf($\neg C'$)) then
18:                 return false
19:             end if
20:         end for
21:         return true
22:     end if
23:     return false
24: end procedure
```

```
Output: Returns whether the concept $C$ is partially absorbable
 1: procedure isPA($C$)
 2:     if $C = C_1 \sqcup C_2$ then
 3:         return isPA($C_1$) $\vee$ isPA($C_2$)          ▷
 4:     else if $C = C_1 \sqcap C_2$ then
 5:         return isPA($C_1$) $\wedge$ isPA($C_2$)
 6:     else if $C = \forall r.C'$ then
 7:         return true          ▷
 8:     else if $C = \leqslant n\, r.C'$ then
 9:         return true          ▷
10:     else if $C = \neg\{a\}$ then
11:         return true
        ...
12:     else if $C = \neg A$ then
13:         if $A$ is not acyclic then
14:             return false
15:         end if
16:         for all $A \equiv C' \in \mathcal{T}$ do
17:             if $\neg$isPA(nnf($\neg C'$)) then
18:                 return false
19:             end if
20:         end for
21:         return true
22:     end if
23:     return false
24: end procedure
```

completely handles an axiom X in one step, i.e., the absorbable (sub-)concepts of X are recursively traversed and all required (simple) inclusion axioms for the absorption are directly generated. In contrast, the original binary absorption algorithm introduces new GCIs for the absorption of an axiom, which are then (possibly) further absorbed in separate steps.

This complete handling of axioms allows for partially absorbing parts of axioms without creating additional disjunctions. Hence, it significantly improves the original binary absorption for more expressive Description Logics. Roughly speaking, partial absorption also uses parts of non-absorbable concepts as conditions to delay the processing of the non-absorbable part of the axiom. For example, the TBox axiom $\geqslant 5\, r.A \sqsubseteq D$ is, without absorption, handled as $\top \sqsubseteq \leqslant 4\, r.A \sqcup D$. 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. The partial absorption is able to extract these conditions, and 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 $A \sqsubseteq \forall r^-.T_1$ and $T_1 \sqsubseteq \leqslant 4\, r.A \sqcup D$.

Algorithms that absorb parts of axioms in separate steps, e.g., by introducing new GCIs that are handled separately, often create additional disjunctions, especially if the axioms have more complex (sub-)concepts. For instance, by simplifying the axiom $\geqslant 5\, r.(\geqslant 3\, s.A) \sqsubseteq D$ into $\geqslant 5\, r.T \sqsubseteq D$ and $T \equiv \geqslant 3\, s.A$, for T a fresh atomic concept that is used to split the original axiom, the GCI $T \equiv \geqslant 3\, s.A$ cannot further be absorbed without creating additional disjunctions. In contrast, the partial absorption creates the axioms $A \sqsubseteq \forall s^-.T_1$, $T_1 \sqsubseteq \forall r^-.T_2$, and $T_2 \sqsubseteq \leqslant 4\, r.(\geqslant 3\, s.A) \sqcup D$, whereby the processing of the disjunction is significantly delayed (or even completely avoided if there is no node in the completion graph for which T_2 is added).

Algorithm 2 collectDisjuncts(C, *absorbable*)

```
Output: 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:     $S \leftarrow (S \setminus (C_1 \sqcup C_2)) \cup \{C_1, C_2\}$
 4: end while
 5: if $absorbable$ then
 6:     return $\{ C \in S \mid \mathsf{isPA}(C) \}$
 7: else
 8:     return $\{ C \in S \mid \neg\mathsf{isCA}(C) \}$
 9: end if
```

Algorithm 3 absorbJoined(S)

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

In the following, we describe the absorption algorithm and we use $C_{(i)}, D_{(i)}$ for (possibly complex) concepts, $A_{(i)}, T_{(i)}$ for atomic concepts with $T_{(i)}$ for fresh concepts, and S for a set of concepts. For simplicity, we also consider $\top$ as an atomic concept. The algorithm uses the following functions to absorb axioms of a (global) TBox $\mathcal{T}$ into the new (global) TBox $\mathcal{T}'$ ($\mathcal{T}$ and $\mathcal{T}'$ are considered to be global for the ease of presentation[2]):

- isCA(C) (isPA(C)), shown in Algorithm 1, returns whether the concept C is completely (partially) absorbable. Note, if a concept C is completely absorbable, then it is also partially absorbable, and moreover, if a concept C is not partially absorbable, then it is also not completely absorbable. We have tagged the lines 3, 7 and 9 with a comment symbol to highlight where isPA might allow additional absorption in comparison to the isCA function.
- collectDisjuncts(C, *absorbable*), shown in Algorithm 2, returns a set of (partially or completely) absorbable disjuncts for a concept C if *absorbable* = *true* and a 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.

The absorption itself is invoked by the absorbTBox procedure (see Algorithm 5). Each axiom of $\mathcal{T}$ is processed and the resulting axioms are added to $\mathcal{T}'$. For a possibly absorbable axiom, the set of all absorbable disjuncts is extracted with collectDisjuncts from the corresponding disjunction and then absorbJoined is called for generating the absorption. Please note that if all disjuncts of an axiom can be completely absorbed, then an empty disjunc-

[2] $\mathcal{T}$ and $\mathcal{T}'$ are used as global variables, i.e., they can be directly accessed from all functions and algorithms. Hence, the algorithms have side effects, which can, however, be eliminated by extending the algorithms with additional parameters.

Algorithm 4 absorbConcept(C)

```
Output: Returns the atomic concept for the absorption of $C$
 1: if $C = C_1 \sqcap C_2$ then
 2:     $A_1 \leftarrow$ absorbJoined(collectDisjuncts($C_1$, $true$))
 3:     $A_2 \leftarrow$ absorbJoined(collectDisjuncts($C_2$, $true$))
 4:     $T \leftarrow$ fresh atomic concept
 5:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_1 \sqsubseteq T, A_2 \sqsubseteq T\}$
 6:     return $T$
 7: else if $C = \forall r.C'$ then
 8:     $A_{nb} \leftarrow$ absorbJoined(collectDisjuncts($C'$, $true$))
 9:     $T \leftarrow$ fresh atomic concept
10:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall \mathsf{inv}(r).T\}$
11:     return $T$
12: else if $C = \leqslant n\, r.C'$ then
13:     $A_{nb} \leftarrow$ absorbJoined(collectDisjuncts(nnf($\neg C'$), $true$))
14:     $T \leftarrow$ fresh atomic concept
15:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall \mathsf{inv}(r).T\}$
16:     return $T$
17: else if $C = \neg\{a\}$ then
18:     $T \leftarrow$ fresh atomic concept
19:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{\{a\} \sqsubseteq T\}$
20:     return $T$
      ...
21: else if $C = \neg A$ then
22:     if $A \equiv C' \notin \mathcal{T}$ then
23:         return $A$
24:     else
25:         for all $A \equiv C' \in \mathcal{T}$ do
26:             $A' \leftarrow$ absorbJoined(collectDisjuncts(nnf($\neg C'$), $true$))
27:             $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A' \sqsubseteq A^+\}$
28:         end for
29:         return $A^+$
30:     end if
31: end if
```

tion is created (line 15 and 19), which corresponds to $\bot$. The methods absorbJoined (Algorithm 3) and absorbConcept (Algorithm 4) are recursively calling each other, whereby absorbJoined is joining several (possibly fresh) atomic concepts with binary absorption axioms and absorbConcept creates the absorption for a specific, absorbable concept, i.e., absorbJoined handles the absorbable disjunctions, whereas the remaining absorbable concepts are handled by absorbConcept.

For instance, a concept of the form $\forall r.C'$ can be absorbed (lines 7–11 of Algorithm 4) 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* technique [30].

Of course, the absorption can be extended to concept constructors of more expressive DLs (which we have denoted by "..." in the algorithms), for example, the $\neg\exists r.\mathsf{Self}$ concept of $\mathcal{SROIQ}$ can be partially absorbed with $\top \sqsubseteq \forall r^{-}.A$. It is even possible to extend the partial absorption technique to datatypes [27]. However, it is clear that some concept constructors, such as existential restrictions of the form $\exists r.C$, cannot be directly absorbed since they do not provide conditions that could be used to trigger other concepts. In particular, to prove that an existential restriction $\exists r.C$ is satisfied, the tableau algorithm has to show that an r-

Algorithm 5 absorbTBox

```
Output: Creates a new TBox $\mathcal{T}'$ with absorbed axioms for the original TBox $\mathcal{T}$
 1: $\mathcal{T}' \leftarrow \emptyset$
 2: for all $X \in \mathcal{T}$ do
 3:     if $X = A \equiv C$ then
 4:         $A' \leftarrow$ absorbJoined(collectDisjuncts(nnf($\neg C$), $true$))
 5:         if isCA(nnf($\neg C$)) then
 6:             $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A \sqsubseteq C, A' \sqsubseteq A, A' \sqsubseteq A^+\}$
 7:         else if $|\{A \sqsubseteq C' \in \mathcal{T}\} \cup \{A \equiv C' \in \mathcal{T}\}| > 1$ then
 8:             $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A \sqsubseteq C, A' \sqsubseteq$ nnf($\neg C \sqcup A$)$, A' \sqsubseteq A^+\}$
 9:         else
10:             $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A \equiv C, A' \sqsubseteq A^+\}$
11:         end if
12:     else if $X = C \sqsubseteq D$ or $X = C \equiv D$ then
13:         $A' \leftarrow$ absorbJoined(collectDisjuncts(nnf($\neg C \sqcup D$), $true$))
14:         $\{D_1, \ldots, D_n\} \leftarrow$ collectDisjuncts(nnf($\neg C \sqcup D$), $false$)
15:         $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A' \sqsubseteq D_1 \sqcup \ldots \sqcup D_n\}$
16:         if $X = C \equiv D$ then
17:             $A'' \leftarrow$ absorbJoined(collectDisjuncts(nnf($\neg D \sqcup C$), $true$))
18:             $\{C_1, \ldots, C_m\} \leftarrow$ collectDisjuncts(nnf($\neg D \sqcup C$), $false$)
19:             $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A'' \sqsubseteq C_1 \sqcup \ldots \sqcup C_m\}$
20:         end if
21:     end if
22: end for
```

neighbour with C in its label can exist, i.e., it is indeed necessary to build such a neighbour node.

For an atomic concept A, which is completely defined by an axiom of the form $A \equiv C$ in $\mathcal{T}$, it is often inefficient to decompose the axiom $A \equiv C$ into $A \sqsubseteq C$ and $C \sqsubseteq A$, because nnf($\neg C$) might not be completely absorbable and then the disjunction $\neg C \sqcup A$ has to be processed for the nodes in the completion graph. In order to determine the satisfiability of a concept, it is, however, for many nodes not relevant whether A or $\neg A$ is in their label as long as it can be ensured that one of both alternatives is not causing a clash. Obviously, if the atomic concept A is only defined once in the knowledge base, then C or $\neg C$ and therefore also A or $\neg A$ must be satisfiable and only in this case the axiom $A \equiv C$ can be directly handled by a separate rule, which unfolds A to C and $\neg A$ to $\neg C$. Thus, there is no need to rewrite the axiom $A \equiv C$ for an efficient handling in $\mathcal{T}'$ (Algorithm 5, line 10). If there are more definitions for A, then $A \sqsubseteq C$ as well as $C \sqsubseteq A$ must be explicitly represented in the new TBox $\mathcal{T}'$ so that possible interactions between these several definitions can be handled. Of course, if nnf($\neg C$) is partially absorbable, then the disjunction $\neg C \sqcup A$ can be triggered with A', which is generated for the absorption of nnf($\neg C$) (Algorithm 5, line 7).

Furthermore, for each atomic concept A, which is completely defined by an axiom $A \equiv C$ in $\mathcal{T}$, a candidate concept A^+ is generated (line 6,8 and 10 of Algorithm 5). An occurrence of A^+ in the label of a node signalises that the node might be an instance of the concept A, which is obviously the case if A itself is in the label, but this is also the case if $\neg A$ cannot safely be added. As described above, we are, however, not interested in forcing the decision between A and $\neg A$ for all nodes in the completion graph. In contrast, we generate the candidate concept A^+ that can be used in the absorption instead of A if $\neg A$ occurs, whereby it is often possible to delay branching significantly. The creation of A^+ is realised by absorbing the concept nnf($\neg C$) for the axiom $A \equiv C$ for which the disjunction $\neg C \sqcup A$ is represented as $\neg A \sqsubseteq \neg C$. The absorption of nnf($\neg C$) generates the atomic concept A' and the axiom $A' \sqsubseteq A^+$ is added to $\mathcal{T}'$ (line 8 of Algorithm 5). If nnf($\neg C$) is not absorbable, then the absorption returns $\top$ for A' and $\top \sqsubseteq A^+$ is added to $\mathcal{T}'$. Note, we also generate the candidate concepts in the absorbConcept function (lines 25-28 of Algorithm 4) in order to make the absorption of a separate concept complete for the proofs. Of course, if a candidate concept is already created, then it is not necessary to create it again. Besides using candidate concepts in the absorption, they can also be used to identify completely defined concepts as possible subsumers [6] and, therefore, it is almost always useful to generate these candidate concepts.

The absorbJoined function creates binary absorption axioms (Algorithm 3, 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. In principle, it is not necessary to always create new axioms with fresh atomic concepts for the absorption of identical concepts. In practice, the binary absorption axioms as well as the axioms for absorbing specific concepts can be reused.

```
absorbTBox
  collectDisjuncts( $\forall r.(\neg A \sqcup \forall r.\neg\{b\}) \sqcup \exists s.\{a\}$, true )
    $\rightsquigarrow \{\forall r.(\neg A \sqcup \forall r.\neg\{b\})\}$
  absorbJoined( $\{\forall r.(\neg A \sqcup \forall r.\neg\{b\})\}$ )
    absorbConcept( $\forall r.(\neg A \sqcup \forall r.\neg\{b\})$ )
      collectDisjuncts( $\neg A \sqcup \forall r.\neg\{b\}$, true )
        $\rightsquigarrow \{\neg A, \forall r.\neg\{b\}\}$
      absorbJoined( $\{\neg A, \forall r.\neg\{b\}\}$ )
        absorbConcept( $\neg A$ )
          $\rightsquigarrow A$
        absorbConcept( $\forall r.\neg\{b\}$ )
          collectDisjuncts( $\neg\{b\}$, true )
            $\rightsquigarrow \{\neg\{b\}\}$
          absorbJoined( $\{\neg\{b\}\}$ )
            absorbConcept( $\neg\{b\}$ )
              $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{\{b\} \sqsubseteq T_1\}$
              $\rightsquigarrow T_1$
            $\rightsquigarrow T_1$
          $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{T_1 \sqsubseteq \forall r^-.T_2\}$
          $\rightsquigarrow T_2$
        $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{(A \sqcap T_2) \sqsubseteq T_3\}$
        $\rightsquigarrow T_3$
      $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{T_3 \sqsubseteq \forall r^-.T_4\}$
      $\rightsquigarrow T_4$
    $\rightsquigarrow T_4$
  collectDisjuncts( $\forall r.(\neg A \sqcup \forall r.\neg\{b\}) \sqcup \exists s.\{a\}$, false )
    $\rightsquigarrow \{\exists s.\{a\}\}$
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{T_4 \sqsubseteq \exists s.\{a\}\}$
```

Fig. 1 Call diagram for the absorbTBox procedure assuming $\mathcal{T} = \{\exists r.(A \sqcap \exists r.\{b\}) \sqsubseteq \exists s.\{a\}\}$, where the axioms corresponds to the disjunction $\forall r.(\neg A \sqcup \forall r.\neg\{b\}) \sqcup \exists s.\{a\}$ in negation normal form

Example 1 As an example, we consider the TBox $\mathcal{T} = \{C \sqsubseteq D\}$ with $C = \exists r.(A \sqcap \exists r.\{b\})$ and $D = \exists s.\{a\}$. Note that $\mathsf{nnf}(\neg C \sqcup D) = \forall r.(\neg A \sqcup \forall r.\neg\{b\}) \sqcup \exists s.\{a\}$. Figure 1 shows the call diagram for the procedure absorbTBox. We indicate the return values of the sub-procedures with $\rightsquigarrow$, but we omit the calls to isPA and isCA. Note that isPA$(\cdot)$ is *true* for

$\forall r.(\neg A \sqcup \forall r.\neg\{b\})$, $\neg A$, $\forall r.\neg\{b\}$, and $\neg\{b\}$, while $\neg\mathsf{isCA}(\cdot)$ is *true* for $\exists s.\{a\}$. During the recursive procedure, the TBox $\mathcal{T}'$ is step-by-step constructed, based on the axiom in $\mathcal{T}$ and when the procedure terminates, we have in $\mathcal{T}'$:

$$\begin{array}{lll} \{b\} \sqsubseteq T_1 & T_1 \sqsubseteq \forall r^-.T_2 & (A \sqcap T_2) \sqsubseteq T_3 \\ T_3 \sqsubseteq \forall r^-.T_4 & T_4 \sqsubseteq \exists s.\{a\}. & \end{array}$$

where $T_1, \ldots, T_4$ are fresh atomic concepts generated by the absorption. To process the one and only axiom in $\mathcal{T}$, the $\mathsf{absorbJoined}$ function is called (line 13 of Algorithm 5) for the absorbable parts of $\forall r.(\neg A \sqcup \forall r.\neg\{b\}) \sqcup \exists s.\{a\}$, which is only the disjunct $\forall r.(\neg A \sqcup \forall r.\neg\{b\})$. First, the $\mathsf{absorbConcept}$ function processes the $\forall$-concept by recursively absorbing its filler $\neg A \sqcup \forall r.\neg\{b\}$, which is split into the disjuncts $\neg A$ and $\forall r.\neg\{b\}$. The disjunct $\neg A$ does not require further absorption since A is already an atomic concept. The other disjunct requires another recursion to first generate $\{b\} \sqsubseteq T_1$ and, afterwards, the associated propagation of the trigger T_2 over the r^--role with $T_1 \sqsubseteq \forall r^-.T_2$. The $\mathsf{absorbJoined}$ function is then joining the atomic concepts A and T_2 by the new binary axiom $(A \sqcap T_2) \sqsubseteq T_3$. Then, the absorption of the outer $\forall$-concept can be finished by adding the axiom $T_3 \sqsubseteq \forall r^-.T_4$. The remaining non-absorbable part of the disjunction is handled with $T_4 \sqsubseteq \exists s.\{a\}$. The new axioms are not causing any non-determinism and, furthermore, $\{b\} \sqsubseteq T_1$ can be handled very efficiently as an assertion ($\{b\} \sqsubseteq T_1$ is equivalent to the assertion $T_1(b)$), by *lazy unfolding* [2] (for axioms of the form $A \sqsubseteq C$) and by a binary absorption rule (for axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$).

Example 2 In order to also illustrate the absorption for more complex axioms with language features of more expressive Description Logics, let the TBox $\mathcal{T}$ contain the axioms

$$A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3 \qquad (A_4 \sqcup \{a\}) \sqcap \exists s.(A_1 \sqcap \geqslant 3\, r.A_5) \sqsubseteq A_6,$$

which can be absorbed into a new TBox $\mathcal{T}'$ consisting of the following axioms:

$$\begin{array}{lll} A_2 \sqsubseteq \forall r^-.T_1 & A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3 & T_1 \sqsubseteq A_1^+ \\ \{a\} \sqsubseteq T_2 & A_4 \sqsubseteq T_3 & T_2 \sqsubseteq T_3 \\ A_5 \sqsubseteq \forall r.T_4 & (A_1^+ \sqcap T_4) \sqsubseteq T_5 & T_5 \sqsubseteq \forall s^-.T_6 \\ (T_3 \sqcap T_6) \sqsubseteq T_7 & T_7 \sqsubseteq \forall s.(\neg A_1 \sqcup \leqslant 2\, r.A_5) \sqcup A_6. & \end{array}$$

First, the axiom $A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3$ is handled by line 4 and 10 of procedure $\mathsf{absorbTBox}$, which generate the first three axioms of $\mathcal{T}'$, i.e., $A_2 \sqsubseteq \forall r^-.T_1$, $A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3$, and $T_1 \sqsubseteq A_1^+$. The axioms $A_2 \sqsubseteq \forall r^-.T_1$ and $T_1 \sqsubseteq A_1^+$ are used to trigger the candidate concept A_1^+ of the completely defined concept A_1, and $A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3$ is used for the unfolding of A_1. Similarly to the previous example, the other eight axioms of $\mathcal{T}'$ are created for the absorption of $(A_4 \sqcup \{a\}) \sqcap \exists s.(A_1 \sqcap \geqslant 3\, r.A_5) \sqsubseteq A_6$. Again, $T_1, \ldots, T_7$ are the fresh atomic concepts and although not all disjunctions can be eliminated for these axioms, it is nevertheless possible to optimise the structure of the axioms for a more efficient handling in the tableau algorithm. Additionally to the axioms of the form $\{a\} \sqsubseteq C$, $A \sqsubseteq C$, $(A_1 \sqcap A_2) \sqsubseteq C$, which are also created in the previous example, the absorption ensures that all remaining axioms of the form $A \equiv C$ can be efficiently handled by lazy unfolding, i.e., A can be unfolded to C and $\neg A$ to $\neg C$ (exploiting that $C \sqsubseteq A$ is equivalent to $\neg A \sqsubseteq \neg C$).[3]

The new axioms dramatically delay or completely avoid non-deterministic branching caused by disjunctions. For example, without absorption, the disjunction $(\neg A_4 \sqcap \neg\{a\}) \sqcup$

[3] Note that this only works as long as there are no other axioms of the form $A \sqsubseteq D$ or $A \equiv D$ with $D \neq C$ in the knowledge base, which is, however, ensured by the absorption.

$\forall s.(\neg A_1 \sqcup \leqslant 2\, r.A_5) \sqcup A_6$ obtained from the second axiom in the unprocessed TBox $\mathcal{T}$ has to be processed for each node in a completion graph. With absorption, we have one remaining disjunction, $\forall s.(\neg A_1 \sqcup \leqslant 2\, r.A_5) \sqcup A_6$, which is triggered by T_7. The concept T_7 is only added to the label of a node if the original disjunction is not trivially satisfiable. For example, if a node does not have any s-neighbours, then T_6 would not be added to the label and, as a consequence of the axiom $(T_3 \sqcap T_6) \sqsubseteq T_7$, T_7 would also not be added. In this case, it would not be necessary to make non-deterministic decisions since the second disjunct $\forall s.(\neg A_1 \sqcup \leqslant 2\, r.A_5)$ of the original disjunction is trivially satisfiable. Please also note that the decision between $\neg A_1$ and A_1 is not enforced for every node in a completion graph for $\mathcal{T}'$ and, nevertheless, A_1 can be partially used in the absorption by replacing it with A_1^+. As long A_1^+ is not in the label of a node of a fully expanded completion graph, we know that $\neg A_1$ must be satisfiable, because $\forall r.\neg A_2$ of the disjunction $\forall r.\neg A_2 \sqcup \exists r.\neg A_3$ cannot not cause a clash. This can be utilised in the absorption of the other axiom, because as long as we know that $\neg A_1$ can be added to a s-neighbour without causing a clash, we also know that the axiom $(A_4 \sqcup \{a\}) \sqcap \exists s.(A_1 \sqcap \geqslant 3\, r.A_5) \sqsubseteq A_6$ can be trivially satisfied.

3.1 Related Absorption Techniques

In comparison to other absorption techniques, the presented approach is often able to further delay non-determinism. In particular, this is the case for axioms that use more expressive concept constructors, such as Axiom (3):

$$\exists r.(A \sqcap \forall r.B_1) \sqsubseteq B_2. \tag{3}$$

Without absorption, this axioms has to be processed as

$$\top \sqsubseteq \forall r.(\neg A \sqcup \exists r.\neg B_1) \sqcup B_2 \tag{4}$$

which would produce a significant amount of non-determinism. Clearly, none of the parts of (3) can be absorbed completely, but it is nevertheless possible to delay the processing of the disjunctions until there is an r-neighbour with the concept A in its label. In order to capture this, the presented absorption algorithm 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 Axiom (5) and (6):

$$A \sqsubseteq \forall r^-.T' \tag{5}$$

$$T' \sqsubseteq \forall r.(\neg A \sqcup \exists r.\neg B_1) \sqcup B_2 \tag{6}$$

In comparison, the original binary absorption algorithm creates Axiom (7) and (8):[4]

$$A \sqsubseteq \exists r.\neg B_1 \sqcup \forall r^-.T' \tag{7}$$

$$T' \sqsubseteq B_2 \tag{8}$$

Consequently, it is necessary to process disjunctions for all those nodes for which the concept A is derived, whereas the processing of disjunctions can further be delayed with the partial absorption technique until there is also an r-neighbour. Although Axiom (6) gives

[4] The binary absorption algorithm presented in [13] has a looping bug for concepts of the form $\forall r.C$, which can, however, easily be repaired. We simply ignored the repeated absorption of concepts that are introduced by the algorithm.

the impression that it contains more disjunctions and thus produces more non-determinism, this is usually not the case in practice. For example, only the disjunct $\exists r.\neg B_1$ can be applied for the inner disjunction $\neg A \sqcup \exists r.\neg B_1$ of Axiom (6) on those nodes for which Axiom (5) has been triggered.

It is also well-known that clausification for the hypertableau algorithm reduces many cases of non-determinism [22]. For Axiom (3), however, clausification creates the DL-clauses (9) and (10):

$$r(x,y) \rightarrow B_2(x) \vee T'(y) \tag{9}$$

$$T'(x) \wedge A(x) \rightarrow \exists r.\neg B_1(x) \tag{10}$$

Hence, the hypertableau algorithm has to choose between B_2 and T' for every node with an r-neighbour, whereas the partial absorption technique allows for further delaying the processing of disjunctions until the node also has the concept A in its label. Besides that, the presented partial absorption technique shares other interesting features with clausification of the hypertableau algorithm. In particular, both use conditions of more expressive concept constructors to trigger the processing of disjunctions. For instance, Axiom (11) is automatically satisfied if there is no r-successor with the concept A in its label.

$$\geqslant 2\, r.A \sqsubseteq B \tag{11}$$

This is utilised by the partial absorption by generating Axiom (12) and (13), and also by clausification for the hypertableau algorithm, which generates the DL-clause (14):

$$A \sqsubseteq \forall r^{-}.T \tag{12}$$

$$T \sqsubseteq\, \leqslant 1\, r.A \sqcup B \tag{13}$$

$$r(x,y_1) \wedge A(y_1) \wedge r(x,y_2) \wedge A(y_2) \rightarrow B(x) \vee y_1 \approx y_2 \tag{14}$$

Note that clausification eliminates at-most cardinality restrictions, whereas the partial absorption has to use the concept $\leqslant 1\, r.A$ in Axiom (13) since $\geqslant 2\, r.A$ is not completely absorbable. Consequently, a ch-rule is not required in the hypertableau algorithm, which possibly further reduces non-determinism in comparison to a standard tableau algorithm that has to choose between $\neg A$ and A for all r-neighbours of nodes with T in their label.

3.2 Correctness

Termination for the absorption algorithm is ensured by the acyclicity of the axioms. However, if it is ensured that the candidate concepts are exclusively created by the absorbTBox procedure for all atomic concepts, which are completely defined with axioms of the form $A \equiv C$ in $\mathcal{T}$, then in the recursion between absorbJoined and absorbConcept only the current axiom has to be processed. Since we generate the candidate concepts in absorbConcept only to make the functions absorbJoined and absorbConcept complete for the absorption of a concept for the proofs, the acyclicity restriction would not be necessary for termination.

In the following we prove the correctness of our modified absorption algorithm. We first show that the complete absorption of a disjunct of an axiom is correct, i.e., it preserves the satisfiability (Lemma 1 and Lemma 2), and then we show that the correctness of a partially absorbed concept disjunct can be reduced to the complete absorption (Lemma 3).

Lemma 1 *Let $\mathcal{T}$ denote a TBox, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an interpretation such that $\mathcal{I} \models \mathcal{T}$, C a concept that is completely absorbable, A the concept returned by* absorbJoined*($\{C\}$), and $\mathcal{T}'$ the extension of $\mathcal{T}$ with all the axioms created by* absorbJoined*($\{C\}$), then*

1. *for every extension $\mathcal{I}'$ of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{T}'$, it holds that $\mathcal{I}' \models \mathcal{T}$,*
2. *for every extension $\mathcal{I}'$ of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{T}'$, it holds for all $\delta \in \Delta^{\mathcal{I}'}$ that $\delta \in A^{\mathcal{I}'}$ if $\delta \notin C^{\mathcal{I}'}$, and*
3. *there exists an interpretation $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ with $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ such that $\mathcal{I}' \models \mathcal{T}'$ and, for each $\delta \in \Delta^{\mathcal{I}'}$, $\delta \in A^{\mathcal{I}'}$ only if $\delta \notin C^{\mathcal{I}'}$.*

Proof (*Claim 1*) Since $\mathcal{T}'$ is an extension of $\mathcal{T}$, it trivially follows that $\mathcal{I}' \models \mathcal{T}$.
(*Claim 2*) We first prove the simple cases where C is completely absorbable and afterwards we show by induction that the lemma also holds for the complex cases.

- If C is of the form $\neg A$ and $A \equiv C' \notin \mathcal{T}$, then absorbConcept($C$) directly returns A, which is then also returned by absorbJoined($\{C\}$). Thus, if $\delta \in \Delta^{\mathcal{I}'}$ and $\delta \notin C^{\mathcal{I}'}$, i.e., $\delta \notin (\neg A)^{\mathcal{I}'}$, then $\delta \in A^{\mathcal{I}'}$. Hence, the lemma holds if C is of the form $\neg A$.
- If C is of the form $\neg\{a\}$, then absorbConcept(C) adds the axiom $\{a\} \sqsubseteq A$ to $\mathcal{T}'$ and returns A, which is then also returned by absorbJoined($\{C\}$). Thus, if $\delta \notin C^{\mathcal{I}'}$, i.e., $\delta \notin (\neg\{a\})^{\mathcal{I}'}$, then $\delta \in a^{\mathcal{I}'}$ and because of the assumption $\mathcal{I}' \models \mathcal{T}'$, i.e., $\mathcal{I}' \models \{a\} \sqsubseteq A$, it follows that $\delta \in A^{\mathcal{I}'}$. Hence, the lemma holds if C is of the form $\neg\{a\}$.

For the complex cases we assume that all nested disjunctions are replaced by a single disjunction with all disjuncts, i.e., $(C_1 \sqcup (C_2 \sqcup C_3))$ is replaced by $(C_1 \sqcup C_2 \sqcup C_3)$. Furthermore, we automatically decompose a disjunction into the set of disjuncts by calling absorbJoined. This conversion is also done by the algorithm with the collectDisjuncts function, which is always called before absorbJoined. Therefore, we can omit collectDisjuncts for calling absorbJoined, which improves the readability. Now, for a disjunct C_j, it follows that C_j is not a disjunction itself and it also follows that absorbJoined($\{C_j\}$) only returns the atomic concept that is returned by absorbConcept(C_j).

Let $C_1, \ldots, C_n$ be completely absorbable concepts and $A_1, \ldots, A_n$ the atomic concepts returned by absorbJoined($\{C_1\}$), . . . ,absorbJoined($\{C_n\}$). By our induction hypothesis, the lemma holds for A_1 w.r.t. $C_1, \ldots, A_n$ w.r.t. C_n.

- If C is now of the form $C_1 \sqcup \ldots \sqcup C_n$, then absorbJoined($\{C_1, \ldots, C_n\}$) collects the atomic concepts $A_1, \ldots, A_n$ by calling absorbConcept(C_j) for each C_j, $1 \leqslant j \leqslant n$, and creates the binary absorption axioms $(A_1 \sqcap A_2) \sqsubseteq T_1, (T_1 \sqcap A_3) \sqsubseteq T_2, \ldots, (T_{n-2} \sqcap A_n) \sqsubseteq A$. Thus, if $\delta \notin C^{\mathcal{I}'}$, i.e., $\delta \notin (C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'}$, then $\delta \in (\neg C_1 \sqcap \ldots \sqcap \neg C_n)^{\mathcal{I}'}$ and, as a consequence, $\delta \in (\neg C_j)^{\mathcal{I}'}$ for $1 \leqslant j \leqslant n$. Therefore, by the induction hypothesis we have $\delta \in A_j^{\mathcal{I}'}$ for all $1 \leqslant j \leqslant n$. Thus, $\delta \in A_1^{\mathcal{I}'}$ and $\delta \in A_2^{\mathcal{I}'}$ and since the interpretation $\mathcal{I}' \models \mathcal{T}'$ with $\{(A_1 \sqcap A_2) \sqsubseteq T_1, (T_1 \sqcap A_3) \sqsubseteq T_2, \ldots, (T_{n-2} \sqcap A_n) \sqsubseteq A\} \subseteq \mathcal{T}'$ it follows that $\delta \in T_1^{\mathcal{I}'}, \delta \in T_2^{\mathcal{I}'}, \ldots, \delta \in A^{\mathcal{I}'}$. Hence, the lemma holds by induction if C is of the form $C_1 \sqcup \ldots \sqcup C_n$.
- If C is of the form $C_1 \sqcap C_2$, then absorbJoined($\{C\}$) returns A, which is obtained by calling absorbConcept(C), where additionally the axioms $A_1 \sqsubseteq A$ and $A_2 \sqsubseteq A$ are created. If $\delta \notin C^{\mathcal{I}'}$, i.e., $\delta \notin (C_1 \sqcap C_2)^{\mathcal{I}'}$, then $\delta \in (\neg C_1 \sqcup \neg C_2)^{\mathcal{I}'}$. There are now two cases: If $\delta \in (\neg C_1)^{\mathcal{I}'}$, then by the induction hypothesis we have $\delta \in A_1^{\mathcal{I}'}$ and, due to the axiom $A_1 \sqsubseteq A$, we have $\delta \in A^{\mathcal{I}'}$. For the other case we have $\delta \in (\neg C_2)^{\mathcal{I}'}$, by the induction hypothesis it follows that $\delta \in A_2^{\mathcal{I}'}$ and, due to the axiom $A_2 \sqsubseteq A$, we also have $\delta \in A^{\mathcal{I}'}$. Hence, the lemma holds by induction if C is of the form $C_1 \sqcap C_2$.

- If C is of the form $\forall r.C_1$, then $\mathsf{absorbConcept}(C)$ creates $A_1 \sqsubseteq \forall\, \mathsf{inv}(r).A$ and A is returned by $\mathsf{absorbJoined}(\{C\})$. Thus, if $\delta \notin C^{\mathcal{I}'}$, i.e., $\delta \notin (\forall r.C_1)^{\mathcal{I}'}$, then $\delta \in (\exists r.\neg C_1)^{\mathcal{I}'}$. It follows that there exists $\gamma \in \Delta^{\mathcal{I}'}$, $(\delta, \gamma) \in r^{\mathcal{I}'}$ with $\gamma \in (\neg C_1)^{\mathcal{I}'}$ and by the induction hypothesis we have $\gamma \in A_1^{\mathcal{I}'}$. As a consequence of the axiom $A_1 \sqsubseteq \forall\, \mathsf{inv}(r).A$ we also have $\delta \in A^{\mathcal{I}'}$. Hence, the lemma holds by induction if C is of the form $\forall r.C_1$.
- If C is of the form $\neg A'$ and A' is completely defined by the axioms $A' \equiv C_1' \in \mathcal{T}, \ldots, A' \equiv C_n' \in \mathcal{T}$, then the candidate concept A'^{+} is returned for A by the absorption. Let $C_1 = \neg C_1', \ldots, C_n = \neg C_n'$, then the candidate concept A'^{+} is implied by $A_1, \ldots, A_n$, which are the atomic concepts created for absorbing $\neg C_1', \ldots, \neg C_n'$, i.e., $C_1, \ldots, C_n$. Now, this case is similar to the case where C is of the form $C_1 \sqcup C_2$, because if $\delta \notin C^{\mathcal{I}'}$, i.e., $\delta \notin (\neg A')^{\mathcal{I}'}$, then $\delta \in A'^{\mathcal{I}'}$ and as a consequence of the axioms $A' \equiv \neg C_1, \ldots, A' \equiv \neg C_n$ we also have $\delta \in (\neg C_j)^{\mathcal{I}'}$ for $1 \leqslant j \leqslant n$. Therefore, by the induction hypothesis it follows that $\delta \in A_j^{\mathcal{I}'}$ for all $1 \leqslant j \leqslant n$ and, because of the axioms that imply the candidate concept A'^{+}, we also have $\delta \in (A'^{+})^{\mathcal{I}'}$. Hence, the lemma holds by induction if C is of the form $\neg A'$ and A' is completely defined with axioms of the form $A' \equiv C'$, where $\mathsf{nnf}(\neg C')$ is completely absorbable.

(*Claim 3*) We construct the interpretation $\mathcal{I}'$ from $\mathcal{I}$ such that, for each $\delta \in \Delta^{\mathcal{I}'}$, $\delta \in A^{\mathcal{I}'}$ only if $\delta \notin C^{\mathcal{I}'}$. Therefore, let $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ be an interpretation with $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}'}$ reduced from $\cdot^{\mathcal{I}}$ such that only the atomic concepts, atomic roles, and individuals occurring in $\mathcal{T}$ are interpreted. Obviously, it still holds that $\mathcal{I}' \models \mathcal{T}$ since the interpretation of all axioms in $\mathcal{T}$ coincides with $\mathcal{I}$. We now define the interpretation of the fresh atomic concepts $A_1, \ldots, A_m$ introduced for the absorption of C in $\mathcal{I}'$. Note that we treat absorption axioms of the form $A' \sqsubseteq \forall r.A_i$ in their equivalent form $\exists\, \mathsf{inv}(r).A' \sqsubseteq A_i$.

Now, for each $\delta \in \Delta^{\mathcal{I}'}$, for $1 \leqslant i \leqslant m$ and for each axiom $H \sqsubseteq A_i$ generated by the absorption, we exhaustively add δ to $A_i^{\mathcal{I}'}$ if

(i) $H = A'$ and $\delta \in A'^{\mathcal{I}'}$,
(ii) $H = \{a\}$ and $\delta \in \{a\}^{\mathcal{I}'}$,
(iii) $H = (A' \sqcap A'')$ and $\delta \in A'^{\mathcal{I}'} \cap A''^{\mathcal{I}'}$, or
(iv) $H = \exists\, \mathsf{inv}(r).A'$ and $\delta \in (\exists\, \mathsf{inv}(r).A')^{I'}$, i.e., δ has some $\mathsf{inv}(r)$-neighbour γ such that $(\gamma, \delta) \in r^{\mathcal{I}'}$ and $\gamma \in A'^{\mathcal{I}'}$.

For each $\delta \in \Delta^{\mathcal{I}'}$, we have $\delta \in A_i^{\mathcal{I}'}$, only if δ satisfies the left-hand side of an axiom $A' \sqsubseteq A_i$, $\{a\} \sqsubseteq A_i$, $(A' \sqcap A'') \sqsubseteq A_i$, or $\exists\, \mathsf{inv}(r).A' \sqsubseteq A_i$. Consequently, it follows that $\mathcal{I}' \models \mathcal{T}'$. Furthermore, $\delta \notin A^{\mathcal{I}'}$ if $\delta \in C^{\mathcal{I}'}$, because of the following cases:

- If C is of the form $\neg A$, $A \equiv C' \notin \mathcal{T}$ and $\delta \in C^{\mathcal{I}'}$, i.e., $\delta \in (\neg A)^{\mathcal{I}'}$, then $\delta \notin A^{\mathcal{I}'}$.
- If C is of the form $\neg\{a\}$ for which the absorption has generated $\{a\} \sqsubseteq A$, and if $\delta \in C^{\mathcal{I}'}$, i.e., $\delta \in (\neg\{a\})^{\mathcal{I}'}$, then $\delta \notin \{a\}^{\mathcal{I}'}$ and then $\delta \notin A^{\mathcal{I}'}$, because the left-hand side of $\{a\} \sqsubseteq A$ is not satisfied and there is also no other axiom that implies A (A is freshly used for the generation of the axiom $\{a\} \sqsubseteq A$).

For the remaining cases, we again assume that the lemma holds for A_1 w.r.t. $C_1, \ldots, A_n$ w.r.t. C_n, where $A_1, \ldots, A_n$ are the atomic concepts for absorbing the completely absorbable concepts $C_1, \ldots, C_n$. Therefore, it follows by induction that $\delta \notin A^{\mathcal{I}'}$ if $\delta \in C^{\mathcal{I}'}$, because of the following cases:

- If C is of the form $C_1 \sqcup \ldots \sqcup C_n$ and $\delta \in C^{\mathcal{I}'}$, i.e., $\delta \in (C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}'}$, then there exists a C_j, $1 \leqslant j \leqslant n$ with $\delta \in C_j^{\mathcal{I}'}$. By the induction hypothesis it follows that $\delta \notin A_j^{\mathcal{I}'}$ and by the binary axiom chain $(A_1 \sqcap A_2) \sqsubseteq T_1, (T_1 \sqcap A_3) \sqsubseteq T_2, \ldots, (T_{j-2} \sqcap A_j) \sqsubseteq T_{j-1}, \ldots, (T_{n-2} \sqcap A_n) \sqsubseteq A$, which is generated for absorbing $C_1 \sqcup \ldots \sqcup C_n$, we have $\delta \notin A^{\mathcal{I}'}$, because the left-hand side of the axiom $(T_{j-2} \sqcap A_j) \sqsubseteq T_{j-1}$ cannot be satisfied.

- If C is of the form $C_1 \sqcap C_2$ and $\delta \in C^{\mathcal{I}'}$, i.e., $\delta \in (C_1 \sqcap C_2)^{\mathcal{I}'}$, then $\delta \in C_1^{\mathcal{I}'}$ and $\delta \in C_2^{\mathcal{I}'}$. By the induction hypothesis we have $\delta \notin A_1^{\mathcal{I}'}$ and $\delta \notin A_2^{\mathcal{I}'}$. The left-hand side of the axioms $A_1 \sqsubseteq A$ and $A_2 \sqsubseteq A$ is not satisfied and the absorptions does not generate other axioms that imply A. Thus, δ is not added to $A^{\mathcal{I}'}$.
- If C is of the form $\forall r.C_1$ and $\delta \in C^{\mathcal{I}'}$, i.e., $\delta \in (\forall r.C_1)^{\mathcal{I}'}$, then for all $\gamma \in \Delta^{\mathcal{I}'}$ with $(\delta, \gamma) \in r^{\mathcal{I}'}$ we also have $\gamma \in C_1^{\mathcal{I}'}$. By the induction hypothesis it follows that $\gamma \notin A_1^{\mathcal{I}}$ and since the left-hand side of the generated axiom $\exists \mathsf{inv}(r).A_1 \sqsubseteq A_i$ is not satisfied, and there are not any other axioms that imply A, we do not add δ to $A^{\mathcal{I}'}$ and, thus, $\delta \notin A^{\mathcal{I}'}$.
- If C is of the form $\neg A'$, A' is completely defined by the axioms $A' \equiv C'_1 \in \mathcal{T}, \ldots, A' \equiv C'_n \in \mathcal{T}$, and $\delta \in C^{\mathcal{I}'}$, i.e., $\delta \in (\neg A')^{\mathcal{I}'}$, then, for $C_1 = \neg C'_1, \ldots, C_n = \neg C'_n$ and, as a consequence of the axioms $A' \equiv C'_1, \ldots, A' \equiv C'_n$, we also have $\delta \in C_1^{\mathcal{I}'}, \ldots, \delta \in C_n^{\mathcal{I}'}$. By the induction hypothesis we have $\delta \notin A_1^{\mathcal{I}'}, \ldots, \delta \notin A_n^{\mathcal{I}'}$ and, thus, the left-hand side of all axioms that imply the candidate concept $A = A'^{+}$ is not satisfied. Therefore, we have $\delta \notin A^{\mathcal{I}'}$. □

We can now use Lemma 1 to show the correctness of the absorption for the case of a completely absorbable concept C in an axiom $C \sqsubseteq D$.

Lemma 2 *For $\mathcal{T}$ a TBox and $C \sqcup D$ a disjunction, where C is completely absorbable and D is not completely absorbable, let $\mathcal{T}_1$ denote the TBox with $\mathcal{T}_1 = \mathcal{T} \cup \{\top \sqsubseteq C \sqcup D\}$ and $\mathcal{T}_2$ denote the TBox with $\mathcal{T}_2 = \mathcal{T} \cup \{A \sqsubseteq D\} \cup X$, where X are the axioms created by $A \leftarrow$* absorbJoined*($\{C\}$). Then, a concept C' is satisfiable with respect to $\mathcal{T}_1$ iff it is satisfiable with respect to $\mathcal{T}_2$.*

Proof If direction: For $\mathcal{I}_2$ an interpretation with $C'^{\mathcal{I}_2} \neq \emptyset$ and $\mathcal{I}_2 \models \mathcal{T}_2$, we show that $\mathcal{I}_2 \models \mathcal{T}_1$. Because of the axiom $A \sqsubseteq D \in \mathcal{T}_2$ it holds for each $\delta \in \Delta^{\mathcal{I}_2}$ that either $\delta \notin A^{\mathcal{I}_2}$ (and thus $\delta \in C^{\mathcal{I}_2}$ by Lemma 1) or $\delta \in D^{\mathcal{I}_2}$. Thus, the axiom $\top \sqsubseteq (C \sqcup D) \in \mathcal{T}_1$ is satisfied for every $\delta \in \Delta^{\mathcal{I}_2}$ and, therefore, $\mathcal{I}_2 \models \mathcal{T}_1$.

Only if direction: For $\mathcal{I}_1$ an interpretation with $C'^{\mathcal{I}_1} \neq \emptyset$ and $\mathcal{I}_1 \models \mathcal{T}_1$, we construct an interpretation $\mathcal{I}'_1$ with $C'^{\mathcal{I}'_1} \neq \emptyset$ and $\mathcal{I}'_1 \models \mathcal{T}_2$. Since $\mathcal{I}_1 \models \mathcal{T}_1$ and $\mathcal{T}_1$ is an extension of $\mathcal{T}$, it follows that $\mathcal{I}_1 \models \mathcal{T}$. Because of Lemma 1, there exists an interpretation $\mathcal{I}'_1$ that can be constructed from $\mathcal{I}_1$ for which it holds that $\mathcal{I}'_1 \models \mathcal{T} \cup X$ and for all $\delta \in \Delta^{\mathcal{I}'_1}$ that $\delta \in A^{\mathcal{I}'_1}$ only if $\delta \notin C^{\mathcal{I}'_1}$. Thus, it also follows that $\mathcal{I}'_1 \models A \sqsubseteq D$, because $\Delta^{\mathcal{I}'_1} = \Delta^{\mathcal{I}_1}$ and for all $\delta \in \Delta^{\mathcal{I}'_1}$ it holds that either $\delta \in C^{\mathcal{I}'_1}$ and thus $\delta \notin A^{\mathcal{I}'_1}$ or $\delta \in D^{\mathcal{I}'_1}$. Thus, if $C'^{\mathcal{I}_1} \neq \emptyset$, then $C'^{\mathcal{I}'_1} \neq \emptyset$. □

In order to show the correctness of the partial absorption of a disjunction $C \sqcup D$, where C is partially absorbable and D is neither completely nor partially absorbable, we reduce the problem to the complete absorption of $C' \sqcup C \sqcup D$, where for C' it holds that C' is completely absorbable and $C' \sqsubseteq C$. We show that the partial absorption of C is equivalent to the complete absorption of the concept C'. Therefore, the partial absorption of $C \sqcup D$ corresponds to the complete absorption of $C' \sqcup C \sqcup D$, which is obviously equisatisfiable to $C \sqcup D$ since C subsumes C'.

Lemma 3 *Let C be a partially absorbable concept, then* absorbJoined*($\{C\}$) generates the absorption of a concept C' for which it holds that $C' \sqsubseteq C$ and C' is completely absorbable.*

Proof If C is already completely absorbable, then the lemma trivially holds since in this case C' is C. Thus, we show in the following for all cases where C is partially absorbable but not completely absorbable that absorbJoined($\{C\}$) generates the absorption of a more specific concept C' for which it holds that $C' \sqsubseteq C$ and C' is completely absorbable.

- If C is of the form $\forall r.D'$ (of the form $\leqslant n\, r.D'$ with $n \geq 0$) and D' ($\mathsf{nnf}(\neg D')$) is neither completely nor partially absorbable, then $\mathsf{absorbConcept}(C)$ creates $\top \sqsubseteq \forall\, \mathsf{inv}(r).A$, which corresponds to the complete absorption of $\forall r.\neg\top$ for which it holds that $\forall r.\neg\top \sqsubseteq \forall r.D'$ ($\forall r.\neg\top \sqsubseteq\, \leqslant n\, r.D'$ for $n \geq 0$).

To prove the complex cases by induction, we assume that the concepts $D_1, \ldots, D_m$ are partially absorbable and the lemma holds for $D_1, \ldots, D_m$, i.e., the absorption completely absorbs the concepts $D'_1, \ldots, D'_m$, for which it holds that $D'_1 \sqsubseteq D_1, \ldots, D'_m \sqsubseteq D_m$, and let $A_1, \ldots, A_m$ be the atomic concepts that are achieved for absorbing $D'_1, \ldots, D'_m$.

- If C is of the form $\forall r.D_1$ (of the form $\leqslant n\, r.D'$ with $n \geq 0$ and $D_1 = \mathsf{nnf}(\neg D')$) and D_1 is partially absorbable, then $\mathsf{absorbConcept}(C)$ creates $A_1 \sqsubseteq \forall\, \mathsf{inv}(r).A$, where A_1 is the atomic concept that is returned by $\mathsf{absorbConcept}(D_1)$ for completely absorbing D'_1. The absorption of C corresponds to the complete absorption of $\forall r.D'_1$ and, by the induction hypothesis, we have $D'_1 \sqsubseteq D_1$. Thus, it also holds that $\forall r.D'_1 \sqsubseteq \forall r.D_1$ ($\forall r.D'_1 \sqsubseteq\, \leqslant n\, r.D'$ for $n \geq 0$, because $D'_1 \sqsubseteq \neg D'$).
- If C is of the form $D_1 \sqcup \ldots \sqcup D_m \sqcup C_1 \sqcup \ldots \sqcup C_n$ with $D_1, \ldots, D_m$ partially absorbable and $C_1, \ldots, C_m$ neither partially nor completely absorbable, then the absorption creates the binary axiom chain $(A_1 \sqcap A_2) \sqsubseteq T_1, (T_1 \sqcap A_3) \sqsubseteq T_2, \ldots, (T_{m-2} \sqcap A_m) \sqsubseteq A$, which corresponds to the complete absorption of $D'_1 \sqcup \ldots \sqcup D'_m$, where $A_1, \ldots, A_m$ are again the atomic concepts for absorbing $D'_1, \ldots, D'_m$. Because of the induction hypothesis it holds that $D'_1 \sqcup \ldots \sqcup D'_m \sqsubseteq D_1 \sqcup \ldots \sqcup D_m \sqcup C_1 \sqcup \ldots \sqcup C_n$.
- If C is of the form $D_1 \sqcap D_2$ with D_1, D_2 partially absorbable, then the absorption creates the axioms $A_1 \sqsubseteq A$ and $A_2 \sqsubseteq A$, which corresponds to the complete absorption of $D'_1 \sqcap D'_2$, where A_1 and A_2 are the atomic concepts for absorbing D'_1 and D'_2. Because of the induction hypothesis it holds that $D'_1 \sqcap D'_2 \sqsubseteq D_1 \sqcap D_2$.
- If C is of the form $\neg A$, A is completely defined by the axioms $A \equiv \hat{D}_1 \in \mathcal{T}, \ldots, A \equiv \hat{D}_m \in \mathcal{T}$ and $D_1 = \neg\hat{D}_1, \ldots, D_m = \neg\hat{D}_m$ are partially absorbable, then the absorption returns A^+ and also creates the axioms that imply the candidate concept A^+. Let A' be the atomic concept that is completely defined by the axioms $A' \equiv \neg D'_1, \ldots, A' \equiv \neg D'_m$, then the complete absorption of $D'_1, \ldots, D'_m$ creates the atomic concepts $A_1, \ldots, A_m$ and additionally the axioms that imply A'^+. The partial absorption of $\neg A$ corresponds to complete absorption of $\neg A'$ and thus it obviously holds by the induction hypothesis that $\neg A' \sqsubseteq \neg A$ and $A'^+ \sqsubseteq A^+$. □

As a consequence of the above lemmas, we find that the above presented absorption algorithms indeed produce a TBox for which concept satisfiability is preserved, which obviously also holds for knowledge base consistency:

Theorem 2 *Let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ denote a knowledge base, and $\mathcal{K}' = \mathcal{T}' \cup \mathcal{A}$ the knowledge base where $\mathcal{T}$ is absorbed into $\mathcal{T}'$ by the* $\mathsf{absorbTBox}$ *function, i.e., $\mathcal{T}' = \mathsf{absorbTBox}(\mathcal{T})$, then $\mathcal{K}$ is consistent iff $\mathcal{K}'$ is consistent.*

Clearly, the presented tableau algorithm in Section 2 can also be used to show the consistency of an absorbed knowledge base, i.e., it builds a clash-free and fully expanded completion graph for $\mathcal{K}$ iff a class-free and fully expanded completion graph exists for $\mathcal{K}'$.

4 Nominal Schema Absorption

Axioms with nominal schemas are very expressive in comparison to many decidable alternatives based on rules. For instance, the atoms in the heads or bodies of DL-safe SWRL rules can only be instantiated with individuals that occur in the ABox. In tableau algorithms, it is, therefore, only necessary to check whether the bodies of such rules are satisfied on tableau nodes that represent ABox individuals/nominals. If this is the case, then the atoms of the heads have to be added, however, also exclusively to individual/nominal nodes. This is no longer the case for axioms with nominal schemas. For example, given the nominal schema axiom

$$\exists t.\exists t.\exists t.(\exists r.\{x\} \sqcap \exists s.\{x\}) \sqsubseteq \exists t.\{x\},$$

we have to check whether the left-hand side is satisfied at any tableau node, although the variable x can only bind to nodes that represent individuals/nominals. That is, checking may also involve blockable nodes in the completion graph that do not represent ABox individuals. Furthermore, such axioms can then also enforce the addition of the right-hand side on blockable nodes. As a consequence, typical approaches for rule processing, such as Rete [5], cannot be used in a straightforward way since blocking easily becomes unsound.

Due to the fact that it would be necessary to also process all blockable nodes with the Rete algorithm, i.e., also the concepts in the label of blockable nodes as well as the roles in the edge labels to these blockable nodes have to be used as input facts for the Rete algorithm, and because Rete does not provide blocking information, it is not clear when the expansion of new successors can be stopped in the tableau algorithm. For example, if the knowledge base also contains the axiom $\top \sqsubseteq \exists t.\top \sqcap \exists r.\{a\} \sqcap \exists s.\{a\}$ and the construction of new successors is already blocked after, e.g., two successively created t-successors, then the Rete algorithm cannot infer the right-hand side of the considered nominal schema axiom for any constructed node since this requires the successive creation of at least three t-successors. Obviously, this is even more complicated if some of the roles are complex.

Our approach to overcome this issue is to emulate well known rule processing algorithms such as Rete by adapted tableau rules, which propagate bindings of variables for concepts through the completion graph. The propagated bindings of variables can be considered in the blocking condition, which allows for ensuring completeness, soundness and termination. As a nice side-effect, the propagation of bindings in the completion graph also means that complex roles can be supported without further adjustments.

This approach works well if the axioms have a typical rule structure, i.e., the axioms have a large absorbable part and almost every nominal schema variable appears at least once in the absorbable part. This is hardly surprising, because ordinary GCIs without nominal schema variables must also have a large absorbable part for reasoning systems to handle such axioms efficiently.

In order to actually bind variables to individuals (or nodes in a completion graph), we use the $\downarrow$ binder operator, as known from Hybrid Logics [3]. The unrestricted extension of a Description Logic with binders easily leads to undecidability of the standard reasoning problems. However, we retain the decidability since we only bind variables to individuals that occur in the ABox. In order to realise this, we extend a knowledge base with nominal schemas with axioms of the form $\{a\} \sqsubseteq O$ for each individual a, where O is a fresh atomic concept, and the axioms created by the absorption ensure that binders are then only triggered in the completion graph if the special concept O occurs in the label of the node. In the remainder of this paper, we assume that all considered knowledge bases already contain the $\{a\} \sqsubseteq O$ axioms for each ABox individual a.

Algorithm 6 Absorption extensions for Algorithms 1

```
1: procedure isCA($C$)
       ...
2:     if $C = \neg\{x\}$ then
3:         return true
4:     end if
       ...
5: end procedure
```

```
1: procedure isPA($C$)
       ...
2:     if $C = \neg\{x\}$ then
3:         return true
4:     end if
       ...
5: end procedure
```

Algorithm 7 Absorption extensions for Algorithm 4

```
1: procedure absorbConcept($C$)
       ...
2:     if $C = \neg\{x\}$ then
3:         $T_x \leftarrow$ fresh atomic concept
4:         $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{O \sqsubseteq \downarrow x.T_x\}$
5:         return $T_x$
6:     end if
       ...
7: end procedure
```

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 without nominal schema variables. Typically, the absorption algorithm can be directly extended 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 for the nominal schema absorption that all conjunctions in absorbable positions are eliminated. This can be done by duplicating the disjunction that is absorbed and replacing the corresponding conjunction in one case with C_1 and in the other case 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 the conjunction $\neg\{x\} \sqcap \neg A$ has to be eliminated by replacing 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 (cf. Algorithms 6 and 7):

- isCA(C) (isPA(C)) is extended to return that a negated occurrence of a nominal schema $\neg\{x\}$ is completely (partially) absorbable.
- absorbConcept(C) must now also handle a negated occurrence of a nominal schema $\neg\{x\}$ by absorbing it to $O \sqsubseteq \downarrow x.T_x$, where T_x is a fresh atomic concept and O is the special atomic concept that is added to the label of every individual $\{a\}$ in the ABox by axioms of the form $\{a\} \sqsubseteq O$.

Other concepts can be absorbed as before, however, the final atomic concept A created by the absorption cannot initiate the addition of the remaining, non-absorbed part of the axiom in the same way. If the remaining disjuncts $D_1, \ldots, D_n$ still contain nominal schemas, then the disjunction has to be grounded with those bindings of variables that have been propagated to 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 schema variables, then $A \sqsubseteq gr(D_1 \sqcup \ldots \sqcup D_n)$ has to be added to the TBox, where $gr(C)$ 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 schema variables.

Example 3 As an example, the axiom $\exists r.(\{x\} \sqcap \exists a.\{y\} \sqcap \exists v.\{z\}) \sqcap \exists s.(\exists a.\{y\} \sqcap \exists v.\{z\}) \sqsubseteq \exists c.\{x\}$ can be almost completely absorbed into the following axioms:

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

Again, 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. In the example, we have reused axioms for the absorption of the same concepts to reduce the total number of axioms. The basic algorithm of Section 3 would generate for each occurrence of $\neg\{y\}$ a separate binder concept, i.e., we would have $O \sqsubseteq \downarrow y.T_y$ as well as $O \sqsubseteq \downarrow y.T'_y$, which is obviously not necessary.

4.2 Tableau Algorithm Extensions to Handle Variable Bindings

We can now extend a standard tableau decision procedure to support (absorbed) nominal schema axioms. The $\downarrow$ binders and $gr(\cdot)$ concepts are handled by new rules. Furthermore, the $\sqsubseteq_1$- and $\sqsubseteq_2$-rules to handle TBox axioms and the $\forall$-rule (for transitivity support also the $\forall^+$-rule) have to be adapted in order to propagate variables bindings.

Roughly speaking, for each concept C in the label of a node v, we keep a set of mappings that records bindings for variables. A mapping set is created, when a concept of the form $\downarrow x.C$ occurs in the label of a node v. In this case, we add C to the label of v and, in order to "remember" the binding $x \mapsto v$, we add the mapping μ with $\mu(x) = v$ to the mappings of C. Note that, as a consequence of our absorption algorithm, a binder concept $\downarrow x.C$ is always such that C does not contain further binders.

Definition 9 (Variable Mapping) A *variable mapping* μ is a (partial) function from variable names to individual names. For a variable mapping μ – and more generally for any (partial) function – the set of elements on which μ is defined is the *domain*, written $\mathsf{dom}(\mu)$, of μ, and the set $\mathsf{ran}(\mu) = \{\mu(x) \mid x \in \mathsf{dom}(\mu)\}$ is the *range* of μ.

We use ϵ for the *empty variable mapping*, i.e., $\mathsf{dom}(\epsilon) = \emptyset$, and we associate a concept fact $C(v)$ with a set of variable mappings, denoted by $\mathcal{B}(C, v)$. Given a (possibly empty) set of variable mappings M, let $M^\epsilon = \{\epsilon\}$ if $M = \emptyset$ and $M^\epsilon = M$ otherwise.

If no confusing is likely to arise, we simply write mapping instead of variable mapping. The mappings for a concept fact 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 $\sqsubseteq_1$-rule (cf. Table 3) to an axiom of the form $A \sqsubseteq C$, we keep the mappings also for the concept C. Similarly, we extend other rules (see Table 3) and we describe the not so straightforward extensions in more detail below. Note, 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 the gr-concept, then the remaining, non-absorbed part of the axiom is grounded and thus corresponds to an ordinary concept.

Some major adjustments are necessary in order to handle binary absorption axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$ correctly (cf. $\sqsubseteq_2$-rule). 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

Table 3 Tableau rule extensions to propagate variable 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 $\mathcal{B}(\forall r.C, v) \nsubseteq \mathcal{B}(C, w)$
	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 $\mathcal{B}(A, v) \nsubseteq \mathcal{B}(C, v)$
	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 $(\mathcal{B}(A_1, v) \bowtie^{\epsilon} \mathcal{B}(A_2, v)) \nsubseteq \mathcal{B}(C, v)$
	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	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C_{[\mu]}\}$

$\mathcal{B}(A_1, v) \neq \emptyset$ 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 10 (Variable Mapping Join) Two variable mappings μ_1 and μ_2 are *compatible* if $\mu_1(x) = \mu_2(x)$ for all $x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$. A variable mapping $\mu_1 \cup \mu_2$ is defined by setting $(\mu_1 \cup \mu_2)(x) = \mu_1(x)$ if $x \in \mathsf{dom}(\mu_1)$, and $(\mu_1 \cup \mu_2)(x) = \mu_2(x)$ otherwise. 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 $\mu_2\} \setminus \{\epsilon\}$.

Note, the extension by the empty variable mapping ϵ is required to propagate variable mappings to a concept C if $\mathcal{B}(A_1, v) = \emptyset$ or $\mathcal{B}(A_2, v) = \emptyset$ for a binary absorption axiom of the form $(A_1 \sqcap A_2) \sqsubseteq C$. For instance, if $\mathcal{B}(A_1, v) = \{\{x \mapsto a\}\}$ and $\mathcal{B}(A_2, v) = \emptyset$, then the variable mapping $\{x \mapsto a\}$ has to be propagated to $C(v)$. This is realised by the $\sqsubseteq_2$-rule since $\{\{x \mapsto a\}\}$ is joined with the singleton variable mapping set consisting only of the empty variable mapping ϵ instead of $\emptyset$, which results in $\{\{x \mapsto a\}\}$. In contrast, we cannot simply associate all concept facts also with the empty variable mapping, because all mappings are compatible with the empty variable mapping. Hence, if $\mathcal{B}(A_1, v) \neq \emptyset$ and $\mathcal{B}(A_2, v) \neq \emptyset$, also the variable mappings of $\mathcal{B}(A_1, v)$ and $\mathcal{B}(A_2, v)$ would directly be propagated to C, whereas only the combination of the mappings should be propagated. For example, if for $\mathcal{B}(A_1, v) = \{\{x \mapsto a\}\}$ and $\mathcal{B}(A_2, v) = \{\{x \mapsto b\}\}$ also the empty variable mapping was associated to the concept facts $A_1(v)$ and $A_2(v)$, i.e., $\mathcal{B}(A_1, v) = \{\{x \mapsto a\}, \emptyset\}$ and $\mathcal{B}(A_2, v) = \{\{x \mapsto b\}, \emptyset\}$, then the join of $\mathcal{B}(A_1, v)$ and $\mathcal{B}(A_2, v)$ would propagate $\{x \mapsto a\}$ and $\{x \mapsto a\}$ to C although these variable mappings are not compatible. Clearly, it would be possible to only associate all concept facts in the initial completion graph with the empty variable mapping. One could then propagate these empty variable mappings to newly added concepts as long as no other variable mappings are created for these concepts through the binder rule ($\downarrow$-rule). However, it would be necessary to adapt all other tableau rules as well, whereas the dynamic extension with ϵ during a join allows for not modifying other tableau rules.

Besides the new $\downarrow$-rule, we also have to handle a grounding concept $gr(C)$ in the label of a node v with the tableau algorithm. Therefore, the gr-rule grounds the concept C based on the variable mappings that are associated to $gr(C)$ on the node v.

Definition 11 (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 $M = \emptyset$, we define the *completion* $\mathsf{comp}_Y^{\mathcal{K}}(M)$ *of* M *w.r.t. the variable set* Y *and a knowledge base* $\mathcal{K}$ containing the individuals $\mathsf{Inds}(\mathcal{K})$ as

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

In order to ground the concept C for a concept fact $gr(C)(v)$, the gr-rule uses the variable mappings of $\mathsf{comp}_{\mathsf{Vars}(C)}^{\mathcal{K}}(\mathcal{B}(gr(C), v))$. Since the mappings that are propagated to $\mathcal{B}(gr(C), v)$ might not contain 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 is realised by the completion function $\mathsf{comp}_{\mathsf{Vars}(C)}^{\mathcal{K}}(\mathcal{B}(gr(C), v))$, where each variable mapping $\mu \in \mathcal{B}(gr(C), v)$ is extended to all possible variable mappings such that also the variables that occur in C but not in $\mathcal{B}(gr(C))$, i.e., $\mathsf{Vars}(C) \setminus \mathsf{dom}(\mu)$, are bound to individuals occurring in $\mathcal{K}$. As a consequence of this completion, all concepts obtained by the grounding of C are fully grounded and can now be added and handled as ordinary concepts in the completion graph. Therefore, it is also not necessary to further propagate variable mappings to the grounded concepts.

In order to support the more expressive Description Logic $\mathcal{SROIQ}$ with our absorption technique of nominal schemas, it would be necessary to further extend the $\forall$-rule to complex roles. This extension can easily be achieved by also propagating the variable mappings over those $\forall$-concepts that are introduced to handle the automata of the role inclusion axioms [7]. Alternatively we could adapt the technique to eliminate role chains (incl. transitivity) [4]. The remaining $\mathcal{SROIQ}$ features are straightforward to support.

Standard pairwise blocking is extended by the new condition in the definition below to ensure that the expansion of the completion graph is not stopped too early, even if variable mappings are propagated through the completion graph.

Definition 12 (Blocking with Variable Mappings) A node v with predecessor v' is *directly blocked* if there exists an ancestor node w of v with predecessor w' such that

- v is directly pairwise blocked by w (see conditions 1 - 4 of Definition 8), and
- $\mathcal{B}(C, v) = \mathcal{B}(C, w)$ and $\mathcal{B}(D, v') = \mathcal{B}(D, w')$ for all $C \in \mathcal{L}(v)$ and $D \in \mathcal{L}(v')$.

Example 3 (continued) The completion graph in Figure 2 is obtained in the course of testing the consistency of a knowledge base containing the axioms of Example 3 and the following assertions:

$$r(a_0, a_1) \quad s(a_0, a_2) \quad a(a_1, a_3) \quad v(a_1, a_4) \quad a(a_2, a_3) \quad v(a_2, a_4).$$

The set of variable mappings that is associated to a concept fact is shown in the superscript of the concept in the label of the corresponding node. Note, we have highlighted those concepts and variable mappings that are responsible for the grounding of new concepts in this example. However, since O and thereby also the binder concepts are added to all

$$\mathcal{L}(a_0) = \left\{ \begin{array}{l} \top, \{a_0\}, O, \downarrow x.T_x, \downarrow y.T_y, \downarrow z.T_z, T_x^{\{\{x\mapsto a_0\}\}}, T_y^{\{\{y\mapsto a_0\}\}}, T_z^{\{\{z\mapsto a_0\}\}}, (\forall a^-.T_1)^{\{\{y\mapsto a_0\}\}}, (\forall v^-.T_2)^{\{\{z\mapsto a_0\}\}}, \\ T_4^{\{\{y\mapsto a_3, z\mapsto a_4\}\}}, T_6^{\{\{x\mapsto a_1, y\mapsto a_3, z\mapsto a_4\}\}}, T_7^{\{\{x\mapsto a_1, y\mapsto a_3, z\mapsto a_4\}\}}, gr(\exists c.\{x\})^{\{\{x\mapsto a_1, y\mapsto a_3, z\mapsto a_4\}\}}, \exists c.\{a_1\} \end{array} \right\}$$

$$\mathcal{L}(a_1) = \left\{ \begin{array}{c} \top, \{a_1\}, O, \downarrow x.T_x, \downarrow y.T_y, \downarrow z.T_z, T_x^{\{\{x\mapsto a_1\}\}}, \\ T_y^{\{\{y\mapsto a_1\}\}}, T_z^{\{\{z\mapsto a_1\}\}}, (\forall a^-.T_1)^{\{\{y\mapsto a_1\}\}}, \\ (\forall v^-.T_2)^{\{\{z\mapsto a_1\}\}}, T_1^{\{\{y\mapsto a_3\}\}}, T_2^{\{\{z\mapsto a_4\}\}}, \\ T_3^{\{\{y\mapsto a_3, z\mapsto a_4\}\}}, (\forall s^-.T_4)^{\{\{y\mapsto a_3, z\mapsto a_4\}\}}, \\ T_5^{\{\{x\mapsto a_1, y\mapsto a_3, z\mapsto a_4\}\}}, (\forall r^-.T_6)^{\{\{x\mapsto a_1, y\mapsto a_3, z\mapsto a_4\}\}} \end{array} \right\}$$

$$\mathcal{L}(a_2) = \left\{ \begin{array}{c} \top, \{a_2\}, O, \downarrow x.T_x, \downarrow y.T_y, \downarrow z.T_z, T_x^{\{\{x\mapsto a_2\}\}}, \\ T_y^{\{\{y\mapsto a_2\}\}}, T_z^{\{\{z\mapsto a_2\}\}}, (\forall a^-.T_1)^{\{\{y\mapsto a_2\}\}}, \\ (\forall v^-.T_2)^{\{\{z\mapsto a_2\}\}}, T_1^{\{\{y\mapsto a_3\}\}}, T_2^{\{\{z\mapsto a_4\}\}}, \\ T_3^{\{\{y\mapsto a_3, z\mapsto a_4\}\}}, (\forall s^-.T_4)^{\{\{y\mapsto a_3, z\mapsto a_4\}\}}, \\ T_5^{\{\{x\mapsto a_2, y\mapsto a_3, z\mapsto a_4\}\}}, (\forall r^-.T_6)^{\{\{x\mapsto a_2, y\mapsto a_3, z\mapsto a_4\}\}} \end{array} \right\}$$

$$\begin{tikzcd} & a_0 \arrow[dl, "{r, c}"'] \arrow[dr, "s"] & \\ a_1 \arrow[d, "a"'] \arrow[drr, "v"] & & a_2 \arrow[d, "v"] \arrow[dll, "a"'] \\ a_3 & & a_4 \end{tikzcd}$$

$$\mathcal{L}(a_3) = \left\{ \begin{array}{c} \top, \{a_3\}, O, \downarrow x.T_x, \downarrow y.T_y, \downarrow z.T_z, \\ T_x^{\{\{x\mapsto a_3\}\}}, T_y^{\{\{y\mapsto a_3\}\}}, T_z^{\{\{z\mapsto a_3\}\}}, \\ (\forall a^-.T_1)^{\{\{y\mapsto a_3\}\}}, (\forall v^-.T_2)^{\{\{z\mapsto a_3\}\}} \end{array} \right\}$$

$$\mathcal{L}(a_4) = \left\{ \begin{array}{c} \top, \{a_4\}, O, \downarrow x.T_x, \downarrow y.T_y, \downarrow z.T_z, \\ T_x^{\{\{x\mapsto a_4\}\}}, T_y^{\{\{y\mapsto a_4\}\}}, T_z^{\{\{z\mapsto a_4\}\}}, \\ (\forall a^-.T_1)^{\{\{y\mapsto a_4\}\}}, (\forall v^-.T_2)^{\{\{z\mapsto a_4\}\}} \end{array} \right\}$$

Fig. 2 Generated completion graph with propagated variable mappings for Example 3

ABox individuals, they automatically create variable mappings for every ABox individual. Obviously, many of these mappings are not necessary and their creation can easily be limited by additional axioms. For example, the variable x only has to be bound if an a-neighbour and also a v-neighbour exists for an individual node, i.e., the binding of x can be delayed with $\exists v.\top \sqsubseteq T'_1$, $\exists a.\top \sqsubseteq T'_2$, $(T'_1 \sqcap T'_2) \sqsubseteq T'_3$ and $(O \sqcap T'_3) \sqsubseteq \downarrow x.T_x$. Similar delays for the creation and propagation of variable mappings can be realised systematically by extending the presented absorption algorithm with a backward chaining technique, which we present in Section 5 in more detail.

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 $gr(\exists c.\{x\})$ 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.

4.3 Correctness

In the following we prove the correctness of our nominal schema absorption technique. For this, we roughly proceed as follows: Given a nominal schema axiom $C \sqsubseteq D$ and an absorbed TBox $\mathcal{T}$, then for $\mathcal{T}_{ns}$ and $\mathcal{T}_{ug}$ as the TBoxes obtained from absorbing $\mathcal{T} \cup \{C \sqsubseteq D\}$ and $\mathcal{T} \cup \{U_1, \ldots, U_h\}$, respectively, where $U_1, \ldots, U_h$ are the upfront grounded axioms of $C \sqsubseteq D$, we show that a fully expanded and clash-free completion graph G_{ns} for $\mathcal{T}_{ns}$ can be converted to a fully expanded and clash-free completion graph G_{ug} for $\mathcal{T}_{ug}$. Furthermore, we show that our extended tableau algorithm constructs a complete and clash-free completion graph G_{ns} for $\mathcal{T}_{ns}$ if there exists a fully expanded and clash-free completion graph G_{ug} for $\mathcal{T}_{ug}$ that is constructed by a standard tableau algorithm.

Please note that we only work with TBoxes instead of knowledge bases. This assumption is w.l.o.g. since in the presence of nominals ABoxes can be internalised (e.g., $C(a)$ is equivalent to the GCI $\{a\} \sqsubseteq A$, $r(a_1, a_2)$ to $\{a\} \sqsubseteq \exists r.\{b\}$, etc.). We assume, therefore, that a completion $\mathsf{comp}_Y^{\mathcal{T}}(M)$ is analogously defined to the completion $\mathsf{comp}_Y^{\mathcal{K}}(M)$ with $\mathcal{K} = (\mathcal{T}, \emptyset)$.

To simplify the conversion between a completion graph for $\mathcal{T}_{ns}$ and a standard completion graph for $\mathcal{T}_{ug}$, we ensure that all concept facts can directly be converted into concept facts for the other completion graph. Therefore, we make the following simplifying assumptions: We assume that the absorption of nominals of the form $\neg\{a\}$ generates $\{a\} \sqsubseteq \top \sqcap T$ instead of $\{a\} \sqsubseteq T$ (cf. Algorithm 4, line 19), which is obviously logically equivalent. As a result, binder concepts such as $\downarrow x.T$ can be directly converted to concepts of the form $\top \sqcap T$. We also assume that the absorption of the upfront grounded axiom $C_{[\mu]} \sqsubseteq D_{[\mu]}$, by the variable mapping μ, creates a new special grounding concept $gr_\mu(D)$ to add the remaining, non-absorbable part of the axiom instead of directly implying $D_{[\mu]}$. This new concept construct retains the mapping μ and corresponds to the grounding concept $gr(D)$ that is created for the absorption of the nominal schema axiom $C \sqsubseteq D$.

Before introducing the actual conversion, we first define the notion of concept and axiom set closure in the context of absorption:

Definition 13 (Absorption Closure) Let $\mathsf{fclos}(C)$ and $\mathsf{fclos}(Z)$ be the full closure of a concept C and the set of axioms Z, respectively, as in Definition 4. For a TBox $\mathcal{T}$ and an axiom $C' \sqsubseteq D'$ with $\mathsf{nnf}(\neg C')$ completely and D' not completely absorbable, the *absorption closure* $\mathsf{aclos}_{\mathcal{T}}(C' \sqsubseteq D')$ *for* $\mathcal{T}$ *and* $C' \sqsubseteq D'$ contains the new concepts introduced by the absorption of $C' \sqsubseteq D'$ and is defined as:

$$\mathsf{aclos}_{\mathcal{T}}(C' \sqsubseteq D') := \mathsf{fclos}(X'_1, \ldots, X'_n) \setminus (\mathsf{fclos}(\mathcal{T}) \cup \mathsf{fclos}(D')),$$

where $X'_1, \ldots, X'_n$ are the axiom introduced by the absorption of $C' \sqsubseteq D'$.

Note that the concepts in the absorption closure are those that are relevant for the conversion between completion graphs since these are the concepts with variable mappings.

Now, the actual conversion of concepts and axioms obtained from the absorption is defined as follows:

Definition 14 (Conversion) Let $C \sqsubseteq D$ be a nominal schema axiom where $\mathsf{nnf}(\neg C)$ is completely and D not completely absorbable, and let μ be a mapping with $\mathsf{dom}(\mu) = \mathsf{Vars}(\neg C \sqcup D)$. Furthermore, let $\mathcal{T}$ be an absorbed TBox, $\mathcal{T}_{ns}$ and $\mathcal{T}_{ug}$ TBoxes obtained by absorbing $\mathcal{T} \cup \{C \sqsubseteq D\}$ and $\mathcal{T} \cup \{U_1, \ldots, U_h\}$, respectively, where $U_1, \ldots, U_h$ are the axioms obtained by the upfront grounding of $C \sqsubseteq D$. We denote the axioms (in creation order) and fresh atomic concepts obtained by absorbing $\mathsf{nnf}(\neg C \sqcup D)$ with $X_1, \ldots, X_n$ and $T_1, \ldots, T_g$, respectively. Similarly, we use $X_1^\mu, \ldots, X_n^\mu$ and $T_1^\mu, \ldots, T_g^\mu$ for the case of absorbing $\mathsf{nnf}((\neg C \sqcup D)_{[\mu]})$.

For the concept C', we inductively define the *concept conversion* $\mathsf{conv}_\mu(C')$ *of* C' *w.r.t.* $\mathcal{T}$*,* $C \sqsubseteq D$ *and* μ as

$$\mathsf{conv}_\mu(C') := \begin{cases} C' & \text{if } C' \notin \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D) \\ (\top \sqcap \mathsf{conv}_\mu(C'')) & \text{if } C' = \downarrow x.C'' \\ gr_\mu(D) & \text{if } C' = gr(D) \\ C'_{[T_1/T_1^\mu, \ldots, T_g/T_g^\mu]} & \text{otherwise,} \end{cases}$$

where $C'_{[T_1/T_1^\mu, \ldots, T_g/T_g^\mu]}$ denotes the syntactic replacement of each occurrence of T_i in C' with T_i^μ, for $1 \leq i \leq g$. The extension to axioms $\mathsf{xconv}_\mu(X)$ is defined as:

$$\mathsf{xconv}_\mu(X) := \begin{cases} \{\mu(x)\} \sqsubseteq \top \sqcap \mathsf{conv}_\mu(D') & \text{if } X = O \sqsubseteq \downarrow x.D' \\ \mathsf{conv}_\mu(C') \sqsubseteq \mathsf{conv}_\mu(D') & \text{otherwise.} \end{cases}$$

In the remainder of the section, we use $C \sqsubseteq D$, μ, $\mathcal{T}$, $\mathcal{T}_{ns}$, $\mathcal{T}_{ug}$, $T_1, \ldots, T_g$, $T_1^\mu, \ldots, T_g^\mu$, $X_1, \ldots, X_n$, and $X_1^\mu, \ldots, X_n^\mu$ as in the above definition.

Note that the restrictions on $C \sqsubseteq D$ are w.l.o.g. since any nominal schema axiom can be transformed into the desired form in an equivalence preserving manner. If $\mathsf{nnf}(\neg C)$ is only partially absorbable, then a completely absorbable concept $\mathsf{nnf}(\neg C')$ can be extracted from C (cf. Lemma 3), which can be used to obtain an axiom $C' \sqsubseteq D'$, where it holds that $C' \sqsubseteq C$, C' is completely absorbable and $D' = \mathsf{nnf}(\neg C') \sqcup D$ is not completely absorbable. Also note that $\neg\top$ and $\bot$ can always be used to extend a disjunction that corresponds to an axiom in order to obtain a completely absorbable and not completely absorbable disjunct w.r.t. our absorption algorithm.

We can now show that we can convert the axioms obtained by absorbing $\mathsf{nnf}(\neg C \sqcup D)$ from the nominal schema axiom $C \sqsubseteq D$ into the axioms that are obtained by absorbing the grounded version $\mathsf{nnf}((\neg C \sqcup D)_{[\mu]})$, which is the first step in the conversion of a completion graph with nominal schema concepts to a standard completion graph:

Lemma 4 *Let $\mathcal{T}$ be an absorbed TBox, $C \sqsubseteq D$ a nominal schema axiom, $U_1, \ldots, U_h$ the upfront grounding, μ a mapping, $\mathcal{T}_{ns}$ and $\mathcal{T}_{ug}$ TBoxes, and $X_1, \ldots, X_n$ and $X_1^\mu, \ldots, X_n^\mu$ axioms as in Definition 14. The set $\{\mathsf{xconv}_\mu(X_1), \ldots, \mathsf{xconv}_\mu(X_n)\}$ is identical to the set $\{X_1^\mu, \ldots, X_n^\mu\}$.*

Proof Let $T_1, \ldots, T_g$ and $T_1^\mu, \ldots, T_g^\mu$ be the fresh atomic concepts introduced by the absorption of $\mathsf{nnf}(\neg C \sqcup D)$ and $\mathsf{nnf}((\neg C \sqcup D)_{[\mu]})$, respectively. Since the concepts $\mathsf{nnf}(\neg C \sqcup D)$ and $\mathsf{nnf}((\neg C \sqcup D)_{[\mu]})$ only differ in the nominal schemas that are replaced by nominals, the absorption of $\mathsf{nnf}(\neg C \sqcup D)$ and $\mathsf{nnf}((\neg C \sqcup D)_{[\mu]})$ is identical expect for axioms of the form $O \sqsubseteq {\downarrow}x.T_i$ and $T_g \sqsubseteq gr(D)$ in $\mathcal{T}_{ns}$, which correspond to axioms of the form $\{a\} \sqsubseteq (\top \sqcap T_i^\mu)$ and $T_g^\mu \sqsubseteq gr_\mu(D)$ in $\mathcal{T}_{ug}$. Hence, by Definition 14, the claim holds. □

For the conversion, we use the implicitly associated sets of variable mappings, which are defined as follows:

Definition 15 (Implicitly Associated Mappings) The implicitly associated set of variable mappings $\mathsf{mapp}^G(C'(v))$ for a concept fact $C'(v)$ and C' in the absorption closure w.r.t. a completion graph $G = (V, E, \mathcal{L}, \dot{\neq}, \mathcal{B})$ is defined as:

$$\mathsf{mapp}^G(C'(v)) := \begin{cases} \{\{x \mapsto v\}\} & \text{if } C' = {\downarrow}x.D' \\ \mathcal{B}(C', v) & \text{if } \mathcal{B}(C', v) \neq \emptyset \\ \{\epsilon\} & \text{otherwise.} \end{cases}$$

Now, let G_{ns} be a completion graph showing the satisfiability of the TBox $\mathcal{T}_{ns}$. We can replace each concept fact $C'(v)$ with the implicitly associated variable mappings M and $C' \in \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$, by the concept facts $(\mathsf{conv}_{\mu_1}(C'))(v), \ldots, (\mathsf{conv}_{\mu_k}(C'))(v)$, where $\mu_1, \ldots, \mu_k$ are the mappings obtained from the completion $\mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(M)$ of M. As a result, we obtain a fully expanded completion graph G_{ug} that shows the satisfiability of the upfront grounded TBox $\mathcal{T}_{ug}$.

Lemma 5 (Soundness) *Let $\mathcal{T}$ be an absorbed TBox, $C \sqsubseteq D$ a nominal schema axiom, $U_1, \ldots, U_h$ the upfront grounding for $C \sqsubseteq D$, and $\mathcal{T}_{ns}$ and $\mathcal{T}_{ug}$ TBoxes as in Definition 14. If there is a fully expanded and clash-free completion graph for $\mathcal{T}_{ns}$, then there is a fully expanded and clash-free completion graph for $\mathcal{T}_{ug}$.*

Proof Let $G_{ns} = (V_{ns}, E_{ns}, \mathcal{L}_{ns}, \dot{\neq}_{ns}, \mathcal{B}_{ns})$ be a fully expanded and clash-free completion graph for $\mathcal{T}_{ns}$. We convert G_{ns} into a fully expanded and clash-free completion graph G_{ug}

by replacing every concept fact $C'(v)$, $C' \in \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$, $v \in V_{ns}$, with the implicitly associated variable mappings $M = \mathsf{mapp}^{G_{ns}}(C'(v))$, by the concept facts $(\mathsf{conv}_{\mu_1}(C'))(v)$, $\ldots, (\mathsf{conv}_{\mu_k}(C'))(v)$ with $\{\mu_1, \ldots, \mu_k\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(M)$. Furthermore, let $\lambda_1, \ldots, \lambda_\ell$ be all possible variable mappings for the variables $\mathsf{Vars}(\neg C \sqcup D)$ w.r.t. $\mathcal{T}$, i.e., $\{\lambda_1, \ldots, \lambda_\ell\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\epsilon\})$.

In the following we show that none of the standard tableau rules for the concepts and axioms used in the absorption are applicable to G_{ug}. Please note that the extended tableau rules (Table 3) coincide with the standard tableau rules (cf. Table 1) if no variable mappings are associated to the concept facts. Also note that the concept facts and axioms, which are not related to the absorption, are not affected by the conversion. Thus, the corresponding rules are not applicable for these concepts and axioms. Furthermore, since identical node labels are converted in the same way, blocking is not affected, i.e., if a node is blocked before the conversion, then it is also blocked after the conversion.

- We firstly consider the application of the $\forall$-rule, which is not applicable for G_{ug}, because $C' = \forall r.D'(v)$ is converted to $(\mathsf{conv}_{\mu_1}(\forall r.D'))(v), \ldots, (\mathsf{conv}_{\mu_k}(\forall r.D'))(v)$ and for each r-neighbour node w of v the concept fact $D'(w)$ is either also not associated with variable mappings (which is ensured by the absorption algorithm by creating separate axioms with fresh atomic concepts for the absorption of concepts that do not contain nominal schemas) or is at least also associated with the same variable mappings (otherwise the $\forall$-rule would be applicable for G_{ns}) and thus $D'(w)$ is at least also converted to $(\mathsf{conv}_{\mu_1}(D'))(w), \ldots, (\mathsf{conv}_{\mu_k}(D'))(w)$.
- We now consider the application of the $\sqsubseteq_1$-rule. The absorption creates axioms of the form $H \sqsubseteq D'$ with $D' \in \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$ and $H = \{a\}$ or $H = A$. If $D' \neq {\downarrow}x.D''$ (the replacement axioms for $O \sqsubseteq {\downarrow}x.D''$ are considered together with the ${\downarrow}$-concepts), $H \notin \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$ and $H = A$ or $H = \{a\}$, then we would have the axioms $H \sqsubseteq \mathsf{conv}_{\lambda_1}(D'), \ldots,$ $H \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$ in $\mathcal{T}_{ug}$ and the $\sqsubseteq_1$-rule is not applicable, because, for every node v in G_{ns} with the concept fact $H(v)$, $D'(v)$ is also present and $\mathcal{B}_{ns}(D', v) = \emptyset$. Thus, $D'(v)$ is replaced by $(\mathsf{conv}_{\lambda_1}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$. If $A \in \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$, then we would have the axioms $\mathsf{conv}_{\lambda_1}(A) \sqsubseteq \mathsf{conv}_{\lambda_1}(D'), \ldots, \mathsf{conv}_{\lambda_\ell}(A) \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$ and the $\sqsubseteq_1$-rule is not applicable, because for every node v in G_{ns} with the concept fact $A(v)$ and the associated variable mappings $\mu_1, \ldots, \mu_k$, $A(v)$ would be replaced by $(\mathsf{conv}_{\mu_1}(A))(v), \ldots, (\mathsf{conv}_{\mu_k}(A))(v)$, and D' is either also not associated with variable mappings (which is ensured by the absorption algorithm) or is at least also associated with the variable mappings $\mu_1, \ldots, \mu_k$ (otherwise G_{ns} would not be fully expanded), and is at least also replaced by $(\mathsf{conv}_{\mu_1}(D'))(v), \ldots, (\mathsf{conv}_{\mu_k}(D'))(v)$. Thus, the $\sqsubseteq_1$-rule is not applicable for G_{ug}.
- Next, we consider the application of the $\sqsubseteq_2$-rule for an axiom $(A_1 \sqcap A_2) \sqsubseteq D'$. There are three cases:
  1. If $\mathcal{B}_{ns}(A_1, v) = \emptyset$ and $\mathcal{B}_{ns}(A_2, v) = \emptyset$, then $\mathcal{B}_{ns}(D, v) = \emptyset$ and every concept fact $D'(v)$ is replaced by $(\mathsf{conv}_{\lambda_1}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ and thus the rule is not applicable for the axioms $(\mathsf{conv}_{\lambda_1}(A_1) \sqcap \mathsf{conv}_{\lambda_1}(A_2)) \sqsubseteq \mathsf{conv}_{\lambda_1}(D'), \ldots, (\mathsf{conv}_{\lambda_\ell}(A_1) \sqcap \mathsf{conv}_{\lambda_\ell}(A_2)) \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$.
  2. If $\mathcal{B}_{ns}(A_1, v) \neq \emptyset$ ($\mathcal{B}_{ns}(A_2, v) \neq \emptyset$), then the $\sqsubseteq_2$-rule is analogously to the $\sqsubseteq_1$-rule not applicable, because either there is no variable mapping that is associated to $A_2(v)$ ($A_1(v)$) and, as a consequence, there is also no variable mapping associated to $D'(v)$ (which is ensured by the absorption algorithm), or every variable mapping that is associated to $A_2(v)$ ($A_1(v)$) is also associated to $D'(v)$ if A_1 (A_2) is also in the label of

v. Thus, the $\sqsubseteq_2$-rule cannot add a $\mathsf{conv}_{\lambda_j}(D')$ concept to v that is not already present, because the corresponding $\mathsf{conv}_{\lambda_j}(A_1)$ ($\mathsf{conv}_{\lambda_j}(A_2)$) is missing.

3. If $\mathcal{B}_{ns}(A_1, v) \neq \emptyset$ and $\mathcal{B}_{ns}(A_2, v) \neq \emptyset$, the $\sqsubseteq_2$-rule is also not applicable to the completion graph that is obtained by the conversion, because $A_1(v)$ and $A_2(v)$ are replaced by the concept facts $(\mathsf{conv}_{\mu_1}(A_1))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_1))(v)$ and $(\mathsf{conv}_{\mu'_1}(A_2))(v), \ldots, (\mathsf{conv}_{\mu'_{k'}}(A_2))(v)$, respectively, where $\mu_1, \ldots, \mu_k$ and $\mu'_1, \ldots, \mu'_{k'}$ are the completion of the sets of variable mappings $\mathsf{mapp}^{G_{ns}}(A_1(v))$ and $\mathsf{mapp}^{G_{ns}}(A_2(v))$. The $\sqsubseteq_2$-rule is, however, only applicable for an axiom $(\mathsf{conv}_\mu(A_1) \sqcap \mathsf{conv}_\mu(A_2)) \sqsubseteq \mathsf{conv}_\mu(D')$ if $\mathsf{conv}_\mu(A_1)$ as well as $\mathsf{conv}_\mu(A_2)$ is in the same label, but $\mathsf{conv}_\mu(D')$ is not already present, i.e., $\mu \in \{\mu_1, \ldots, \mu_k\}$ and $\mu \in \{\mu'_1, \ldots, \mu'_{k'}\}$, but $\mu \notin \{\mu_1, \ldots, \mu_k\} \bowtie^\epsilon \{\mu'_1, \ldots, \mu'_{k'}\}$, which is a contradiction, because $\{\mu_1, \ldots, \mu_k\} \bowtie^\epsilon \{\mu'_1, \ldots, \mu'_{k'}\}$ is the same as the completion of $\mathcal{B}_{ns}(A_1, v) \bowtie^\epsilon \mathcal{B}_{ns}(A_2, v)$ to all possible variables used in $C \sqsubseteq D$.

- The $\downarrow$-concepts are more complicated. Concept facts of the form $\downarrow x.D'(a)$ are not explicitly associated with variable mappings. However, because of the axiom $O \sqsubseteq \downarrow x.D'$, they only occur in the label of ABox individual nodes. Thus, we can use the implicit information that x will be bound to the ABox individual node a, and we use the completion of the variable mapping $\{x \mapsto a\}$ for $\mu_1, \ldots, \mu_k$. Therefore, we replace $\downarrow x.D'(a)$ with the concept facts $(\top \sqcap \mathsf{conv}_{\mu_1}(D'))(a), \ldots, (\top \sqcap \mathsf{conv}_{\mu_k}(D'))(a)$. It is not hard to see that $(\top \sqcap \mathsf{conv}_{\mu_1}(D')), \ldots, (\top \sqcap \mathsf{conv}_{\mu_k}(D'))$ cannot be unfolded in G_{ug}, because the $\downarrow$-rule ensures that D' is also already present in the label of the node and is associated with the variable mapping $\{x \mapsto a\}$ and, thus, D' is also replaced by $\mathsf{conv}_{\mu_1}(D'), \ldots, \mathsf{conv}_{\mu_k}(D')$. Analogously, for the axioms $\{a\} \sqsubseteq \top \sqcap \mathsf{conv}_{\mu_1}(D'), \ldots, \{a\} \sqsubseteq \top \sqcap \mathsf{conv}_{\mu_k}(D')$ that we have to consider in G_{ug} instead of $O \sqsubseteq \downarrow x.D'$, the rules for these axioms are also not applicable, because the concept $\downarrow x.D'$ in the label of a has been replaced by the concepts $\top \sqcap \mathsf{conv}_{\mu_1}(D'), \ldots, \top \sqcap \mathsf{conv}_{\mu_k}(D')$ and $\downarrow x.D'$ is in the label of a, because it is added to every ABox individual node due to the axiom $O \sqsubseteq \downarrow x.D'$.
- The argumentation for the gr-concepts and the corresponding rules is very similar. As mentioned before, we assume that the grounding concept is always used to add the remaining, non-absorbable part of the axiom. Thus, $gr(D)$ is always in $\mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$, even if $\mathsf{Vars}(D) = \emptyset$. Furthermore, we also use the assumption that the absorption of an upfront grounded axiom, by the variable mapping μ, also uses a special grounding concept $gr_\mu(D)$, which has to be unfolded to $D_{[\mu]}$ and is, therefore, not problematic for the tableau algorithm, because it corresponds to a conjunction with only one conjunct. Thus, a concept fact $gr(D)(v)$ is replaced by $(\mathsf{conv}_{\mu_1}(gr(D)))(v), \ldots, (\mathsf{conv}_{\mu_k}(gr(D)))(v)$, which is the same as $(gr_{\mu_1}(D))(v), \ldots, (gr_{\mu_k}(D))(v)$. Obviously, these replaced grounding concepts cannot be unfolded to $D_{[\mu_1]}, \ldots, D_{[\mu_k]}$, because $D_{[\mu_1]}, \ldots, D_{[\mu_k]}$ are already present due to the application of the gr-rule for $gr(D)(v)$, for which also the completion of the associated set of variable mappings is used for the grounding of D. □

Next, we show that we can steer our extended tableau algorithm to construct a complete and clash-free completion graph G_{ns} for $\mathcal{T}_{ns}$ if there exists a fully expanded and clash-free completion graph G_{ug} for $\mathcal{T}_{ug}$ that is constructed by a standard tableau algorithm.

Lemma 6 (Completeness) *Let $\mathcal{T}$ be an absorbed TBox, $C \sqsubseteq D$ a nominal schema axiom, $U_1, \ldots, U_h$ the upfront grounding for $C \sqsubseteq D$, and $\mathcal{T}_{ns}$ and $\mathcal{T}_{ug}$ TBoxes as in Definition 14. If there is a fully expanded and clash-free completion graph for $\mathcal{T}_{ug}$, then there is a fully expanded and clash-free completion graph for $\mathcal{T}_{ns}$.*

Proof Let G_{ug} be a completion graph for $\mathcal{T}_{ug}$ that is obtained by applying only rules for concepts and axioms of $\mathcal{T}$. Since our extended rules coincide with the standard tableau rules if no variable mappings are associated to concept facts, our extended tableau algorithm can create G_{ns}, which exactly coincides with G_{ug}. We show that the application of a rule in Table 3 to G_{ns} deterministically adds only concept facts and possibly variable mappings, for which the conversion of these facts and variable mappings are also consequences in G_{ug} that are added in the course of applying standard tableau rules to G_{ug}. Thus, G_{ug} can obviously be used for steering the non-deterministic decisions for G_{ns} to construct a fully expanded and clash-free completion graph if G_{ug} is fully expanded and clash-free.

Now, let G_{ns} and G_{ug} be completion graphs for $\mathcal{T}_{ns}$ and $\mathcal{T}_{ug}$, respectively, and G_{ns} and G_{ug} coincide with the inferred facts so far, i.e., the conversion of concept facts and variable mappings from G_{ns} corresponds to the contained concept facts in G_{ug}. To show by induction that each rule application for G_{ns} only adds concept facts and variable mappings, for which the conversion of these facts and variable mappings are also consequences in G_{ug}, let $\lambda_1, \ldots, \lambda_\ell$ be all possible variable mappings, i.e., $\{\lambda_1, \ldots, \lambda_\ell\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\epsilon\})$. Please note, it suffices to consider only the extended rules for concepts and axioms used for absorbing $C \sqsubseteq D$, because only the concepts in $\mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$ can be associated with variable mappings, for which the extended rules differ to standard rules.

- First, we consider the $\forall$-rule for a concept fact $\forall r.D'(v)$, $\forall r.D' \in \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$, which adds the concept fact $D'(w)$ to an r-neighbour w of v in G_{ns} and possibly the variable mapping $\mu \in \mathcal{B}_{ns}(\forall r.D', v)$ to $B_{ns}(D', w)$. If the $\forall$-rule only adds the concept fact $D'(w)$ for cases where $\mathcal{B}(\forall r.D', v) = \emptyset$, then $\mathsf{mapp}^{G_{ns}}(D'(w)) = \{\epsilon\}$ (which is ensured by the absorption algorithm) and we have to show that in the completion graph G_{ug} the concept facts $(\mathsf{conv}_{\lambda_1}(D'))(w), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(w)$ are also added by rule applications. Obviously, this is the case, because the concept fact $\forall r.D'(v)$ corresponds to $(\mathsf{conv}_{\lambda_1}(\forall r.D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(\forall r.D'))(v)$ in the completion graph G_{ug} and by applying the $\forall$-rule for all concept facts $(\mathsf{conv}_{\lambda_j}(\forall r.D'))(v)$, $1 \leq j \leq \ell$, we have the concepts $\mathsf{conv}_{\lambda_1}(D'), \ldots, \mathsf{conv}_{\lambda_\ell}(D')$ in the label of all neighbour nodes. If the $\forall$-rule adds a variable mapping $\mu \in \mathcal{B}_{ns}(\forall r.D', v)$ to $B_{ns}(D', w)$, then we have to show that $(\mathsf{conv}_{\mu_1}(D'))(w), \ldots, (\mathsf{conv}_{\mu_k}(D'))(w)$ with $\{\mu_1, \ldots, \mu_k\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\mu\})$ are added to G_{ug} by rule applications. But this is also the case since $\forall r.D'(v)$ corresponds to $(\mathsf{conv}_{\mu_1}(\forall r.D'))(w), \ldots, (\mathsf{conv}_{\mu_k}(\forall r.D'))(w)$ in G_{ug} and applying the $\forall$-rule for $(\mathsf{conv}_{\mu_1}(\forall r.D'))(w), \ldots, (\mathsf{conv}_{\mu_k}(\forall r.D'))(w)$ adds $(\mathsf{conv}_{\mu_1}(D'))(w), \ldots, (\mathsf{conv}_{\mu_k}(D'))(w)$ to the label of all neighbour nodes.
- Next, we consider the $\sqsubseteq_1$-rule for an axiom $H \sqsubseteq D'$ with $H = A$ or $H = \{a\}$ and $D' \neq {\downarrow}x.D''$ (we consider the addition of the binder concepts together with the $\downarrow$-rule). If $\mathcal{B}_{ns}(H, v) = \emptyset$ and the $\sqsubseteq_1$-rule adds only the concept fact $D'(v)$ to a node, then we have to show that $(\mathsf{conv}_{\lambda_1}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ are also added to G_{ug} by rule applications. Again, this is obviously the case, because for G_{ug} we have the rules $\mathsf{conv}_{\lambda_1}(H) \sqsubseteq \mathsf{conv}_{\lambda_1}(D'), \ldots, \mathsf{conv}_{\lambda_\ell}(H) \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$. If $\mathcal{B}_{ns}(H, v) \neq \emptyset$, then $H = A$ and the $\sqsubseteq_1$-rule adds also a variable mapping μ to $\mathcal{B}_{ns}(D', v)$, whereby we have to show that $(\mathsf{conv}_{\mu_1}(D'))(v), \ldots, (\mathsf{conv}_{\mu_k}(D'))(v)$ with $\{\mu_1, \ldots, \mu_k\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\mu\})$ are added to G_{ug} by rule applications. Again, this is a consequence of the concept facts $(\mathsf{conv}_{\mu_1}(A))(v), \ldots, (\mathsf{conv}_{\mu_k}(A))(v)$ in G_{ug} and the axioms $\mathsf{conv}_{\mu_1}(A) \sqsubseteq \mathsf{conv}_{\mu_1}(D')$, $\ldots, \mathsf{conv}_{\mu_k}(A) \sqsubseteq \mathsf{conv}_{\mu_k}(D')$ that we have to consider for G_{ug}.
- Let us now consider the $\sqsubseteq_2$-rule for an axiom $(A_1 \sqcap A_2) \sqsubseteq D'$. If the $\sqsubseteq_2$-rule only adds the concept fact $D'(v)$, then we have to show that $(\mathsf{conv}_{\lambda_1}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ are also added to G_{ug} by rule applications. However, this is the case, because $A_1(v)$ and $A_2(v)$ corresponds to $(\mathsf{conv}_{\lambda_j}(A_1))(v)$ and $(\mathsf{conv}_{\lambda_j}(A_2))(v)$ in G_{ug}, respectively, and, since we have the axiom $(\mathsf{conv}_{\lambda_j}(A_1) \sqcap \mathsf{conv}_{\lambda_j}(A_2)) \sqsubseteq \mathsf{conv}_{\lambda_j}(D)$ for each $1 \leq j \leq \ell$, it follows that

all $(\mathsf{conv}_{\lambda_1}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ are also added to G_{ug}. If the $\sqsubseteq_2$-rule also adds the variable mapping μ to $\mathcal{B}_{ns}(D', v)$, then we have to show that $(\mathsf{conv}_{\mu_1}(D'))(v), \ldots,$ $(\mathsf{conv}_{\mu_k}(D'))(v)$ with $\{\mu_1, \ldots, \mu_k\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\mu\})$ are also added to the completion graph G_{ug} by rule applications. Let us first assume that $\mathcal{B}_{ns}(A_1, v) = \emptyset$ and $\mathcal{B}_{ns}(A_2, v) \neq \emptyset$. As a consequence, we have in G_{ug} the concept facts $(\mathsf{conv}_{\lambda_1}(A_2))(v)$, $\ldots$, $(\mathsf{conv}_{\lambda_\ell}(A_2))(v)$ and $(\mathsf{conv}_{\mu_1}(A_2))(v)$, $\ldots$, $(\mathsf{conv}_{\mu_k}(A_2))(v)$. As a result of the axioms $(\mathsf{conv}_{\lambda_j}(A_1) \sqcap \mathsf{conv}_{\lambda_j}(A_2)) \sqsubseteq \mathsf{conv}_{\lambda_j}(D)$, for all $1 \leq j \leq \ell$, the concept facts $(\mathsf{conv}_{\mu_1}(D'))(v), \ldots, (\mathsf{conv}_{\mu_k}(D'))(v)$ are also added to G_{ug} by rule applications. Analogously, this holds for the case where $\mathcal{B}_{ns}(A_2, v) = \emptyset$ and $\mathcal{B}_{ns}(A_1, v) \neq \emptyset$. Let us now assume that $\mathcal{B}_{ns}(A_1, v) \neq \emptyset$ as well as $\mathcal{B}_{ns}(A_2, v) \neq \emptyset$. We show that $(\mathsf{conv}_{\mu_1}(D'))(v), \ldots,$ $(\mathsf{conv}_{\mu_k}(D'))(v)$ has to be added to G_{ug}, because $(\mathsf{conv}_{\mu_1}(A_1))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_1))(v)$ as well as $(\mathsf{conv}_{\mu_1}(A_2))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_2))(v)$ are in G_{ug}. Obviously, there exists the variable mappings $\mu' \in \mathcal{B}_{ns}(A_1, v)$ and $\mu'' \in \mathcal{B}_{ns}(A_2, v)$ with $\mu = \mathsf{dom}(\mu') \cup \mathsf{dom}(\mu'')$ and for each $x \in (\mathsf{dom}(\mu') \cap \mathsf{dom}(\mu''))$ it holds that $\mu'(x) = \mu''(x)$. Thus, $\mu' \subseteq \mu$ and $\mu'' \subseteq \mu$ and as a consequence of the completion of μ' and μ'' it follows that $\{\mu_1, \ldots, \mu_k\} \subseteq \{\mu'_1, \ldots, \mu'_k\}$ and $\{\mu_1, \ldots, \mu_k\} \subseteq \{\mu''_1, \ldots, \mu''_k\}$. Therefore, the concept facts $(\mathsf{conv}_{\mu_1}(A_1))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_1))(v)$ and $(\mathsf{conv}_{\mu_1}(A_2))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_2))(v)$ are also in G_{ug}.

- The $\downarrow$-rule for a concept fact $\downarrow x.D'(a)$ adds D' to the label of a and the variable mapping $\{x \mapsto a\}$ to $\mathcal{B}_{ns}(D', a)$. We have to show that the concept facts $(\mathsf{conv}_{\mu_1}(D'))(a)$, $\ldots$, $(\mathsf{conv}_{\mu_k}(D'))(a)$ with $\{\mu_1, \ldots, \mu_k\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{x \mapsto a\})$ are also added to G_{ug} by rule applications. But this is obviously the case, because in G_{ug} we have the concept facts $(\mathsf{conv}_{\mu_1}(\downarrow x.D'))(a), \ldots, (\mathsf{conv}_{\mu_k}(\downarrow x.D'))(a)$, which is nothing else than the unfolding of the concept facts $(\top \sqcap \mathsf{conv}_{\mu_1}(D'))(a), \ldots, (\top \sqcap \mathsf{conv}_{\mu_k}(D'))(a)$. Furthermore, we have to show that $(\top \sqcap \mathsf{conv}_{\mu_1}(D'))(a), \ldots, (\top \sqcap \mathsf{conv}_{\mu_k}(D'))(a)$ is added to G_{ug}, because, as a consequence of the axiom $O \sqsubseteq \downarrow x.D'$, $\downarrow x.D'(a)$ is added to G_{ns}. Obviously, this is the case, because for G_{ug} we have the axioms $\{a\} \sqsubseteq \top \sqcap \mathsf{conv}_{\mu_1}(D'), \ldots, \{a\} \sqsubseteq \top \sqcap \mathsf{conv}_{\mu_k}(D')$.
- The application of the gr-rule adds for a concept fact $gr(D')(v)$ and a (possibly empty) variable mapping $\mu \in \mathcal{B}^{\epsilon}_{ns}(gr(D'), v)$ the concept facts $D_{[\mu_1]}, \ldots, D_{[\mu_k]}$ with $\{\mu_1, \ldots, \mu_k\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\mu\})$. We have to show that $D_{[\mu_1]}, \ldots, D_{[\mu_k]}$ are also added to the completion graph G_{ug} by rule applications. Again, this is obviously the case, because in G_{ug} we have the concept facts $(\mathsf{conv}_{\mu_1}(gr(D')))(v)$, $\ldots$, $(\mathsf{conv}_{\mu_k}(gr(D')))(v)$, which is the same as $gr_{\mu_1}(D')(v), \ldots, gr_{\mu_k}(D')(v)$. □

The extended tableau algorithm is still terminating. This is due to the fact that the number of variable mappings is limited by the number of ABox individuals and the number of variables in axioms. Thus, blocking is ensured since the nodes in the completion graph can only be labelled with a limited number of concepts and only a limited number of variable mappings can be associated to these concepts.

Lemma 7 (Termination) *Let $\mathcal{L}$ be a Description Logic without nominal schemas and $\mathcal{LV}$ its extension with nominal schemas. Extending a tableau decision procedure for the satisfiability of $\mathcal{L}$-TBoxes based on the rules of Table 1 with the rules of Table 3 results in a terminating algorithm for absorbed $\mathcal{LV}$-TBoxes.*

As a result, we obtain a terminating tableau algorithm that is sound and complete for absorbed TBoxes with nominal schema axioms:

Theorem 3 *Let $\mathcal{L}$ be a Description Logic without nominal schemas and $\mathcal{LV}$ its extension with nominal schemas. Extending a tableau decision procedure based on the rules of Table 1*

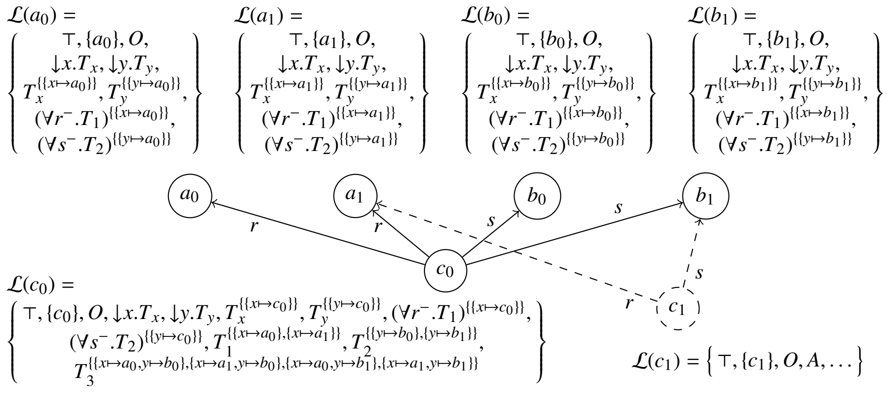
$\mathcal{L}(a_0) = \{\top, \{a_0\}, O, \downarrow x.T_x, \downarrow y.T_y, T_x^{\{\{x\mapsto a_0\}\}}, T_y^{\{\{y\mapsto a_0\}\}}, (\forall r^-.T_1)^{\{\{x\mapsto a_0\}\}}, (\forall s^-.T_2)^{\{\{y\mapsto a_0\}\}}\}$
$\mathcal{L}(a_1) = \{\top, \{a_1\}, O, \downarrow x.T_x, \downarrow y.T_y, T_x^{\{\{x\mapsto a_1\}\}}, T_y^{\{\{y\mapsto a_1\}\}}, (\forall r^-.T_1)^{\{\{x\mapsto a_1\}\}}, (\forall s^-.T_2)^{\{\{y\mapsto a_1\}\}}\}$
$\mathcal{L}(b_0) = \{\top, \{b_0\}, O, \downarrow x.T_x, \downarrow y.T_y, T_x^{\{\{x\mapsto b_0\}\}}, T_y^{\{\{y\mapsto b_0\}\}}, (\forall r^-.T_1)^{\{\{x\mapsto b_0\}\}}, (\forall s^-.T_2)^{\{\{y\mapsto b_0\}\}}\}$
$\mathcal{L}(b_1) = \{\top, \{b_1\}, O, \downarrow x.T_x, \downarrow y.T_y, T_x^{\{\{x\mapsto b_1\}\}}, T_y^{\{\{y\mapsto b_1\}\}}, (\forall r^-.T_1)^{\{\{x\mapsto b_1\}\}}, (\forall s^-.T_2)^{\{\{y\mapsto b_1\}\}}\}$
a_0
a_1
b_0
b_1
r
r
s
s
c_0
r
s
c_1
$\mathcal{L}(c_0) = \{\top, \{c_0\}, O, \downarrow x.T_x, \downarrow y.T_y, T_x^{\{\{x\mapsto c_0\}\}}, T_y^{\{\{y\mapsto c_0\}\}}, (\forall r^-.T_1)^{\{\{x\mapsto c_0\}\}}, (\forall s^-.T_2)^{\{\{y\mapsto c_0\}\}}, T_1^{\{\{x\mapsto a_0\},\{x\mapsto a_1\}\}}, T_2^{\{\{y\mapsto b_0\},\{y\mapsto b_1\}\}}, T_3^{\{\{x\mapsto a_0,y\mapsto b_0\},\{x\mapsto a_1,y\mapsto b_0\},\{x\mapsto a_0,y\mapsto b_1\},\{x\mapsto a_1,y\mapsto b_1\}\}}\}$
$\mathcal{L}(c_1) = \{\top, \{c_1\}, O, A, \ldots\}$

Fig. 3 Naive propagation and resulting combinatorial explosion of variable mappings

for satisfiability of $\mathcal{L}$-TBoxes with the rules of Table 3 yields a decision procedure for the satisfiability of absorbed $\mathcal{LV}$-TBoxes.

5 Backward Chaining Optimisations

In comparison to the upfront grounding approach, the nominal schema absorption is usually a huge improvement for knowledge bases, where axioms with absorbable nominal schemas do not match to every combination of ABox individuals. However, the propagation of variable mappings can still lead to practical problems. On the one hand, it is unfavourable that the mappings are created and propagated to many nodes and even to such nodes, where the conditions of the absorptions cannot be satisfied. On the other hand, if there are several neighbour nodes that satisfy some absorption condition, then the join potentially creates quite a lot of new mappings.

Example 4 In order to illustrate the problems, let us assume that we have an axiom $\exists r.\{x\} \sqcap \exists s.\{y\} \sqcap A \sqsubseteq B$, which is absorbed as follows:

$$O \sqsubseteq \downarrow x.T_x \qquad T_x \sqsubseteq \forall r^-.T_1 \qquad O \sqsubseteq \downarrow y.T_y$$
$$T_y \sqsubseteq \forall s^-.T_2 \qquad (T_1 \sqcap T_2) \sqsubseteq T_3 \qquad (T_3 \sqcap A) \sqsubseteq T_4$$
$$T_4 \sqsubseteq gr(B).$$

Moreover, the knowledge base contains the assertions:

$$r(c_0, a_0) \qquad r(c_0, a_1) \qquad s(c_0, b_0) \qquad s(c_0, b_1).$$

Subsequently, the completion graph in Figure 3 is generated by testing the consistency of the knowledge base. Obviously, both of the aforementioned problems occur in the generated completion graph. Due to the missing concept A in the label of node c_0, it is impossible to propagate the variable mappings to the grounding concept $gr(B)$. Nevertheless, the algorithm creates mappings with new bindings for the variables x and y for each node. Furthermore, the concept T_3 in the label of c_0 is already associated with four new variable mappings that are created by joining the mappings associated to T_1 and T_2.

Although the problems cannot be completely avoided in worst-case scenarios, it is nevertheless possible to optimise the creation and combination of variable mappings with backward chaining for many practical knowledge bases. In Example 4, no individual can ever have the grounding concept in its label. To make the example more interesting, let us assume that the knowledge base of Example 4 is extended by the assertions $r(c_1, a_1)$, $s(c_1, b_1)$ and $A(c_1)$ (cf. Figure 3, extension in dashed lines). Now, the basic idea is to first detect, with a simpler method, "interesting" nodes that can satisfy the conditions of the absorption, i.e., nodes that can possibly have the grounding concept in their label, and also those ABox individuals that might be "candidates" for binding the nominal schema variables.

Let us assume that we are able to detect c_1 as "interesting" with a_1 as a "candidate" for x and b_1 as a "candidate" for y. We now propagate these binding candidates back from the interesting node to the concepts and nodes that are relevant to imply the grounding concept for c_1. For example, the axiom $(T_1 \sqcap T_2) \sqsubseteq T_3$ tells us that the variable mappings that are associated to T_1 and T_2 on the node c_1 have to be joined and then propagated to T_3 before the grounding concept can be implied. If we can use the information of the back propagated binding candidates, then we can limit the join of variable mappings such that only those mappings are combined, which represent the expectation of the candidates, i.e., we combine only the mapping μ_1 and μ_2 for which we know that for every $z \in \mathsf{dom}(\mu_1 \cup \mu_2)$, $(\mu_1 \cup \mu_2)(z)$ is a back propagated candidate for z. Thus, we can control the join with the back propagated binding candidates. Of course, to retain completeness, the set of binding candidates must be a superset of those bindings in variable mappings that are indeed required to ground all necessary concepts. Moreover, we can further propagate the candidates back over the r and s roles to control the binder concepts, which is, as a consequence of the axioms $T_x \sqsubseteq \forall r^-.T_1$ and $T_y \sqsubseteq \forall s^-.T_2$, also a requirement to imply the grounding concept.

We next describe a more systematic approach to this idea of back propagation from nodes such as c_1 in the above example. A nice feature of this approach is that our absorption algorithms only require slight modifications for this purpose.

In the following, we extend the tableau and absorption algorithm such that we can propagate candidates for variables and use the (back) propagated candidates to control the creation and combination of variable mappings. For now, let us assume that for each nominal schema variable x all individuals $a_1, \ldots, a_m$, which are a candidate for x, are already identified. Thus, we get the *binding candidates* $x/a_1, \ldots, x/a_m$ and we encode these bindings also as variable mappings, i.e., if a_j is an individual that is a candidate for the variable x, then we encode this in the variable mapping $\{x \mapsto a_j\}$. Hence, we can also use the rules in Table 3 for the back propagation of the binding candidates. To consider the back propagated bindings in the $\downarrow$- and $\sqsubseteq_2$-rules, it is necessary to know to which $\downarrow x.C$ concepts and $(A_1 \sqcap A_2) \sqsubseteq C$ axioms the back propagated bindings are associated. Moreover, to distinguish the previously introduced concepts (rules) from the new ones for the optimisation, we use $\downarrow_{\mathsf{BP}} x.C$ and $(A_1, A_2) \rightarrow_{\mathsf{BP}} C$ for the new binder and join concepts ($\downarrow_{\mathsf{BP}}$ and $\rightarrow_{\mathsf{BP}}$ for the new rules), where BP denotes the considered back propagation. Note, instead of using a binary absorption axiom of the form $(A_1 \sqcap A_2) \sqsubseteq C$, we now use the concept $(A_1, A_2) \rightarrow_{\mathsf{BP}} C$, because then the binding candidates can be directly propagated to the concept $(A_1, A_2) \rightarrow_{\mathsf{BP}} C$ and this makes it easier to consider the candidates in the associated rule.

The new rules with the considered backward chaining propagation are shown in Table 4. The main difference is that now only those variable mappings are created and combined, for which all the individuals can be found in the back propagated binding candidates. Thus, the $\downarrow_{\mathsf{BP}}$-rule creates a new variable mapping $\{x \mapsto v\}$ on a node v only if the concept $\downarrow_{\mathsf{BP}} x.C$ is in the label of v and the backward chaining has propagated the binding candidate x/v to the concept $\downarrow_{\mathsf{BP}} x.C$ in the node v. Since the binding x/v is also propagated as mapping, we can use $\{x \mapsto v\} \in \mathcal{B}(\downarrow_{\mathsf{BP}} x.C, v)$ as condition to trigger the application of the $\downarrow_{\mathsf{BP}}$-rule. Analogously, the $\rightarrow_{\mathsf{BP}}$-rule only creates the join of the variable mappings μ_1 and μ_2 on a

Table 4 Tableau rules to handle variable mappings with considered backward chaining propagations

$\rightarrow_{\mathsf{BP}}$-rule:	if	$(A_1, A_2) \rightarrow_{\mathsf{BP}} C \in \mathcal{L}(v)$, $\{A_1, A_2\} \subseteq \mathcal{L}(v)$, v not indirectly blocked, and
		if $\mathcal{B}(A_1, v) \cup \mathcal{B}(A_2, v) = \emptyset$, $C \notin \mathcal{L}(v)$
		then $\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$
		if there exist $\mu \in (\mathcal{B}(A_1, v) \bowtie^{\epsilon} \mathcal{B}(A_2, v))$ with $\{x \mapsto \mu(x)\} \in \mathcal{B}((A_1, A_2) \rightarrow_{\mathsf{BP}} C, v)$ for all $x \in \mathsf{dom}(\mu)$, $C \notin \mathcal{L}(v)$ or $\mu \notin \mathcal{B}(C, v)$
		then $\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$ and $\mathcal{B}(C(v)) \longrightarrow \mathcal{B}(C(v)) \cup \{\mu\}$
$\downarrow_{\mathsf{BP}}$-rule:	if	$\downarrow_{\mathsf{BP}} x.C \in \mathcal{L}(v)$, v not indirectly blocked, $\{x \mapsto v\} \in \mathcal{B}(\downarrow_{\mathsf{BP}} x.C, v)$, 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 $\mathcal{B}(C, v) \longrightarrow \{\{x \mapsto v\}\}$

Algorithm 8 $\mathsf{absorbJoined}_{\mathsf{BP}}(S, A_{\mathsf{BP}})$

```
Output: Returns the atomic concept that is implied by the join of the absorptions of $S$
 1: $S' \leftarrow \emptyset$
 2: for all $C \in S$ with $\mathsf{Vars}(C) = \emptyset$ do
 3:     $A' \leftarrow \mathsf{absorbConcept}(C)$
 4:     $S' \leftarrow S' \cup \{A'\}$
 5: end for
 6: $S'' \leftarrow \{\ \mathsf{absorbJoined}(S') \setminus \{\top\}\ \}$
 7: for all $C \in S$ with $\mathsf{Vars}(C) \neq \emptyset$ do
 8:     $A' \leftarrow \mathsf{absorbConcept}_{\mathsf{BP}}(C, A_{\mathsf{BP}})$
 9:     $S'' \leftarrow S'' \cup \{A'\}$
10: end for
11: while $A_1 \in S''$ and $A_2 \in S''$ and $A_1 \neq A_2$ do
12:     $T \leftarrow$ fresh atomic concept
13:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{\mathsf{BP}} \sqsubseteq (A_1, A_2) \rightarrow_{\mathsf{BP}} T\}$
14:     $S'' \leftarrow (S'' \cup \{T\}) \setminus \{A_1, A_2\}$
15: end while
16: if $S'' = \emptyset$ then return $\top$
17: else return the element $A' \in S''$                    ▷ $S''$ is a singleton
18: end if
```

node v if all individuals for the variables can be found in the back propagated bindings, i.e., $\{x \mapsto (\mu_1 \cup \mu_2)(x)\} \in \mathcal{B}((A_1, A_2) \rightarrow_{\mathsf{BP}} C, v)$ for every $x \in \mathsf{dom}(\mu_1 \cup \mu_2)$.

In order to support the backward chaining propagation in the absorption algorithm, some minor adjustments in the absorbJoined and absorbConcept functions are necessary, which results in the new $\mathsf{absorbJoined}_{\mathsf{BP}}$ and in the new $\mathsf{absorbConcept}_{\mathsf{BP}}$ function. Both new functions are extended to create the concepts and axioms for the back propagation of binding candidates during the absorption.

Again, $\mathsf{absorbJoined}_{\mathsf{BP}}$ is joining several (possibly fresh) atomic concepts, which are created or simply returned by the functions absorbConcept or $\mathsf{absorbConcept}_{\mathsf{BP}}$. Additionally, $\mathsf{absorbJoined}_{\mathsf{BP}}$ (Algorithm 8) propagates the bindings candidates, which were propagated to the concept A_{BP}, back to all join concepts of the form $(A_1, A_2) \rightarrow_{\mathsf{BP}} C$ (line 13). Since not all concepts of S necessarily still contain nominal schema variables, the concepts without nominal schema variables are absorbed as before by the absorbJoined and absorbConcept functions (lines 2-6).

The adjustment of $\mathsf{absorbConcept}_{\mathsf{BP}}$ in Algorithm 9 is very similar. The function is also called with an additional atomic concept A_{BP}, for which the back propagation for the new absorption is appended. As an example, for the absorption of a $\forall r.C$ concept, the back propagation is extended with the propagation over an r-edge to a fresh atomic concept $T_{\mathsf{BP}-nb}$ (lines 2-3). The bindings that are back propagated to $T_{\mathsf{BP}-nb}$ can then be used to control the join or also to limit the creation of new variable mappings.

Algorithm 9 $\mathsf{absorbConcept}_{\mathsf{BP}}(C, A_{\mathsf{BP}})$

```
Output: Returns the atomic concept for the absorption of $C$
 1: if $C = \forall r.C'$ then
 2:     $T_{\mathsf{BP}-nb} \leftarrow$ fresh atomic concept
 3:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{\mathsf{BP}} \sqsubseteq \forall r.T_{\mathsf{BP}-nb}\}$
 4:     $A_{nb} \leftarrow \mathsf{absorbJoined}_{\mathsf{BP}}(\mathsf{collectDisjuncts}(C', true), T_{\mathsf{BP}-nb})$
 5:     $T \leftarrow$ fresh atomic concept
 6:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall\, \mathsf{inv}(r).T\}$
 7:     return $T$
 8: else if $C = \leqslant n\ r.C'$ then
 9:     $T_{\mathsf{BP}-nb} \leftarrow$ fresh atomic concept
10:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{\mathsf{BP}} \sqsubseteq \forall r.T_{\mathsf{BP}-nb}\}$
11:     $A_{nb} \leftarrow \mathsf{absorbJoined}_{\mathsf{BP}}(\mathsf{collectDisjuncts}(\mathsf{nnf}(\neg C'), true), T_{\mathsf{BP}-nb})$
12:     $T \leftarrow$ fresh atomic concept
13:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall\, \mathsf{inv}(r).T\}$
14:     return $T$
15: else if $C = \neg\{x\}$ then
16:     $T_x \leftarrow$ fresh atomic concept
17:     $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{(O \sqcap A_{\mathsf{BP}}) \sqsubseteq {\downarrow_{\mathsf{BP}}} x.T_x\}$
18:     return $T_x$
        ...
19: end if
```

Example 5 If we absorb the axiom $\exists r.\{x\} \sqcap \exists s.\{y\} \sqcap A \sqsubseteq B$ of Example 4 with the new absorption algorithms, which also generate the back propagation of binding candidates, then we obtain the following axioms:

$$T_{\mathsf{BP1}} \sqsubseteq \forall r.T_{\mathsf{BP2}} \qquad (O \sqcap T_{\mathsf{BP2}}) \sqsubseteq {\downarrow} x.T_x \qquad T_x \sqsubseteq \forall r^-.T_1$$
$$T_{\mathsf{BP1}} \sqsubseteq \forall s.T_{\mathsf{BP3}} \qquad (O \sqcap T_{\mathsf{BP3}}) \sqsubseteq {\downarrow} y.T_y \qquad T_y \sqsubseteq \forall s^-.T_2$$
$$T_{\mathsf{BP1}} \sqsubseteq (T_1, T_2) \rightarrow_{\mathsf{BP}} T_3 \qquad T_{\mathsf{BP1}} \sqsubseteq (T_3, A) \rightarrow_{\mathsf{BP}} T_4 \qquad T_4 \sqsubseteq gr(B).$$

The algorithm is called with T_{BP1} as the initial atomic concept for the back propagation, i.e., $\mathsf{absorbJoined}_{\mathsf{BP}}(\{\forall r.\neg\{x\}, \forall s.\neg\{x\}, \neg A, B\}, T_{\mathsf{BP1}})$, and T_{BP2}, T_{BP3} are atomic concepts that are additionally created for the back propagation of binding candidates. Now, if we ensure that such binding candidates are automatically propagated to T_{BP1}, then only the desired mappings are propagated to the grounding concept and the creation, combination and propagation of other mappings can be limited significantly.

5.1 Identification of Interesting Nodes and Binding Candidates

So far, we have assumed that the interesting nodes and the binding candidates used for the backward chaining are already available. In the following, we present different approaches of how these interesting nodes and bindings candidates can be identified.

A very simple, but already very effective method is to first absorb the absorbable part of an axiom W, where all nominal schemas are replaced by O. In comparison to the absorption of the original nominal schema axiom, we do not generate a grounding concept since we are only interested in the atomic concept, say A_O, which is generated for absorbing W, where the nominal schemas are replaced by O. During the expansion of a completion graph, the concept A_O now marks all nodes for which it is possible that the variable mappings are propagated to the grounding concept $gr(C)$ that is created for the absorption of the nominal schema axiom W. Obviously, if A_O is not in the label of a node v, then there is no variable mapping, which could be propagated to $gr(C)$, because the absorption with O is more general by allowing every combination of ABox individuals to match the conditions of the

Table 5 Additional tableau rules for creating binding candidates

$\rightarrow_{\mathsf{CP}}$-rule:	if	$(A_1, A_2) \rightarrow_{\mathsf{CP}} C \in \mathcal{L}(v)$, $\{A_1, A_2\} \subseteq \mathcal{L}(v)$, v not indirectly blocked,
	if	$C \notin \mathcal{L}(v)$, and $\mathcal{B}(A_1, v) \cup \mathcal{B}(A_2, v) = \emptyset$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$
	if	for all $x \in \{y \mid y \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$ with $\mu_1 \in \mathcal{B}(A_1, v), \mu_2 \in \mathcal{B}(A_2, v)\}$,
		there exists a mapping $\mu' \in \mathcal{B}(A_1, v) \cap \mathcal{B}(A_2, v)$ with $x \in \mathsf{dom}(\mu')$, and
		1. $\mu \in (\mathcal{B}(A_1, v) \cap \mathcal{B}(A_2, v))$, $\mu \in \mathcal{B}((A_1, A_2) \rightarrow_{\mathsf{CP}} C, v)$, or
		2. $\mu \in \mathcal{B}(A_1, v)$, $\mu \in \mathcal{B}((A_1, A_2) \rightarrow_{\mathsf{CP}} C, v)$ and there exists no $\mu'' \in$
		$\mathcal{B}(A_2, v)$ with $\mathsf{dom}(\mu) \cap \mathsf{dom}(\mu'') \neq \emptyset$, or
		3. $\mu \in \mathcal{B}(A_2, v)$, $\mu \in \mathcal{B}((A_1, A_2) \rightarrow_{\mathsf{CP}} C, v)$ and there exists no $\mu'' \in$
		$\mathcal{B}(A_1, v)$ with $\mathsf{dom}(\mu) \cap \mathsf{dom}(\mu'') \neq \emptyset$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$ and $\mathcal{B}(C(v)) \longrightarrow \mathcal{B}(C(v)) \cup \{\mu\}$

absorption. Thus, the concept A_O marks interesting nodes, which are good candidates to start the backward chaining propagation.

However, A_O does not provide direct means for controlling the creation and combination of mappings. Thus, for the backward chaining all possible bindings with all ABox individuals have to be propagated as candidates. Of course, in practice, also a flag or a special variable mapping can be propagated, which represents all bindings and does not lead to an unnecessary overhead. Hence, the preceding absorption with O and the resulting A_O concept ensures that the variable mappings are not uselessly created in the completion graph. At least, all the required facts have to exist in the neighbourhood of a node, even if the connection between the nodes does not exactly match the absorption conditions of the axiom. Regarding the simplicity of this preceding absorption, its usage is usually worthwhile for all axioms with nominal schema variables.

So far, the combinatorial explosion of the join is still unhandled. In the worst-case, the join creates, for an absorbed axiom with n nominal schema variables and a knowledge base with m ABox individuals, m^n different mappings and this possibly for each node in the completion graph. Of course, this cannot be avoided in the worst-case, but for many practical knowledge bases the amount of created combinations can be restricted significantly.

In the following, we present a preceding method that generates in the worst-case an amount of at most $m \cdot n$ binding candidates and can be used to control and limit the actual join, whereby we can possibly avoid creating all m^n mappings. Therefore, we simplify the $\rightarrow_{\mathsf{BP}}$-rule such that specific variable mappings are further propagated instead of combining them with the join. Again, we use a concept itself to handle the binary absorption axioms for the compatibility with the introduced backward chaining. The new $\rightarrow_{\mathsf{CP}}$-rule (depicted in Table 5) propagates a variable mapping $\mu \in \mathcal{B}((A_1, A_2) \rightarrow_{\mathsf{CP}} C, v)$ to $\mathcal{B}(C, v)$ if for every variable x that occurs in one mapping of $\mathcal{B}(A_1)$ as well as in one mapping of $\mathcal{B}(A_2)$, there exists a variable mapping $\mu' \in (\mathcal{B}(A_1, v) \cap \mathcal{B}(A_2, v))$, and

- μ is both in $\mathcal{B}(A_1, v)$ and $\mathcal{B}(A_2, v)$, or
- $\mu \in \mathcal{B}(A_1, v)$ ($\mu \in \mathcal{B}(A_2, v)$), and the variable $x \in \mathsf{dom}(\mu)$ is not used in the mappings that are associated to A_2 (A_1).

As a result, only those variable mappings of $\mathcal{B}(A_1, v)$ and $\mathcal{B}(A_2, v)$ are filtered, which are using variables that occur in the mappings of $\mathcal{B}(A_1, v)$ as well as in the mappings of $\mathcal{B}(A_2, v)$. Note, the $\rightarrow_{\mathsf{CP}}$-rule does not combine or create new variable mappings and, thus, the mappings that are propagated to $\mathcal{B}(C, v)$ also map only one variable to an individual name if the mappings of $\mathcal{B}(A_1, v)$ and $\mathcal{B}(A_2, v)$ map only one variable to an individual name. Thus, the mappings that are created by the binder concepts can be filtered with the $\rightarrow_{\mathsf{CP}}$-rule and we create the desired binding candidates encoded as mappings that map one variable to an individual name. For example, if the mappings $\{x \mapsto a_1\}, \{x \mapsto a_2\}, \{y \mapsto b\}$ are associated to A_1 and $\{x \mapsto a_1\}, \{z \mapsto c\}$ are associated to A_2, then the $\rightarrow_{\mathsf{CP}}$-rule propagates $\{x \mapsto a_1\}, \{y \mapsto b\}, \{z \mapsto c\}$ to the concept C.

We simply get an absorption algorithm that generates these kind of binding candidates by changing Algorithm 8 and 9 such that $A_{BP} \sqsubseteq (A_1, A_2) \rightarrow_{\mathsf{CP}} A'$ is added to $\mathcal{T}'$ in line 13 of Algorithm 8 instead of $A_{BP} \sqsubseteq (A_1, A_2) \rightarrow_{\mathsf{BP}} A'$. Note, for the creation of binding candidates it is also useful to use the absorption with the backward chaining optimisation, because then the creation of binding candidates can be triggered with A_O.

6 Variable Elimination Optimisations

It is often the case that the number of nominal schema variables can be reduced by rewriting the axiom. The basic idea is to replace unimportant nominal schemas with O, the special atomic concept that is added to the label of every ABox individual. A nominal schema $\{x\}$ can obviously be replaced if $\{x\}$ occurs only once in the axiom and only in a completely absorbable position. Such an occurrence merely requires that there exists an ABox individual for x, but it is not relevant to remember the individual since x is only used once. Thus, we can use O instead of $\{x\}$. If $\{x\}$ is not in a completely absorbable position, then we have to be more careful. For example, $\{x\}$ cannot be replaced by O in the axioms $A \sqsubseteq \geqslant 3\ r.\{x\}$ and $A \sqsubseteq \forall r.\{x\} \sqcap \geqslant 3\ r.\top$.

We can sometimes also eliminate nominal schema variables that have more than one occurrence in the axiom. Therefore, we need an analogous definition of *safe environments* of a nominal schema as Krötzsch et al. [17]:

Definition 16 (Safety) Let C be a completely absorbable concept in NNF, i.e., $\mathsf{isCA}(\mathsf{C}) = \mathit{true}$, and C', D sub-concepts of C such that $\forall r.D \in \mathsf{collectDisjuncts}(C', \mathit{true})$. An occurrence of a nominal schema $\{x\}$ in D is *safe* if x is the only nominal schema variable in D, i.e., $\mathsf{Vars}(D) = \{x\}$, and $\neg\{o\} \in \mathsf{collectDisjuncts}(C', \mathit{true})$, where $\{o\}$ is a nominal or a nominal schema. In this case, C' is a *safe environment* for this occurrence of $\{x\}$. A nominal schema variable x is safe for an axiom $C \sqsubseteq D$ if $\{x\}$ does not occur in any non-absorbable disjunct of $\mathsf{nnf}(\neg C \sqcup D)$, i.e., $x \notin \mathsf{Vars}(E)$ for all $E \in \mathsf{collectDisjuncts}(\mathsf{nnf}(\neg C \sqcup D), \mathit{false})$, and there is at most one not safe occurrence of $\{x\}$ in all absorbable disjuncts of $\mathsf{nnf}(\neg C \sqcup D)$.

A safe nominal schema variable can be eliminated by rewriting the axiom. For example, for $C \sqsubseteq D$, let us consider the axiom

$$\exists r.(\{x\} \sqcap \exists a.\{y\} \sqcap \exists v.\{z\}) \sqcap \exists s.(\exists a.\{y\} \sqcap \exists v.\{z\}) \sqsubseteq \exists c.\{x\}.$$

By transfering $C \sqsubseteq D$ into the form $\mathsf{nnf}(\neg C \sqcup D)$, we obtain the following disjunction

$$\forall r.(\neg\{x\} \sqcup \forall a.\neg\{y\} \sqcup \forall v.\neg\{z\}) \sqcup \forall s.(\forall a.\neg\{y\} \sqcup \forall v.\neg\{z\}) \sqcup \exists c.\{x\},$$

which contains the completely absorbable disjuncts

$$\forall r.(\neg\{x\} \sqcup \forall a.\neg\{y\} \sqcup \forall v.\neg\{z\}) \text{ and } \forall s.(\forall a.\neg\{y\} \sqcup \forall v.\neg\{z\}).$$

The occurence of the nominal schema $\{y\}$ is safe in $\forall r.(\neg\{x\} \sqcup \forall a.\neg\{y\} \sqcup \forall v.\neg\{z\})$, because of the disjuncts $\neg\{x\}$ and $\forall a.\neg\{y\}$; analogously the occurrence of $\{z\}$ is safe. Both occurrences of these nominal schemas are, however, not safe in $\forall s.(\forall a.\neg\{y\} \sqcup \forall v.\neg\{z\})$, but they do not occur in the non-absorbable part $\exists c.\{x\}$ and, hence, y and z are safe for the axiom.

In order to eliminate y and z, one builds the inverse path of universal restrictions from the safe occurrences of $\{y\}$ and $\{z\}$ to the nominal schema $\{x\}$ in the safe environment and then appends this path together with O to the not safe occurrences of y and z, respectively. For instance, the safe environment of $\{y\}$ is the concept $\forall r.(\neg\{x\} \sqcup \forall a.\neg\{y\} \sqcup \forall v.\neg\{z\})$, where $\{y\}$ is used within the universal restriction $\forall a.\neg\{y\}$. To eliminate the nominal schema variable y, we replace the not safe occurrence of $\{y\}$, which is the nominal schema $\{y\}$ in the (sub-)concept $\forall s.(\forall a.\neg\{y\} \sqcup \forall v.\neg\{z\})$, with $O \sqcap \exists a^{-}.\{x\}$, i.e., $\neg\{y\}$ is replaced by $\neg O \sqcup \forall a^{-}.\neg\{x\}$. Obviously, we can also remove $\forall a.\neg\{y\}$ from $\forall r.(\neg\{x\} \sqcup \forall a.\neg\{y\} \sqcup \forall v.\neg\{z\})$ since $\forall a.\neg\{y\}$ is now expressed by $\neg O \sqcup \forall a^{-}.\neg\{x\}$. Hence, we obtain the disjunction

$$\forall r.(\neg\{x\} \sqcup \forall v.\neg\{z\}) \sqcup \forall s.(\forall a.(\neg O \sqcup \forall a^{-}.\neg\{x\}) \sqcup \forall v.\neg\{z\}) \sqcup \exists c.\{x\}.$$

By analogously eliminating the nominal schema variable z, we obtain the disjunction

$$\forall r.\neg\{x\} \sqcup \forall s.(\forall a.(\neg O \sqcup \forall a^{-}.\neg\{x\}) \sqcup \forall v.(\neg O \sqcup \forall v^{-}.\neg\{x\})) \sqcup \exists c.\{x\}$$

which corresponds to the axiom

$$\exists r.\{x\} \sqcap \exists s.(\exists a.(O \sqcap \exists a^{-}.\{x\}) \sqcap \exists v.(O \sqcap \exists v^{-}.\{x\})) \sqsubseteq \exists c.\{x\}$$

with x as the only remaining nominal schema variable.

7 Implementation and Evaluation

Our reasoning system Konclude[5] [29] is able to deal with $\mathcal{SROIQ}$ knowledge bases and uses, besides many other optimisations, an absorption technique that is based on the one presented in Section 3. We have extended Konclude to $\mathcal{SROIQV}$ by integrating (i) an upfront grounding of nominal schema axioms and (ii) tableau extensions with different optimisations for propagating variable mappings in order to support the presented nominal schema absorption in Section 4. The upfront grounding is not only used to compare our nominal schema absorption technique, but also to eliminate axioms with nominal schemas that cannot be absorbed at all. The upfront grounding is more efficient for concepts that are certainly used in the completion graph, because upfront grounded concepts can be better preprocessed and it is not necessary to dynamically extend the knowledge base during the construction of the completion graph for the grounded concepts. This is especially useful for Konclude, since it supports the parallel processing of non-deterministic alternatives for which it would be necessary to synchronise the extensions of the knowledge base or to separately extend the knowledge base in each alternative.

Unfortunately, a straightforward implementation of the proposed propagation of variable mappings still bears the following sources of inefficiency:

- If a knowledge base contains an ABox individual a that is connected to many other individuals by a role r, some mappings are propagated to a, and the propagation is continued over r, then these mappings are propagated to all such connected r-neighbours, even if the mappings are only required on a few of them.
- Dependency information for each propagated mapping has to be held separately in order to support dependency directed backtracking [1,31].

[5] Available at `http://www.konclude.com/`

Table 6 Ontology metrics

Ontology	Expressiveness	Axioms	Classes	Properties	Individuals	Rules
$\text{UOBM}_1\backslash\text{D}$	$\mathcal{SHOIN}$	190093	69	36	25453	0
family\D	$\mathcal{ALCHOIN}$	212	19	16	23	12
ODGI\D	$\mathcal{SHIN}$	2391	346	83	356	2

Table 7 Hand-crafted DL-safe rules R1-R5 for the evaluation of $\text{UOBM}_1\backslash\text{D}$

Name	DL-safe Rule	Matches
R1	$\mathit{isFriendOf}(?x, ?y), \mathit{like}(?x, ?z), \mathit{like}(?y, ?z) \rightarrow \mathit{friendWithSameInterest}(?x, ?y)$	4,037
R2	$\mathit{isFriendOf}(?x, ?y), \mathit{takesCourse}(?x, ?z), \mathit{takesCourse}(?y, ?z) \rightarrow \mathit{friendWithSameCourse}(?x, ?y)$	82
R3	$\mathit{takesCourse}(?x, ?z), \mathit{takesCourse}(?y, ?z), \mathit{hasSameHomeTownWith}(?x, ?y) \rightarrow \mathit{classmateWithSameHomeTown}(?x, ?y)$	940
R4	$\mathit{hasDoctoralDegreeFrom}(?x, ?z), \mathit{hasMasterDegreeFrom}(?x, ?w), \mathit{hasDoctoralDegreeFrom}(?y, ?z), \mathit{hasMasterDegreeFrom}(?y, ?w), \mathit{worksFor}(?x, ?v), \mathit{worksFor}(?y, ?v), \rightarrow \mathit{workmateSameDegreeFrom}(?x, ?y)$	369
R5	$\mathit{isAdvisedBy}(?x, ?z), \mathit{isAdvisedBy}(?y, ?z), \mathit{like}(?x, ?w), \mathit{like}(?y, ?w), \mathit{like}(?z, ?w) \rightarrow \mathit{personWithSameAdviserAllSameInterest}(?x, ?y)$	286

- To create the correct dependencies, the variable mappings have to be joined on each node separately, even if we already have joined the same sets of variable mappings on other nodes.

We have, therefore, also implemented a variant of the propagation, where we create a *representative* for a set of variable mappings. We then propagate only these representatives and track the dependencies only for propagated representatives as in [28]. If a clash is discovered, then we extract and backtrack only the dependencies for those mappings, which are involved in the creation of the clash. In order to extract the relevant dependencies, we save for the representatives how they are composed from other representatives and variable mappings.

Furthermore, Konclude uses a batch processing mode for the variable mappings, i.e., Konclude tries to apply standard deterministic rules first and then the new rules that handle variable mappings are applied in the same order as the corresponding concepts and axioms are created in the absorption. Usually, this is a big advantage, because then many variable mappings can be handled together and the overhead of separate rule applications is minimized.

Our evaluation is primarily based on DL-safe rules to enable a comparison between the propagation of variable mappings that is integrated in Konclude and the DL reasoners HermiT 1.3.7[6] and Pellet 2.3.0 [24], which have dedicated rule support. To the best of our knowledge, these are the only reasoning systems that support DL-safe rules for such expressive ontologies. We have integrated a converter into Konclude, which transforms the DL-safe rules into nominal schema axioms. The conversion is straightforward, however, especially for nominal schema axioms that are obtained from DL-safe rules, it is often possible to eliminate variables as discussed in Section 6, which is intensively used by Konclude. For example, the DL-safe Rule (R1) is converted to the Nominal Schema Axiom (R1′) and then

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

Table 8 Comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by the rules R1–R5 between different techniques in seconds (additional preprocessing time in parentheses)

Rule	upfront grounding		direct propagation		representative propagation	
			without BP	with BP	without BP	with BP
R1	(10.99)	mem	9.12	7.10	5.06	3.38
R2	(10.92)	4.05	3.33	2.33	2.13	2.11
R3	(13.33)	3.55	1.98	0.62	2.20	0.76
R4	(16.44)	0.30	1.08	0.09	1.06	0.07
R5	(time)	–	1.87	0.50	1.80	0.43

$\{x\}$ and $\{z\}$ are replaced by O, which results in Axiom (R1″).

$$\mathit{isFriendOf}(?x, ?y), \mathit{like}(?x, ?z), \mathit{like}(?y, ?z) \rightarrow \mathit{friendWithSameInterest}(?x, ?y) \qquad \text{(R1)}$$

$$\{x\} \sqcap \exists \mathit{like}.(\{z\} \sqcap \exists \mathit{like}^{-}.\{y\}) \sqcap \exists \mathit{isFriendOf}.\{y\} \sqsubseteq \exists \mathit{friendWithSameInterest}.\{y\} \qquad \text{(R1}'\text{)}$$

$$O \sqcap \exists \mathit{like}.(O \sqcap \exists \mathit{like}^{-}.\{y\}) \sqcap \exists \mathit{isFriendOf}.\{y\} \sqsubseteq \exists \mathit{friendWithSameInterest}.\{y\} \qquad \text{(R1}''\text{)}$$

All experiments were carried out on an Intel Core i7 940 quad core processor running at 2.93 GHz, however, all reasoners are restricted to use only one core for the computation. All results are the average of three separate runs. The execution of a test was aborted if either a reasoner required more than 24 hours or more than 10 GByte memory which is denoted by *time* resp. *mem* in the results.

7.1 UOBM-Benchmarks

For the evaluation, we have extended the University Ontology Benchmark (UOBM) [20] with DL-safe rules. We are only using the smallest UOBM ontology since many reasoning systems already require for this ontology a lot of memory as well as reasoning time if it is extended with rules. Furthermore, we have removed all data properties from the ontology because these are not yet supported by Konclude. We refer to this ontology as $\text{UOBM}_1\backslash\text{D}$ (cf. Table 6).

The hand-crafted DL-safe rules R1–R5 are depicted in Table 7, where the number of matches for each rule in the consistency check is shown in the column on the right side, i.e., how often a rule can be instantiated with different variable bindings. However, since the $\text{UOBM}_1\backslash\text{D}$ ontology is not completely deterministic, these numbers might vary between different executions and between different reasoners. All these rules contain at least one cycle, i.e., they do not have a tree-shaped form, and hence, these rules are not completely trivial.

The $\text{UOBM}_1\backslash\text{D}$ ontology without rules can be preprocessed by Konclude in 1.03 seconds and the corresponding consistency test requires only 1.09 seconds. Table 8 shows *the increases* in reasoning time of the consistency tests for Konclude using different approaches and optimisations to handle the nominal schema axioms that are obtained by converting the DL-safe rules R1–R5 of Table 7. The additional required preprocessing time for the upfront grounding is shown in parentheses (column 2). Note, we show only the additional required times in order to facilitate the comparison, i.e., to get the actual required times, 1.03 and 1.09 have to be added to the preprocessing and reasoning times, respectively.

Clearly, the upfront grounding requires additional preprocessing time for the grounding of the rules, but the majority of the time is required for further processing the grounded axioms (e.g., absorption, lexical normalisation, etc.). Note, the upfront grounding totally fails

Table 9 Comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by the rules R1–R5 between Konclude, HermiT and Pellet in seconds

Rule	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
R1	3.38	31.46	6.33
R2	2.11	4.79	7.4
R3	0.76	1.67	142.25
R4	0.07	1.42	122.85
R5	0.43	28.41	mem

Table 10 Comparison of the increases in memory consumption of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by the rules R1–R5 between different techniques in MBytes

Rule	upfront grounding	direct propagation		representative propagation	
		without BP	with BP	without BP	with BP
R1	mem	4387	3451	1522	1234
R2	1116	729	475	350	349
R3	1089	491	125	433	160
R4	292	237	44	221	36
R5	time	375	144	367	110

Table 11 Comparison of the increases in memory consumption of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by the rules R1–R5 between HermiT, Pellet and Konclude in MBytes

Rule	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
R1	1234	≤ 10	659
R2	349	≤ 10	963
R3	160	≤ 10	3654
R4	36	≤ 10	9420
R5	110	≤ 10	mem

for R5, because the variable elimination can only eliminate two of four variables and the upfront grounding tries to generate $647,855,209$ new axioms. Furthermore, the reasoning time for the upfront grounding is not as good as for the other approaches. This is due to the fact that the implicit propagation of variable mappings as concepts requires a lot of separate rule applications and each rule application has some overhead, e.g., the dependencies have to be managed correctly. The direct propagation of variable mappings (as presented in Section 4) and the propagation of representatives are shown without as well as with the back propagation optimisation (BP) in Table 8. Of course, the back propagation can only improve the reasoning time if it enables a significant reduction of the creation, combination and propagation of variable mappings. For example, the savings for R2 are very limited, because most of the students in the UOBM ontology take courses and also have friends and thus, almost all students match the conditions to be identified as a candidate for which the creation and propagation of variable mappings is necessary. Usually, the propagation of representatives further improves the reasoning time and, moreover, since the dependencies can be stored in a more compact way, it also saves a significant amount of memory.

Table 9 shows a comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by the DL-safe rules R1–R5 of Table 7 between Konclude, HermiT and Pellet. Again, the times for the consistency test of $\text{UOBM}_1\backslash\text{D}$ without rules is not included to facilitate the comparison, i.e., to get the actual required times, it would be necessary to add for Konclude 1.09, for HermiT 23.24 and for Pellet 2.22 seconds. HermiT already requires a lot of time for the consistency test itself, which is possibly also a consequence of a delayed clausification of axioms and rules. However, since HermiT is not supporting com-

Table 12 DL-safe rules Q1–Q15 obtained from UOBM queries

Name	DL-safe Rule
Q1	$UndergraduateStudent(?x), takesCourse(?x, D0.U0/Course0) \rightarrow Q1(?x)$
Q2	$Employee(?x) \rightarrow Q2(?x)$
Q3	$Student(?x), isMemberOf(?x, D0.U0) \rightarrow Q3(?x)$
Q4	$Publication(?x), publicationAuthor(?x, ?y), Faculty(?y), isMemberOf(?y, D0.U0) \rightarrow Q4(?x)$
Q5	$ResearchGroup(?x), subOrganizationOf(?x, U0) \rightarrow Q5(?x)$
Q6	$Person(?x), hasAlumnus(U0, ?x) \rightarrow Q6(?x)$
Q7	$Person(?x), hasSameHomeTownWith(?x, D0.U0/FullProfessor0) \rightarrow Q7(?x)$
Q8	$SportsLover(?x), hasMember(D0.U0, ?x) \rightarrow Q8(?x)$
Q9	$GraduateCourse(?x), isTaughtBy(?x, ?y), isMemberOf(?y, ?z), subOrganizationOf(?z, U0) \rightarrow Q9(?x)$
Q10	$isFriendOf(?x, D0.U0/FullProfessor0) \rightarrow Q10(?x)$
Q11	$Person(?x), like(?x, ?y), Chair(?z), isHeadOf(?z, D0.U0), like(?z, ?y) \rightarrow Q11(?x)$
Q12	$Student(?x), takesCourse(?x, ?y), isTaughtBy(?y, D0.U0/FullProfessor0) \rightarrow Q12(?x)$
Q13	$PeopleWithHobby(?x), isMemberOf(?x, D0.U0) \rightarrow Q13(?x)$
Q14	$Woman(?x), Student(?x), isMemberOf(?x, ?y), subOrganizationOf(?y, U0) \rightarrow Q14(?x)$
Q15	$PeopleWithManyHobbies(?x), isMemberOf(?x, D0.U0) \rightarrow Q15(?x)$

Table 13 Comparison of the increases in reasoning time of the consistency tests for $UOBM_1 \setminus D$ extended by the rules Q1–Q15 between Konclude, HermiT and Pellet in seconds

Rule	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
Q1	≤ 0.1	≤ 0.1	3.9
Q2	≤ 0.1	≤ 0.1	720.0
Q3	≤ 0.1	≤ 0.1	2.5
Q4	≤ 0.1	≤ 0.1	2.5
Q5	≤ 0.1	≤ 0.1	12.4
Q6	≤ 0.1	≤ 0.1	2.0
Q7	≤ 0.1	≤ 0.1	mem
Q8	≤ 0.1	≤ 0.1	113.4
Q9	0.2	0.9	460.9
Q10	≤ 0.1	1.5	5.5
Q11	0.1	26.3	12.6
Q12	0.2	1.4	4.6
Q13	≤ 0.1	0.6	256.9
Q14	0.3	1.2	2730.4
Q15	1.2	≤ 0.1	2.9

plex roles in the body of rules, the consistency test might be incomplete for some rules. For example, the role *hasSameHomeTownWith* of rule R3 is transitive. Konclude uses the propagation of representatives combined with the back propagation optimisation and often tests the consistency for the $UOBM_1 \setminus D$ ontology even with rules faster than HermiT or Pellet. In contrast, HermiT hardly requires additional memory for the reasoning with the rules (less than 10 MBytes for R1–R5, cf. Table 11), whereby the memory consumption of HermiT is often better than the memory consumption of Konclude and Pellet. This is possibly a consequence of the not supported complex roles, wherefore it is not necessary to manage the dependencies as long as the new consequences from the rules are not instantiated. Table 10 and 11 shows the complete comparison of the increases in memory consumption in MBytes. Here, too, we would have to add for Konclude 564, for HermiT 684 and for Pellet 551 MBytes to get the actual memory consumption.

In addition, we have converted the accompanying queries 1–15 from the University Ontology Benchmark [20] to the DL-safe rules Q1–Q15 (see Table 12). The comparison of the increases in reasoning time of the consistency tests for UOBM$_1$\D extended by these rules is shown in Table 13. Konclude is dominating the other reasoners for nearly all rules, which is possible due to the variable elimination optimisation, which allows for absorbing the relatively simple and tree-shaped rules to ordinary concepts. HermiT can also handle many of these rules without significant performance losses, however, some of these rules again use transitive roles, wherefore HermiT might be incomplete (e.g., Q5, Q7, Q9, Q14). Pellet has more problems with the handling of the these rules and also requires significantly more memory. Pellet is even reaching the memory limit of 10 GBytes for rule Q7, whereas the other reasoners can process the consistency tests for all rules with less than one GByte.

Note, the consistency checking is also influenced by many optimisations in the reasoning system that are not related to the rule processing mechanism. For example, dependency directed backtracking [1,31] allows for evaluating only "relevant" non-deterministic alternatives with the tableau algorithm, i.e., reasoning systems that use a better dependency directed backtracking technique can prune bigger parts of the search tree. This may also influences the increase in reasoning time for the addition of rules to the knowledge base, since the rules possibly add non-determinism and cause clashes for which backtracking is required. A worse dependency directed backtracking technique then requires the evaluation of more non-deterministic alternatives, which results in more processing time even if the integrated rule processing mechanism is better. Hence, the consistency test is not an optimal way to compare different reasoners for their rule processing mechanism.

However, such comparisons are even more problematic for higher level reasoning tasks such as instance retrieval or classification. In particular, Konclude uses sophisticated caching mechanisms, where also the fully expanded and clash-free completion graph from the initial consistency check of the ABox is cached and reused for subsequent tests [29]. In addition, for each subsequent test only the nodes for those ABox individuals are processed for which it cannot be safely ensured that these nodes could be expanded in the same way as for the cached completion graph from the initial consistency check. As a consequence, only small parts of the ABox have to be reactivated for processing of subsequent instance tests or satisfiability tests of concepts that use nominals. Obviously, this significantly influences the processing time of higher level reasoning tasks. For example, Konclude needs only 0.2 seconds to classify the UOBM$_1$\D ontology after the consistency check due to these caching mechanisms, whereas Pellet requires 9.9 and HermiT 271.7 seconds. As a consequence, the comparison of the nominal schema absorption technique in Konclude with rule processing mechanisms of other reasoners for higher level reasoning tasks such as realisation is hardly informative since Konclude can often fall back on cached data, whereas many other reasoners usually repeat the construction of the ABox (especially if the knowledge base is non-deterministic).

7.2 OpenRuleBench-Benchmarks

We adapted tests from OpenRuleBench [19] to compare the reasoning systems for some specific test cases. Basically, the LargeJoin test case gives an impression of the capability of handling binary joins. Therefore, the reasoners have to create new role instantiations for the roles *c1*, *b1*, *b2* and finally *a* between ABox individuals based on the following non-recursive, tree-shaped rules:

$$\mathit{d1}(?x, ?y), \mathit{d2}(?y, ?z) \rightarrow \mathit{c1}(?x, ?z) \qquad \mathit{c1}(?x, ?y), \mathit{c2}(?y, ?z) \rightarrow \mathit{b1}(?x, ?z)$$
$$\mathit{c3}(?x, ?y), \mathit{c4}(?y, ?z) \rightarrow \mathit{b2}(?x, ?z) \qquad \mathit{b1}(?x, ?y), \mathit{b2}(?y, ?z) \rightarrow a(?x, ?z)$$

The assertions for the base roles *d1*, *d2*, *c2*, *c3* and *c4* were randomly generated. Table 14

Table 14 Comparison of the reasoning times of the consistency tests for the LargeJoin test case between Konclude, HermiT and Pellet in seconds

# individuals	# assertions per base role	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
50,000	10,000	1.2	1.0	3.7
50,000	20,000	3.4	2.0	6.2
50,000	30,000	6.9	3.4	10.5
100,000	20,000	3.5	1.6	7.0
100,000	40,000	11.1	3.8	25.4
100,000	60,000	23.6	10.5	97.1

Table 15 Comparison of the memory consumption of the consistency tests for the LargeJoin test case between Konclude, HermiT and Pellet in MBytes

# individuals	# assertions per base role	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
50,000	10,000	507	985	1452
50,000	20,000	847	1240	1739
50,000	30,000	1480	1453	2548
100,000	20,000	1129	1175	2023
100,000	40,000	2380	1358	3535
100,000	60,000	4304	1940	7400

shows the comparison of the reasoning times of the consistency tests between Konclude, HermiT and Pellet for different data sizes, i.e., the number of individuals is shown in the first column and the number of randomly added assertions for each base role is shown in the second column. Clearly, HermiT is dominating the other systems for this test case. Again, Konclude uses the back propagation optimisation and the propagation of representatives. However, the rules are very simple, wherefore the benefits of these optimisations are limited. Furthermore, Konclude does not directly add the implied role instantiation, but adds an existential restriction for the corresponding role, which is processed in a separate step and is, therefore, more costly.

Table 15 shows the comparison of the memory consumption of the consistency tests for the LargeJoin test case between Konclude, HermiT and Pellet. Also for this test case Konclude has a significant higher memory consumption than HermiT, which is not very surprising, because Konclude saves intermediate results (e.g., the propagated variable mappings with dependency information) and also requires some additional memory for the grounded existential restrictions.

LargeJoin-M is a variant of the test case LargeJoin, where only the final role a has to be instantiated by the reasoners, which is achieved by merging the rules of the LargeJoin test case together into the following rule:

$$\mathit{d1}(?x, ?y_1), \mathit{d2}(?y_1, ?y_2), \mathit{c2}(?y_2, ?y_3), \mathit{c3}(?y_3, ?y_4), \mathit{c4}(?y_4, ?z) \rightarrow a(?x, ?z).$$

Table 16 and 17 shows the comparison of the reasoning times and memory consumption, respectively, of the consistency tests for LargeJoin-M with the same test data as for LargeJoin. Now, the back propagation optimisation significantly improves the reasoning time and memory consumption for Konclude and there are also some improvements for Pellet, which is possibly due to the reduced number of role instantiations that have to be created. In contrast, HermiT performs worse for bigger data sizes than for the LargeJoin test case.

The transitive closure test TransClos-T, which is also adapted from OpenRuleBench [19], demonstrates the performance for a simple transitive/recursive problem. All individuals in the test are connected in a big cycle with the transitive role *par*, and the rule

$$\mathit{par}(?x, ?y) \rightarrow \mathit{tc}(?x, ?y)$$

Table 16 Comparison of the reasoning times of the consistency tests for the LargeJoin-M test case between Konclude, HermiT and Pellet in seconds

# individuals	# assertions per base role	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
50,000	10,000	0.7	0.9	1.6
50,000	20,000	1.1	1.4	4.2
50,000	30,000	2.6	2.6	7.6
100,000	20,000	1.4	1.6	5.7
100,000	40,000	5.1	5.3	18.0
100,000	60,000	11.8	20.4	94.9

Table 17 Comparison of the memory consumption of the consistency tests for the LargeJoin-M test case between Konclude, HermiT and Pellet in MBytes

# individuals	# assertions per base role	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
50,000	10,000	342	974	1361
50,000	20,000	483	1248	1526
50,000	30,000	702	1289	1623
100,000	20,000	723	1170	1719
100,000	40,000	1405	1393	3435
100,000	60,000	2569	1602	7114

Table 18 Comparison of the reasoning times of the consistency tests for the TransClos-T test case between Konclude and Pellet in seconds

# individuals	# random assertions	Konclude 0.4.1	Pellet 2.3.0
500	100	1.2	8021.1
500	200	1.8	7963.6
1,000	200	6.7	time
1,000	400	8.2	time

Table 19 Comparison of the reasoning times of the consistency tests for the TransClos-R test case between Konclude, HermiT and Pellet in seconds

# individuals	# random assertions	Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
500	100	2.1	1.0	mem
500	200	1.1	1.0	mem
1,000	200	5.1	4.7	mem
1,000	400	3.3	5.1	mem

is used to enforce that the transitive closure is explicitly represented with the role *tc*. Additionally, some random assertions for the role *par* were added to make the test case not completely straightforward. Table 18 shows the comparison of the reasoning times of the consistency tests for the test case TransClos-T between Konclude and Pellet. HermiT does not consider the transitivity in the body of the rule and is, therefore, omitted in this comparison. The transitivity can, however, be simulated for this test case by using the recursive rule

$$tc(?x, ?y), par(?y, ?z) \rightarrow tc(?x, ?z),$$

which is denoted by TransClos-R, i.e., TransClos-R is a variant of TransClos-T in which this rule is used instead of the transitivity axiom for the role *par*. Thus, this test case is also correctly handled by HermiT. The comparison of the reasoning times of the consistency tests for this test case is shown in Table 19. Konclude requires more or less the same amount of resources for both test cases. However, there are some differences in the propagation, wherefore the memory consumption for the test case TransClos-T is slightly higher. Although Konclude and HermiT can easily process TransClos-R with less than one GByte and approximately the same amount of reasoning time, this test case is much more problematic

Table 20 Comparison of the reasoning times of the consistency tests for ontologies with DL-safe rules between Konclude, HermiT and Pellet in seconds

Ontology		Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
family\D$_{+100}$	without rules	0.2	0.6	1.3
family\D$_{+100}$	with rules	2.5	1.3	mem
ODGI\D$_{+100}$	without rules	11.4	367.6	13.3
ODGI\D$_{+100}$	with rules	13.6	467.4	17.8

Table 21 Comparison of the memory consumption of the consistency tests for ontologies with DL-safe rules between Konclude, HermiT and Pellet in MBytes

Ontology		Konclude 0.4.1	HermiT 1.3.7	Pellet 2.3.0
family\D$_{+100}$	without rules	147	473	771
family\D$_{+100}$	with rules	954	744	mem
ODGI\D$_{+100}$	without rules	704	1302	1658
ODGI\D$_{+100}$	with rules	884	1456	1852

for Pellet, which suddenly requires more than 10 GBytes for TransClos-R compared to less than 2 GBytes for TransClos-T.

7.3 Benchmarks for Ontologies with Rules

We use family\D$_{+100}$ and ODGI\D$_{+100}$ to compare the reasoning times for ontologies with already integrated rules. The family\D ontology (see Table 6) is obtained from the family.swrl.owl demo ontology in the Protégé Ontology Library[7] by removing all data properties and rules with SWRL Built-Ins. In order to get an interesting benchmark size, we created family\D$_{+100}$ from family\D by adding 100 copies of the ABox with renamed non-nominal individuals. Analogously, we obtained ODGI\D (see Table 6) and ODGI\D$_{+100}$ from the ontology for disease genetic investigation (ODGI) from the NCBO BioPortal[8].

Table 20 shows the comparison of the reasoning times of the consistency tests for these ontologies with and without rules. Although the family\D$_{+100}$ ontology contains the rule

$$Person(?y), hasChild(?y, ?x), hasChild(?y, ?z), ?x \neq ?z \rightarrow hasSibling(?x, ?z),$$

which uses the atom $?x \neq ?z$ in the body of the rule that states that two individuals have to be different, the reasoning time of HermiT is hardly affected. Usually, such rules cannot completely absorbed and, therefore, the application of such rules lead to non-determinism. Due to the fact that this rule has to be applied quite often, the reasoning with the family\D$_{+100}$ ontology is highly non-deterministic. This seems to be more problematic for Konclude and Pellet, possibly due to their different dependency management. However, HermiT performs worse for ODGI\D$_{+100}$ than Konclude and Pellet. The ODGI\D$_{+100}$ ontology also contains transitive roles, however, they are not used in the rule bodies, and hence, also the results for HermiT are complete. Analogously, Table 21 shows the comparison of the memory consumption of the consistency tests for these ontologies with and without rules.

[7] http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
[8] http://bioportal.bioontology.org/ontologies/1086

7.4 Benchmarks beyond Rules

Unfortunately, there exists no test suite for nominal schemas and, to the best of our knowledge, there are also no other reasoners with nominal schema support. Therefore, the evaluation of nominal schema axioms, which are more expressive than DL-safe rules, is less conclusive, since the results cannot be compared to the results of other systems. Also, the comparison to the upfront grounding is often not very interesting, since the performance of the upfront grounding mainly depends on the additional required preprocessing time, which primarily relies on the number of nominal schema variables and the number of ABox individuals. Hence, the performance of the upfront grounding and the performance gain with the absorption and the propagation of variable mappings is more or less predictable and, therefore, it is easily possible to create tests, where the nominal schema absorption approach performs as much better as desired. Note, nominal schema axioms that are more expressive than DL-safe rules often reduce the absorbable amount on the left-hand side of the axiom. This is, however, not necessarily problematic, because the nominal schemas can often also be absorbed in other parts of the axiom. For instance, the nominal schema axiom

$$\{x\} \sqcap \exists \mathit{like}.\{y\} \sqcap \forall \mathit{isFriendOf}.\exists \mathit{like}.\{y\} \sqsubseteq \mathit{HasInterestAndAllFriendsSameInterest}$$

identifies for the $\mathrm{UOBM}_1\backslash\mathrm{D}$ ontology all ABox individuals, which have an interest and the friends of these individuals have at least also the same interest. Obviously, the universal restriction $\forall \mathit{isFriendOf}.\exists \mathit{like}.\{y\}$ on the left-hand side cannot be absorbed. However, $\{y\}$ also occurs in the absorbable existential restriction $\exists \mathit{like}.\{y\}$ in an absorbable position, wherefore the nominal schema absorption approach is still able to significantly reduce the grounding effort. For $\mathrm{UOBM}_1\backslash\mathrm{D}$ extended with this nominal schema axiom, Konclude requires for the consistency test with the upfront grounding 6.1 seconds of additional preprocessing time and 3.4 seconds of additional reasoning time, whereas the nominal schema absorption approach only increases the reasoning time by 0.9 seconds.

Of course, there still exists nominal schema axioms with nominal schemas that only occur in non-absorbable positions. For example, the axiom

$$\begin{aligned}\{x\} \sqcap \exists \mathit{like}.\{y\} \sqcap \forall \mathit{isFriendOf}.(\{z\} \sqcap \exists \mathit{like}.\{y\})\\ \sqsubseteq \mathit{HasInterestAndAllFriendsSameInterest}\end{aligned}$$

extends the previous nominal schema axiom with the nominal schema $\{z\}$, which further enforces that all friends also have to be known ABox individuals. For this axiom, the grounding concept that is created by the absorption has to handle the disjunction

$$\mathit{HasInterestAndAllFriendsSameInterest} \sqcup \exists \mathit{isFriendOf}.(\neg\{z\} \sqcup \neg\forall \mathit{like}.\neg\{y\})$$

and, since only bindings for the variable y are created and propagated, the grounding rule has to complete the variable mappings to the variable z by combining them with every ABox individual. As a consequence, the grounding rule adds quite a lot of different disjunctions and this for many nodes in the completion graph, wherefore Konclude with the nominal schema absorption is running out of memory. However, this would obviously also be the case with the upfront grounding if the preprocessing for the upfront grounding were able to finish within the time limit, which fails analogously to the processing of rule R5, since only the nominal schema variable x can be eliminated.

In principle, it would be possible to use a more sophisticated axiom rewriting in order to improve the absorption and, as a consequence, also the overall performance for the reasoning with such axioms. For example, the existentially required connection between $\{z\}$ and

$\{y\}$ over the role *like* within the universal restriction $\forall isFriendOf.(\{z\} \sqcap \exists like.\{y\})$ can also be expressed in the inverse direction within the absorbable existential restriction $\exists like.\{y\}$, which results in the axiom:

$$\{x\} \sqcap \exists like.(\{y\} \sqcap \exists like^{-}.\{z\}) \sqcap \forall isFriendOf.(\{z\} \sqcap \exists like.\{y\}) \\ \sqsubseteq HasInterestAndAllFriendsSameInterest.$$

Although now also $\{z\}$ can be partially absorbed, Konclude still requires for the consistency test of $\text{UOBM}_1 \backslash \text{D}$ extended by this nominal schema axiom 35.9 seconds and 8.2 GBytes due to the huge number of individuals that are found for $\{z\}$. Unsurprisingly, the intensive use of nominal schemas in not completely absorbable constructs and the frequent grounding of many such not absorbed concepts results in a lot of work for the reasoning system.

8 Conclusions

We have significantly improved the reasoning performance for nominal schemas with (i) an extended absorption algorithm as well as (ii) slight modifications of the standard tableau calculus. The resulting calculus creates bindings for nominal schema variables and is able to propagate them through the completion graph in order to use these bindings to ground the remaining and non-absorbable part of the nominal schema axioms. As a consequence, our approach allows for "collecting" the bindings for those nominal schema axioms that have to be grounded and considered for a specific node in the completion graph. We have shown the correctness of the nominal schema absorption and, moreover, we have also presented techniques for further optimisations.

The approach only improves the handling of "absorbable" axioms, but, to the best of our knowledge, this restriction is satisfied for the majority of all nominal schema axioms that are used in practical ontologies. The presented techniques have been integrated into the novel reasoning system Konclude, which is now able to handle $\mathcal{SROIQV}$ knowledge bases, and the empirical evaluation, which is primarily based on DL-safe rules, shows that our approach performs well even when compared to other well-known DL reasoners with dedicated rule support. In particular, the performance for rules with complex roles is significantly better than in other reasoning systems for more expressive DLs.

The presented techniques are also interesting for the extension of existing tableau-based DL reasoners to ordinary DL-safe rules, since they allow a direct integration of the support of DL-safe rules into the tableau algorithm, whereby an additional/separate inference mechanism for the rules is not required.

Acknowledgements

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

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, second edn. Cambridge University Press (2007)

2. Baader, F., Hollunder, B., Nebel, B., Profitlich, H.J., Franconi, E.: An empirical analysis of optimization techniques for terminological representation systems. J. of Applied Intelligence **4**(2), 109–132 (1994)
3. Blackburn, P., Tzakova, M.: Hybridizing concept languages. Annals of Mathematics and Artificial Intelligence **24**(1–4), 23–49 (1998)
4. Demri, S., Nivelle, H.: Deciding regular grammar logics with converse through first-order logic. J. of Logic, Language and Information **14**(3), 289–329 (2005)
5. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence **19**(1), 17–37 (1982)
6. Glimm, B., Horrocks, I., Motik, B., Shearer, R., Stoilos, G.: A novel approach to ontology classification. J. of Web Semantics: Science, Services and Agents on the World Wide Web **14**, 84–101 (2012)
7. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06), pp. 57–67. AAAI Press (2006)
8. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B.N., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004). Available at `http://www.w3.org/Submission/SWRL/`
9. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. J. of of Logic and Computation **9**(3), 385–410 (1999)
10. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. J. of Automated Reasoning **39**(3), 249–276 (2007)
11. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In: D.A. McAllester (ed.) Proc. 17th Int. Conf. on Automated Deduction (CADE'00), *LNCS*, vol. 1831, pp. 482–496. Springer (2000)
12. Horrocks, I., Tobies, S.: Reasoning with axioms: Theory and practice. In: Proc. 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'00), pp. 285–296. Morgan Kaufmann (2000)
13. Hudek, A.K., Weddell, G.E.: Binary absorption in tableaux-based reasoning for description logics. In: Proc. 19th Int. Workshop on Description Logics (DL'06), vol. 189. CEUR (2006)
14. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: G. Brewka, J. Lang (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08), pp. 274–284. AAAI Press (2008)
15. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note (22 June 2010). Available at `http://www.w3.org/TR/rif-overview/`
16. Krisnadhi, A., Hitzler, P.: A tableau algorithm for description logics with nominal schema. In: Proc. 6th Int. Conf. on Web Reasoning and Rule Systems (RR'12), *LNCS*, vol. 7497, pp. 234–237 (2012)
17. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: nominal schemas for integrating rules and ontologies. In: Proc. 20th Int. Conf. on World Wide Web (WWW'11), pp. 645–654. ACM (2011)
18. Krötzsch, M., Rudolph, S.: Nominal schemas in description logics: Complexities clarified. In: Proc. 14th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'14). AAAI Press (2014). Accepted, `http://korrekt.org/page/Nominal_Schemas_in_Description_Logics:_Complexities_Clarified`
19. Liang, S., Fodor, P., Wan, H., Kifer, M.: Openrulebench: an analysis of the performance of rule engines. In: Proc. 18th Int. Conf. on World wide web (WWW'09), pp. 601–610. ACM, New York, NY, USA (2009)
20. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Proc. 3rd European Semantic Web Conf. (ESWC'06), *LNCS*, vol. 4011, pp. 125–139. Springer (2006)
21. Martínez, D.C., Wang, C., Hitzler, P.: Towards an efficient algorithm to reason over description logics extended with nominal schemas. In: Proc. 7th Int. Conf. on Web Reasoning and Rule Systems (RR'13), *LNCS*, vol. 7994, pp. 65–79. Springer (2013)
22. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research **36**, 165–228 (2009)
23. Simančík, F.: Elimination of complex RIAs without automata. In: Proc. 25th Int. Workshop on Description Logics (DL'12), vol. 846. CEUR (2012)
24. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. of Web Semantics **5**(2), 51–53 (2007)
25. Steigmiller, A., Glimm, B., Liebig, T.: Extending absorption to nominal schemas. In: Proc. 26th Int. Workshop on Description Logics (DL'13), vol. 1014. CEUR (2013)
26. Steigmiller, A., Glimm, B., Liebig, T.: Nominal schema absorption. In: Proc. 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI'13), pp. 1104–1110. AAAI Press/The MIT Press (2013)
27. Steigmiller, A., Glimm, B., Liebig, T.: Optimised absorption for expressive description logics. In: Proc. 27th Int. Workshop on Description Logics (DL'14) (2014). Accepted

28. Steigmiller, A., Liebig, T., Glimm, B.: Extended caching, backjumping and merging for expressive description logics. In: Proc. 6th Int. Joint Conf. on Automated Reasoning (IJCAR'12), *LNCS*, vol. 7364, pp. 514–529. Springer (2012)
29. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: system description. J. of Web Semantics (2014). Accepted
30. Tsarkov, D., Horrocks, I.: Efficient reasoning with range and domain constraints. In: Proc. 17th Int. Workshop on Description Logics (DL'04), vol. 104. CEUR (2004)
31. Tsarkov, D., Horrocks, I., Patel-Schneider, P.F.: Optimizing terminological reasoning for expressive description logics. J. of Automated Reasoning **39**, 277–316 (2007)
32. 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/`
33. Wang, C., Hitzler, P.: A resolution procedure for description logics with nominal schemas. In: Proc. 2nd Joint Int. Semantic Technology Conf. (JIST'12), *LNCS*, vol. 7774, pp. 1–16. Springer (2012)