ulm university universität ulm

Nominal Schema Absorption Technical Report

Andreas Steigmiller, Birte Glimm, Thorsten Liebig

Ulmer Informatik-Berichte

Nr. 2013-06
Juni 2013

Nominal Schema Absorption

Technical Report

Andreas Steigmiller[1], Birte Glimm[1], and Thorsten Liebig[2]

[1] Ulm University, Ulm, Germany, <first name>.<last name>@uni-ulm.de
[2] derivo GmbH, Ulm, Germany, liebig@derivo.de

Abstract. Nominal schemas have recently been introduced as a new approach for the integration of DL-safe rules into the Description Logic framework. The efficient processing of knowledge bases with nominal schemas remains, however, challenging. We address this by extending the well-known optimisation of absorption as well as the standard tableau calculus to directly handle the (absorbed) nominal schema axioms. We implement the resulting extension of standard tableau calculi in 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.

1 Introduction

We address the problem of an efficient handling of so-called nominal schema axioms in tableau calculi for Description Logics (DLs). Nominal schemas have been introduced recently [14] as a feature for expressing arbitrary DL-safe rules (as specified in the W3C standards SWRL [7] or RIF [12]) natively in DLs and, consequently, in OWL ontologies [17]. Hence, DLs with nominal schemas provide a unified basis for OWL and rules. Although some attempts (see, e.g., [13]) 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 [10]. The resulting calculus extends a standard tableau calculus by additional rules to deal with the absorbed nominal schema axioms and shows a considerable performance improvement over existing techniques.

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

We use the same running example (or parts thereof) as Krisnadhi and Hitzler [13], 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):

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

For brevity, we shorten *hasReviewAssignment* to *r*, *hasAuthor* to *a*, *atVenue* to *v*, *hasSubmittedPaper* to *s*, and *hasConflictingAssignedPaper* to *c* in the remainder. Obviously, this axiom can neither be directly expressed in a DL knowledge base nor as ordinary DL-safe rule (e.g., if we were to express the complex concepts as role atoms, we would have to introduce a variable for the submitted paper, which then would only bind to known ABox individuals). However, 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 ABox 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 [14].

For efficient reasoning in OWL ontologies extended with nominal schemas, i.e., $\mathcal{SROIQV}$ knowledge bases, 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* [13]. 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.

The paper is organised as follows: Section 2 summarises some preliminaries about model construction calculi and absorption. 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 $\mathcal{SROIQ}$ [6], which underpins OWL 2, but our technique to handle nominal schemas can easily be used with $\mathcal{SROIQV}$ too, since we simply add a number of additional rules to an existing tableau calculus.

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*.*

*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 $a \in N_I$, $x \in N_V$, $n \in \mathbf{N}_0$, $A \in N_C$, 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:

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

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 $r(a, b)$, 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. A* knowledge base *$\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$ 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.*

Note, 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.

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".

Definition 4 (Completion Graph). *A* completion graph *for $\mathcal{K}$ is a directed graph $G = (V, E, \mathcal{L}, \dot{\neq})$. Each node $v \in V$ (edge $\langle v, v' \rangle \in E$) is labelled with a set of concepts (roles) occurring in $\mathcal{K}$. The symmetric binary relation $\dot{\neq}$ is used to keep track of inequalities between nodes in V.*

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 5 (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*

Table 1. Basic expansion rules for $\mathcal{ALC}$ TBoxes

$\sqsubseteq_1$-rule:	if	$A \in \mathcal{L}(v)$, $A \sqsubseteq C \in \mathcal{K}$, v not indirectly blocked, and $\{C\} \notin \mathcal{L}(v)$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C\}$
$\sqcap$-rule:	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$, v not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule:	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$, v not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then	$\mathcal{L}(v) \longrightarrow \mathcal{L}(v) \cup \{H\}$ for some $H \in \{C_1, C_2\}$
$\exists$-rule:	if	$\exists r.C \in \mathcal{L}(v)$, v not blocked, and n has no r-neighbour v' with $C \in \mathcal{L}(v')$
	then	create new node v' and 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 not indirectly blocked, and
		there is an r-neighbour v' of v with $C \notin \mathcal{L}(v')$
	then	$\mathcal{L}(v') \longrightarrow \mathcal{L}(v') \cup \{C\}$

predecessor, and descendant *is the transitive closure of successor. A node v_2 is called an* r-successor *of a node v_1 if $r \in \mathcal{L}(\langle v_1, v_2 \rangle)$; v_2 is called an* r-predecessor *of v_1 if v_1 is an r-successor of v_2. 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 (r^-- successor) of v_2.*

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 duality between existential and universal restrictions, and between atmost and atleast number restrictions [9]. 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). Note, Table 1 shows only the expansion rules for $\mathcal{ALC}$, i.e., to support the concept constructors of more expressive Description Logics, we have to add the corresponding expansion rules, e.g., for $\mathcal{ALCOIQ}$ also the o-, ch-, $\geqslant$-, $\leqslant$- and NN-rules are necessary. However, these rules are not affected by our extension for handling absorbed nominal schema axioms and, thus, we omit their presentation here.

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.

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* [8] 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 6 (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 a node w 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

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

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

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 indirectly adds the internalised concept with an auxiliary axiom $\top \sqsubseteq C_I$ by the $\sqsubseteq_1$-rule, since the concept $\top$ is added to every node of a completion graph. Clearly, internalisation 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 added to the label of the node. 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 node label. This *lazy unfolding* for atomic concepts is already realised with the $\sqsubseteq_1$-rule of Table 1, 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, nevertheless, avoid the handling of $\top \sqsubseteq \mathsf{nnf}(\neg C \sqcup D)$ one uses elaborate transformations in practise. Such transformations are performed in a preprocessing step called absorption, which we describe in more detail in Section 2.3 and 3.

2.3 Absorption

The absorption algorithm extracts those conditions of a disjunction for which it can be ensured that if one of these conditions is not satisfied for a node in a completion graph, then at least one alternative of the disjunction is trivially satisfiable. The extracted conditions are then used for expressing the disjunction in such a way that nondeterminism can be avoided as much as possible in the tableau algorithm. 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. Thus, we could only add the disjunction to nodes that have at least one r-successor. We can, however, go even further by first identifying nodes that satisfy 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:

$$B_1 \sqsubseteq T \qquad B_2 \sqsubseteq T \qquad T \sqsubseteq \forall r^-.(\exists s.A_2),$$

where T is a fresh atomic concept. Here, B_1 and B_2 have been absorbed (i.e., moved to the left-hand side of the axiom) and the concept T is used to enforce the semantics of the original axiom. We call $\forall r.(\neg B_1 \sqcap \neg B_2)$ completely absorbable since it no longer contributes a disjunct. The goal of the absorption preprocessing step is, therefore, the extraction of such easy to verify conditions that allow for expressing a GCI by possibly several 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* [11], 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 binary, 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, which adds the concept C to a label only if A_1 and A_2 are already present.

3 Absorption Algorithm

Since our handling of nominal schemas is based on absorption methods, we next present an improved variant of a recursive binary absorption algorithm, which we then extend

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 ⊔ C_2 then
 3:         return isCA(C_1) ∧ isCA(C_2)                ▷
 4:     else if C = C_1 ⊓ C_2 then
 5:         return isCA(C_1) ∧ isCA(C_2)
 6:     else if C = ∀r.C' then
 7:         return isCA(C')                             ▷
 8:     else if C = ⩽ n r.C' then
 9:         return false                                ▷
10:     else if C = ¬{a} then
11:         return true
        ...
12:     else if C = ¬A then
13:         if A is not acyclic then
14:             return false
15:         end if
16:         for all A ≡ C' ∈ T do
17:             if ¬isCA(nnf(¬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 ⊔ C_2 then
 3:         return isPA(C_1) ∨ isPA(C_2)                ▷
 4:     else if C = C_1 ⊓ C_2 then
 5:         return isPA(C_1) ∧ isPA(C_2)
 6:     else if C = ∀r.C' then
 7:         return true                                 ▷
 8:     else if C = ⩽ n r.C' then
 9:         return n ⩾ 0                                ▷
10:     else if C = ¬{a} then
11:         return true
        ...
12:     else if C = ¬A then
13:         if A is not acyclic then
14:             return false
15:         end if
16:         for all A ≡ C' ∈ T do
17:             if ¬isPA(nnf(¬C')) then
18:                 return false
19:             end if
20:         end for
21:         return true
22:     end if
23:     return false
24: end procedure
```

to nominal schemas in the next section. Our algorithm is well-suited for further optimisations (e.g., backward chaining) and it improves the original binary absorption by allowing the partial absorption of parts of the axioms without creating additional disjunctions. For example, the TBox axiom $\exists r.(A \sqcap \forall r.C) \sqsubseteq D$ is, without absorption, handled as $\top \sqsubseteq \forall r.(\neg A \sqcup \exists r.\neg C) \sqcup D$. None of the disjuncts can be absorbed completely, but it is nevertheless possible to delay the processing of the disjunction until there is an r-neighbour with the concept A in its label. In order to capture this, the absorption rewrites the axiom such that the disjunction is propagated from a node with A in its label to all r^--neighbours (if there are any), which results in $A \sqsubseteq \forall r^-.(\forall r.(\neg A \sqcup \exists r.\neg C) \sqcup D)$.

In the following, $C_{(i)}, D_{(i)}$ are (possibly complex) concepts, $A_{(i)}, T_{(i)}$ are atomic concepts with $T_{(i)}$ used for fresh concepts and S is a set of concepts. Our 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):

- 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

Algorithm 2 collectDisjuncts(C, *absorbable*)

Output: Returns the absorbable/not absorbable disjuncts of the concept C

$S \leftarrow \{C\}$
while $(C_1 \sqcup C_2) \in S$ **do**
    $S \leftarrow (S \setminus (C_1 \sqcup C_2)) \cup \{C_1, C_2\}$
end while
if *absorbable* **then**
    return $\{ C \in S \mid \mathsf{isPA}(C) \}$
else
    return $\{ C \in S \mid \neg\mathsf{isCA}(C) \}$
end if

Algorithm 3 absorbJoined(S)

Output: Returns the atomic concept that is implied by the join of the absorptions of S

$S' \leftarrow \emptyset$
for all $C \in S$ **do**
    $A' \leftarrow \mathsf{absorbConcept}(C)$
    $S' \leftarrow S' \cup \{A'\}$
end for
while $A_1 \in S'$ **and** $A_2 \in S'$ **and** $A_1 \neq A_2$ **do**
    $T \leftarrow$ fresh atomic concept
    $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{(A_1 \sqcap A_2) \sqsubseteq T\}$
    $S' \leftarrow (S' \cup \{T\}) \setminus \{A_1, A_2\}$
end while
if $S' = \emptyset$ **then return** $\top$
else return the element $A' \in S'$ ▷ S' is a singleton
end if

a comment symbol to highlight where isPA might allow additional absorption in comparison to isCA. In order to avoid an infinite recursion, we require that the concepts are acyclic for the absorption, 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 practise, because a cyclic concept A with the definition $A \sqsubseteq C$ can simply be made acyclic by representing $A \sqsubseteq C$ as $\top \sqsubseteq \neg A \sqcup C$.

- 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

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: $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall r^-.T\}$
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: $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall r^-.T\}$
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**

an empty disjunction is created (line 16 and 20, which corresponds to $\bot$. The methods absorbJoined (Algorithm 3) and absorbConcept (Algorithm 4) are recursively calling each other, whereby absorbJoined is joining several 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 [20]. Of course, the absorption can be extended to concept constructors of more expressive DLs

Algorithm 5 absorbTBox

Output: Creates a new TBox $\mathcal{T}'$ with absorbed axioms for the original TBox $\mathcal{T}$

```
 1: for all X ∈ T do
 2:     if X = A ⊑ C then
 3:         T' ← T' ∪ {A ⊑ C}
 4:     else if X = A ≡ C then
 5:         A' ← absorbJoined(collectDisjuncts(nnf(¬C), true))
 6:         if isCA(nnf(¬C)) then
 7:             T' ← T' ∪ {A ⊑ C, A' ⊑ A, A' ⊑ A+}
 8:         else if |{A ⊑ C' ∈ T} ∪ {A ≡ C' ∈ T}| > 1 then
 9:             T' ← T' ∪ {A ⊑ C, A' ⊑ nnf(¬C ⊔ A), A' ⊑ A+}
10:         else
11:             T' ← T' ∪ {A ≡ C, A' ⊑ A+}
12:         end if
13:     else if X = C ⊑ D or X = C ≡ D then
14:         A' ← absorbJoined(collectDisjuncts(nnf(¬C ⊔ D), true))
15:         {D1, ..., Dn} ← collectDisjuncts(nnf(¬C ⊔ D), false)
16:         T' ← T' ∪ {A' ⊑ D1 ⊔ ... ⊔ Dn}
17:         if X = C ≡ D then
18:             A'' ← absorbJoined(collectDisjuncts(nnf(¬D ⊔ C), true))
19:             {C1, ..., Cm} ← collectDisjuncts(nnf(¬D ⊔ C), false)
20:             T' ← T' ∪ {A'' ⊑ C1 ⊔ ... ⊔ Cm}
21:         end if
22:     end if
23: end for
```

(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 $\top \sqsubseteq \forall r^{-}.A$.

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 $A \equiv C$ in $A \sqsubseteq C$ and $C \sqsubseteq A$, because $\mathsf{nnf}(\neg C)$ 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 11). 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 $\mathsf{nnf}(\neg C)$ is partially absorbable, then the disjunction $\neg C \sqcup A$ can be triggered with A', which is generated for the absorption of $\mathsf{nnf}(\neg C)$ (Algorithm 5, line 8.

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 7,9 and 11 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 $\mathsf{nnf}(\neg C)$ 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 $\mathsf{nnf}(\neg C)$ generates the atomic concept A' and the axiom $A' \sqsubseteq A^+$ is added to $\mathcal{T}'$ (line 9 of Algorithm 5). If $\mathsf{nnf}(\neg C)$ 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 and, therefore, it is almost always useful to generate the candidate concepts [5].

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 practise, the binary absorption axioms as well as the axioms for absorbing specific concepts can be reused.

Example 1. As an example, the TBox $\mathcal{T} = \{\exists r.(\{a\} \sqcap \exists r.\{b\} \sqcap \exists r.\{c\}) \sqsubseteq \exists s.\{a\}\}$ is rewritten to $\mathcal{T}'$ by the absorbTBox procedure, where the new axioms in $\mathcal{T}'$ are:

$$\begin{array}{lll} \{a\} \sqsubseteq T_1 & \{b\} \sqsubseteq T_2 & T_2 \sqsubseteq \forall r^-.T_3 \\ \{c\} \sqsubseteq T_4 & T_4 \sqsubseteq \forall r^-.T_5 & (T_1 \sqcap T_3) \sqsubseteq T_6 \\ (T_5 \sqcap T_6) \sqsubseteq T_7 & T_7 \sqsubseteq \forall r^-.T_8 & T_8 \sqsubseteq \exists s.\{a\}. \end{array}$$

$T_1, \ldots, T_8$ are fresh atomic concepts generated by the absorption. To process the one and only axiom $\exists r.(\{a\} \sqcap \exists r.\{b\} \sqcap \exists r.\{c\}) \sqsubseteq \exists s.\{a\}$ in $\mathcal{T}$ the absorbJoined function is called for the absorbable parts of $\forall r.(\neg\{a\} \sqcup \forall r.\neg\{b\} \sqcup \forall r.\neg\{c\}) \sqcup \exists s.\{a\}$, which is only the disjunct $\forall r.(\neg\{a\} \sqcup \forall r.\neg\{b\} \sqcup \forall r.\neg\{c\})$. An atomic concept for this disjunct is created with the absorbConcept function by processing the $\forall$-concept and recursively absorbing its qualification $\neg\{a\} \sqcup \forall r.\neg\{b\} \sqcup \forall r.\neg\{c\}$, where the qualification can be simplified into the disjuncts $\neg\{a\}, \forall r.\neg\{b\}$ and $\forall r.\neg\{c\}$. The disjunct $\neg\{a\}$ can be directly absorbed to $\{a\} \sqsubseteq T_1$, for the other disjuncts another recursion is necessary to firstly generate $\{b\} \sqsubseteq T_2$ and $\{c\} \sqsubseteq T_4$ and afterwards the associated propagation of the triggers T_3 and T_5 over the r^--role with $T_2 \sqsubseteq \forall r^-.T_3$ and $T_4 \sqsubseteq \forall r^-.T_5$. The absorbJoined function is joining the atomic concepts T_1, T_3 and T_5 by the new binary axioms $(T_1 \sqcap T_3) \sqsubseteq T_6$ and $(T_5 \sqcap T_6) \sqsubseteq T_7$. Now, the absorption of the outer $\forall$-concept can be finished by adding the axiom $T_7 \sqsubseteq \forall r^-.T_8$. The remaining non-absorbable part of the disjunction is handled with $T_8 \sqsubseteq \exists s.\{a\}$. The new axioms are not causing any non-determinism and, furthermore, they can be handled very efficiently as assertions (e.g., $\{a\} \sqsubseteq C$ is equivalent to the assertion $C(a)$), by *lazy unfolding* [1] (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. For a more complex example, let $\mathcal{T} = \{A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3, (A_4 \sqcup \{a\}) \sqcap \exists s.(A_1 \sqcap \geqslant 3\ r.A_5) \sqsubseteq A_6\}$, which is rewritten to $\mathcal{T}'$ with the 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 absorbed, which generates the first three axioms of $\mathcal{T}'$, where $A_2 \sqsubseteq \forall r^-.T_1$ and $T_1 \sqsubseteq A_1^+$ are triggering the candidate concept A_1^+ of the completely defined concept A_1, and the axiom $A_1 \equiv \exists r.A_2 \sqcap \forall r.A_3$ is used for the unfolding of A_1. The other eight axioms of $\mathcal{T}'$ are created for the absorption of $(A_4 \sqcup \{a\}) \sqcap \exists s.(A_1 \sqcap \geqslant 3\ r.A_5) \sqsubseteq A_6$, similar to the previous example. 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 their structure 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 also be efficiently handled by lazy unfolding, i.e., A can be unfolded to C and $\neg A$ to $\neg C$. Moreover, the new axioms dramatically delay or completely avoid non-deterministic branching caused by disjunctions. For example, without absorption, the disjunction $(\neg A_4 \sqcap \neg\{a\}) \sqcup \forall s.(\neg A_1 \sqcup \leqslant 2\ r.A_4) \sqcup A_5$ 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, $\forall s.(\neg A_2 \sqcup \leqslant 2\ r.A_3) \sqcup A_4$, 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 $\forall s.(\neg A_1 \sqcup \leqslant 2\ r.A_4)$ 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 Correctness

Termination for the absorption algorithm itself 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 $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ such that $\mathcal{I}' \models \mathcal{T}'$ with $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\delta \in A^{\mathcal{I}'}$ 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, by 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 simplification 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 $\delta \in (\neg C_2)^{\mathcal{I}'}$ and by the induction hypothesis $\delta \in A_2^{\mathcal{I}'}$ 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 absorbConcept(C) creates $A_1 \sqsubseteq \forall r^-.A$ and A is returned by 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 $A_1 \sqsubseteq \forall r^-.A$ 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 nnf($\neg C'$) is completely absorbable.

(*Claim 3*) We construct the interpretation $\mathcal{I}'$ from $\mathcal{I}$ such that $\delta \in A^{\mathcal{I}'}$ 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 $\exists r^-.A' \sqsubseteq A_i$.

Now, for $1 \leqslant i \leqslant m$ and for each axiom $H \sqsubseteq A_i$ generated by the absorption, we exhaustively add $\delta \in \Delta^{\mathcal{I}'}$ 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) $H = \exists r^-.A'$ and $\delta \in (\exists r^-.A')^{\mathcal{I}'}$, i.e., δ has some r-neighbour γ such that $(\gamma, \delta) \in r^{\mathcal{I}'}$ and $\gamma \in A'^{\mathcal{I}'}$. We have $\delta \in A_i^{\mathcal{I}'}$ only if δ satisfies the left-hand side of an axiom $A' \sqsubseteq A_i$, $\{a\} \sqsubseteq A_i$, or $(A' \sqcap A'') \sqsubseteq A_i$, or $\exists r^-.A' \sqsubseteq A_i$. 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, because A is freshly used for $\{a\} \sqsubseteq A$.

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:

- 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 $\exists r^{-}.A_1 \sqsubseteq A_i$ 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 neither completely nor partially 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 $A \sqsubseteq D \in \mathcal{T}_2$ for each $\delta \in \Delta^{\mathcal{I}_2}$ it holds 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 $C' \sqsubseteq C$ 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' (nnf($\neg D'$)) is neither completely nor partially absorbable, then absorbConcept(C) creates $\top \sqsubseteq \forall r^{-}.A$, 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 $D_1 = \text{nnf}(\neg D')$) and D_1 is partially absorbable, then absorbConcept(C) creates $A_1 \sqsubseteq \forall r^{-}.A$, where A_1 is the atomic concept that is returned by 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:

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

4 Nominal Schema Absorption

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 enforce the addition of the right-hand side also on blockable nodes. As a consequence, typical approaches for rule processing, such as Rete [4], 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 [2]. 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.

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):

- $\mathsf{isCA}(C)$ ($\mathsf{isPA}(C)$) is extended to return that a negated occurrence of a nominal schema $\neg\{x\}$ is completely (partially) absorbable.
- $\mathsf{absorbConcept}(C)$ 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 $\{a\} \sqsubseteq O$.

Algorithm 6 Absorption extensions for Algorithms 1

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

Algorithm 7 Absorption extensions for Algorithm 4

```
1: procedure absorbConcept(C)
       ...
2:     if C = ¬{x} then
3:         T_x ← fresh atomic concept
4:         T' ← T' ∪ {O ⊑ ↓x.T_x}
5:         return T_x
6:     end if
       ...
7: end procedure
```

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 7 (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 $\mathcal{B}(C, v)$.*

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 2) 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 2) 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 $\mathcal{B}(A_1, v) \neq \emptyset$ 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 8 (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. Given two (possibly empty) sets of variable mappings M_1, M_2, let $M_1^\epsilon = \{\epsilon\}$ ($M_2^\epsilon = \{\epsilon\}$) if $M_1 = \emptyset$ ($M_2 = \emptyset$) and $M_1^\epsilon = M_1$ ($M_2^\epsilon = M_2$) otherwise. The* join *$M_1 \bowtie^\epsilon M_2$ is defined as $\{\mu_1 \cup \mu_2 \mid \mu_1 \in M_1^\epsilon, \mu_2 \in M_2^\epsilon$ and μ_1 is compatible with $\mu_2\} \setminus \{\epsilon\}$.*

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

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 $\mathcal{B}(A_2, v) = \emptyset$. 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. 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 9 (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 $\mathsf{comp}^{\mathcal{T}}_{Y}(M)$ of M w.r.t. the variable set Y and a knowledge base $\mathcal{K}$ *containing the individuals* $\mathsf{Inds}(\mathcal{K})$ *as*

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

In order to ground the concept C for a concept fact $gr(C)(v)$, the gr-rule uses the variable mappings of $\mathsf{comp}^{\mathcal{K}}_{\mathsf{Vars}(C)}(\mathcal{B}(gr(C), v))$. Since the mappings that are propagated

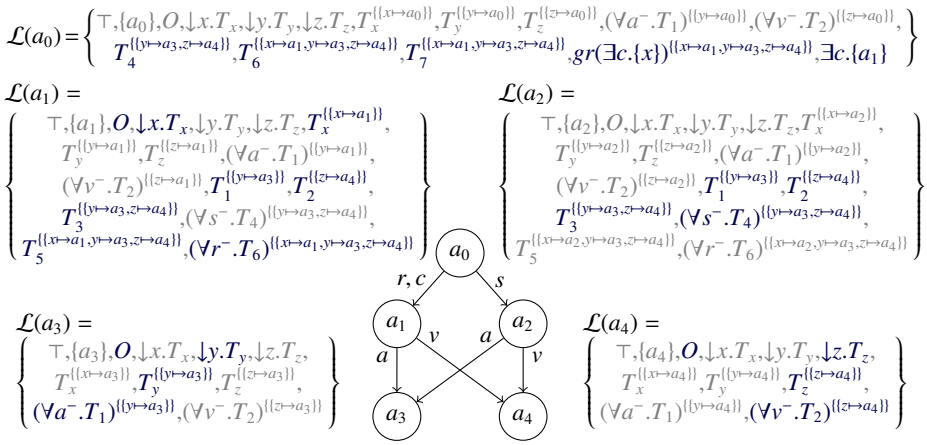

Fig. 1. Variable mapping propagation example

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 completion ensures that 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 propagating the variable mappings also over those $\forall$-concepts that are introduced to handle the automata of the role inclusion axioms [6]. Alternatively we could adapt the technique to eliminate role chains (incl. transitivity) [3]. 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 10 (Blocking with Variable Mappings). *A node v with predecessor v' is* directly blocked *if there exists a node w with predecessor w' such that*

- *v is directly pairwise blocked by w (see conditions 1 - 4 of Definition 6), 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 $D \in \mathcal{L}(v')$.*

The completion graph in Figure 1 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 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 $(O \sqcap T'_3) \sqsubseteq \downarrow x.T_x$.

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 $\mathsf{comp}^{\mathcal{T}}_Y(M)$ is analogously defined to the completion $\mathsf{comp}^{\mathcal{K}}_Y(M)$ 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:

Definition 11 (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'. Additionally,* $\mathsf{fclos}(Z)$ *is the*

extension to a set of axioms Z:

$$\mathsf{fclos}(Z) := \bigcup_{C' \sqsubseteq D' \in Z} \mathsf{clos}(\neg C' \sqcup D').$$

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:*

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

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 12 (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 $X^{\mu}_1, \ldots, X^{\mu}_n$ and $T^{\mu}_1, \ldots, T^{\mu}_g$ 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*

$$\mathsf{conv}_{\mu}(C') := \begin{cases} C' & \text{if } C' \notin \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D) \\ (\top \sqcap \mathsf{conv}_{\mu}(C'')) & \text{if } C' = {\downarrow}x.C'' \\ gr_{\mu}(D) & \text{if } C' = gr(D) \\ C'_{[T_1/T^{\mu}_1, \ldots, T_g/T^{\mu}_g]} & \text{otherwise,} \end{cases}$$

where $C'_{[T_1/T^{\mu}_1, \ldots, T_g/T^{\mu}_g]}$ denotes the syntactic replacement of each occurrence of T_i in C' with T^{μ}_i, for $1 \leq i \leq g$. The extension to axioms $\mathsf{fconv}_{\mu}(X)$ is defined as:

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

In the remainder of the section, we use $C \sqsubseteq D, \mu, \mathcal{T}, \mathcal{T}_{ns}, \mathcal{T}_{ug}, T_1, \ldots, T_g, T^{\mu}_1, \ldots, T^{\mu}_g$, $X_1, \ldots, X_n$, and $X^{\mu}_1, \ldots, X^{\mu}_n$ 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 12. The set $\{\mathsf{fconv}_\mu(X_1), \ldots, \mathsf{fconv}_\mu(X_n)\}$ 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 12, the claim holds. □

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

Definition 13 (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 12. 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 $\lambda_2, \ldots, \lambda_\ell$ be all possible variable mappings for $\mathsf{Vars}(\neg C \sqcup D)$ w.r.t. $\mathcal{T}$, i.e., $\{\lambda_2, \ldots, \lambda_\ell\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\epsilon\})$.

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 2) 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 $H \sqsubseteq \mathsf{conv}_{\lambda_2}(D'), \ldots, H \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$ 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 $(\mathsf{conv}_{\lambda_2}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$. If $A \in \mathsf{aclos}_{\mathcal{T}}(C \sqsubseteq D)$, then we would have the axioms $\mathsf{conv}_{\lambda_2}(A) \sqsubseteq \mathsf{conv}_{\lambda_2}(D'), \ldots, \mathsf{conv}_{\lambda_\ell}(A) \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$ 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 $(\mathsf{conv}_{\lambda_2}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ and thus the rule is not applicable for $(A_1 \sqcap A_2) \sqsubseteq \mathsf{conv}_{\lambda_2}(D'), \ldots, (A_1 \sqcap A_2) \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$.
  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 again not applicable after 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 set 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 $D_{\lfloor\mu\rfloor}$ 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 $D_{\lfloor\mu_1\rfloor}, \ldots, D_{\lfloor\mu_k\rfloor}$, because $D_{\lfloor\mu_1\rfloor}, \ldots, D_{\lfloor\mu_k\rfloor}$ 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 12. 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 2 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 $\lambda_2, \ldots, \lambda_\ell$ be all possible variable mappings, i.e., $\{\lambda_2, \ldots, \lambda_\ell\} = \mathsf{comp}^{\mathcal{T}}_{\mathsf{Vars}(\neg C \sqcup D)}(\{\epsilon\})$. 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 $(\mathsf{conv}_{\lambda_2}(D'))(w), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(w)$ are also added by rule applications. Obviously, this is the case, because the concept fact $\forall r.D'(v)$ corresponds to $(\mathsf{conv}_{\lambda_2}(\forall r.D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(\forall r.D'))(v)$ in 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 $\mathsf{conv}_{\lambda_2}(D'), \ldots, \mathsf{conv}_{\lambda_\ell}(D')$ 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 $(\mathsf{conv}_{\lambda_2}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ are also

added to G_{ug} by rule applications. Again, this is obviously the case, because for G_{ug} we have the rules $\mathsf{conv}_{\lambda_2}(H) \sqsubseteq \mathsf{conv}_{\lambda_2}(D'), \ldots, \mathsf{conv}_{\lambda_\ell}(H) \sqsubseteq \mathsf{conv}_{\lambda_\ell}(D')$. If $\mathcal{B}_{ns}(H, v) \neq \emptyset$, then $H = A$, the $\sqsubseteq_1$-rule adds also a variable mapping μ to $\mathcal{B}_{ns}(D', v)$ and 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 $(\mathsf{conv}_{\lambda_2}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ 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 $(\mathsf{conv}_{\lambda_2}(D'))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(D'))(v)$ 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 G_{ug} by rule applications. Let us first assume that $\mathcal{B}_{ns}(A_1, v) = \emptyset$ ($\mathcal{B}_{ns}(A_2, v) = \emptyset$). As a consequence, we have in G_{ug} the concept facts $(\mathsf{conv}_{\lambda_2}(A_2))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(A_2))(v)$ and $(\mathsf{conv}_{\mu_1}(A_2))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_2))(v)$ ($(\mathsf{conv}_{\lambda_2}(A_1))(v), \ldots, (\mathsf{conv}_{\lambda_\ell}(A_1))(v)$ and $(\mathsf{conv}_{\mu_1}(A_1))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_1))(v)$). As a consequence 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. 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, $(\mathsf{conv}_{\mu_1}(A_1))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_1))(v)$ and $(\mathsf{conv}_{\mu_1}(A_2))(v), \ldots, (\mathsf{conv}_{\mu_k}(A_2))(v)$ are at least 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 $(\top \sqcap \mathsf{conv}_{\mu_1}(D'))(a), \ldots, (\top \sqcap \mathsf{conv}_{\mu_k}(D'))(a)$. 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 $D_{[\mu_1]}, \ldots, D_{[\mu_k]}$ is also added to 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 2 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 2 *Let $\mathcal{L}$ be a Description Logic without nominal schemas and $\mathcal{LV}$ its extension with nominal schemas. Extending a tableau decision procedure for satisfiability of $\mathcal{L}$-TBoxes based on the rules of Table 1 with the rules of Table 2 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 2 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

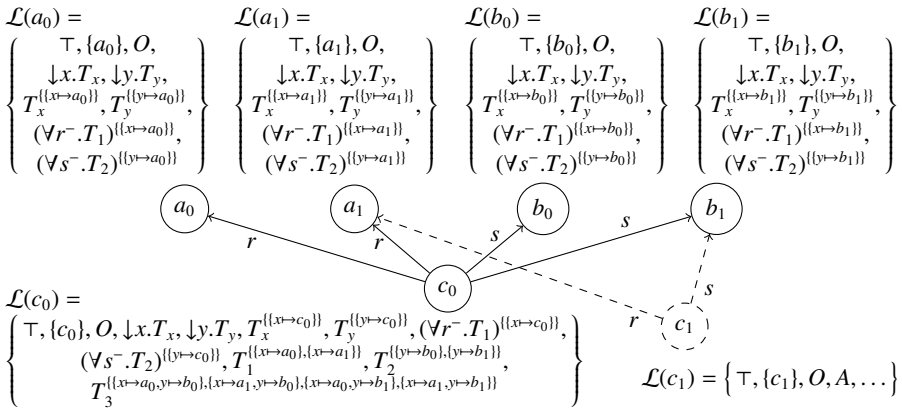

Fig. 2. Naive propagation and resulting combinatorial explosion of variable mappings

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 2, 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 the 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.

Table 3. 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 $\mathcal{B}(C, v) \longrightarrow \{\{x \mapsto v\}\}$

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 2 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 3. 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 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 nec-

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: $\quad A' \leftarrow \mathsf{absorbConcept}(C)$
4: $\quad S' \leftarrow S' \cup \{A'\}$
5: **end for**
6: $S'' \leftarrow \{\ \mathsf{absorbJoined}(S') \backslash \{\top\}\ \}$
7: **for all** $C \in S$ with $\mathsf{Vars}(C) \neq \emptyset$ **do**
8: $\quad A' \leftarrow \mathsf{absorbConcept}_{\mathsf{BP}}(C, A_{\mathsf{BP}})$
9: $\quad S'' \leftarrow S'' \cup \{A'\}$
10: **end for**
11: **while** $A_1 \in S''$ **and** $A_2 \in S''$ **and** $A_1 \neq A_2$ **do**
12: $\quad T \leftarrow$ fresh atomic concept
13: $\quad \mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{\mathsf{BP}} \sqsubseteq (A_1, A_2) \rightarrow_{\mathsf{BP}} T\}$
14: $\quad S'' \leftarrow (S'' \cup \{T\}) \setminus \{A_1, A_2\}$
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**

essary, 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 atomic concepts, which are created by $\mathsf{absorbConcept}$ or $\mathsf{absorbConcept}_{\mathsf{BP}}$. Additionally, $\mathsf{absorbJoined}_{\mathsf{BP}}$ (Algorithm 8) distributes the bindings candidates, which were already propagated back to the concept A_{BP}, 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 $\mathsf{absorbJoined}$ and $\mathsf{absorbConcept}$ 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.

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 con-

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

Output: Returns the atomic concept for the absorption of C

if $C = \forall r.C'$ **then**
  $T_{\mathsf{BP}-nb} \leftarrow$ fresh atomic concept
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{\mathsf{BP}} \sqsubseteq \forall r.T_{\mathsf{BP}-nb}\}$
  $A_{nb} \leftarrow \mathsf{absorbJoined}_{\mathsf{BP}}(\mathsf{collectDisjuncts}(C', \mathit{true}), T_{\mathsf{BP}-nb})$
  $T \leftarrow$ fresh atomic concept
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall r^{-}.T\}$
  return T
else if $C = \leqslant n\ r.C'$ **then**
  $T_{\mathsf{BP}-nb} \leftarrow$ fresh atomic concept
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{\mathsf{BP}} \sqsubseteq \forall r.T_{\mathsf{BP}-nb}\}$
  $A_{nb} \leftarrow \mathsf{absorbJoined}_{\mathsf{BP}}(\mathsf{collectDisjuncts}(\mathsf{nnf}(\neg C'), \mathit{true}), T_{\mathsf{BP}-nb})$
  $T \leftarrow$ fresh atomic concept
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{A_{nb} \sqsubseteq \forall r^{-}.T\}$
  return T
else if $C = \neg\{x\}$ **then**
  $T_x \leftarrow$ fresh atomic concept
  $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{(O \sqcap A_{\mathsf{BP}}) \sqsubseteq \downarrow_{\mathsf{BP}} x.T_x\}$
  return T_x
  ...
end if

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

Table 4. 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 $\mathcal{B}(C(v)) \longrightarrow \mathcal{B}(C(v)) \cup \{\mu\}$

However, A_O does not provide a 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, it is usually worth to use it 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 4) 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, because 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. [14]:

Definition 14 (Safety). *Let C be a concept in a completely absorbable position, then an occurrence of a nominal schema $\{x\}$ is* safe *if it occurs in D with* $\mathsf{Vars}(D) = \{x\}$, $\exists r.D \in \mathsf{collectDisjuncts}(C, true)$ *and* $\neg\{o\} \in \mathsf{collectDisjuncts}(C, true)$*, where $\{o\}$ is a nominal or a nominal schema. Thus, C is a* safe environment *for such an occurrence of $\{x\}$. A nominal schema variable x is safe for an axiom W if every occurrence of $\{x\}$ is completely absorbable and at most one occurrence of $\{x\}$ is not safe.*

A safe nominal schema variable x can be eliminated by rewriting the axiom. For example, the axiom

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

has the two safe nominal schema variables y and z, because $\{x\} \sqcap \exists a.\{y\} \sqcap \exists v.\{z\}$ is a safe environment for y and z, there is only one other not safe occurrence of y and z, and the left-hand side of the axiom is completely absorbable. We can now eliminate y and z by building the inverse path of existential restrictions from the safe occurrences of $\{y\}$ and $\{z\}$ to the nominal schema $\{x\}$ in the safe environment and appending this path together with O to the not safe occurrence of y and z. Thus, the axiom is rewritten to

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

with x as the only remaining nominal schema variable.

7 Implementation and Evaluation

Our reasoning system Konclude 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 those 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 hold separately in order to support dependency directed backtracking.
- 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 [19]. 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 rea-

Table 5. Ontology metrics

Ontology	Expressiveness	Axioms	Classes	Properties	Individuals	Rules
$UOBM_1$\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

soners HermiT 1.3.7[3] and Pellet 2.3.0 [18], 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 $\{x\}$ and $\{z\}$ are replaced by O, which results in Axiom (R1″).

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

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

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

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) [16] 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 $UOBM_1$\D (cf. Table 5).

The hand-crafted DL-safe rules R1–R5 are depicted in Table 6, 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 $UOBM_1$\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 $UOBM_1$\D ontology without rules can be preprocessed by Konclude in 1.03 seconds and the corresponding consistency test requires only 1.09 seconds. Table 7

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

Table 6. Hand-crafted DL-safe rules R1-R5 for the evaluation of $\text{UOBM}_1\setminus\text{D}$

Name	DL-safe Rule	Matches
R1	$\mathit{isFirendOf}(?x,?y), \mathit{like}(?x,?z), \mathit{like}(?y,?z) \rightarrow \mathit{friendWithSameInterest}(?x,?y)$	4,037
R2	$\mathit{isFirendOf}(?x,?y), \mathit{takesCourse}(?x,?z), \mathit{takesCourse}(?y,?z) \rightarrow \mathit{friendWithSameCourse}(?x,?y)$	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

Table 7. Comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\setminus\text{D}$ 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

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 6. 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 some 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 for R5, because the variable elimination can only eliminate two

Table 8. Comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\setminus\text{D}$ 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	0.46	1.67	142.25
R4	0.07	1.42	122.85
R5	0.43	28.41	mem

Table 9. 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 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 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

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 7. 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 8 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 6 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 complex 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 $\text{UOBM}_1\backslash\text{D}$

Table 11. DL-safe rules Q1–Q15 obtained from UOBM queries

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

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 10), 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 9 and 10 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 [16] to the DL-safe rules Q1–Q15 (see Table 11). The comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ extended by these rules is shown in Table 12. 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 test is not an optimal way to compare different reasoners for their rule processing mechanism since there are also many other optimisations that influence the required reasoning time. However, this is even more problematic for more high level reasoning tasks such as instance retrieval or classification. For example, Kon-

Table 12. Comparison of the increases in reasoning time of the consistency tests for $\text{UOBM}_1\backslash\text{D}$ 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

clude needs only 0.2 seconds to classify the $\text{UOBM}_1\backslash\text{D}$ ontology after the consistency check due to sophisticated caching mechanisms, whereas Pellet requires 9.9 and HermiT 271.7 seconds.

7.2 OpenRuleBench-Benchmarks

We adapted tests from OpenRuleBench [15] 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:

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

The assertions for the base roles *d1*, *d2*, *c2*, *c3* and *c4* were randomly generated. Table 13 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 14 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

Table 13. 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 14. 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

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:

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

Table 15 and 16 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 [15], 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

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

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 17 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

Table 15. 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 16. 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

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

Table 17. 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 18. 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

7.3 Benchmarks for Ontologies with Rules

We use family$\backslash D_{+100}$ and ODGI$\backslash D_{+100}$ to compare the reasoning times for ontologies with already integrated rules. The family$\backslash$D ontology (see Table 5) is obtained from the family.swrl.owl demo ontology in the Protégé Ontology Library[4] by removing all data properties and rules with SWRL Built-Ins. In order to get an interesting benchmark size, we created family$\backslash D_{+100}$ from family$\backslash$D by adding 100 copies of the ABox with renamed non-nominal individuals. Analogously, we obtained ODGI$\backslash$D (see Table 5) and ODGI$\backslash D_{+100}$ from the ontology for disease genetic investigation (ODGI) from the NCBO BioPortal[5].

Table 19 shows the comparison of the reasoning times of the consistency tests for these ontologies with and without rules. Although the family$\backslash D_{+100}$ ontology contains the rule

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

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$\backslash D_{+100}$ 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$\backslash D_{+100}$ than Konclude and Pellet. The ODGI$\backslash D_{+100}$ 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 20 shows the comparison of the memory consumption of the consistency tests for these ontologies with and without rules.

[4] `http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library`
[5] `http://bioportal.bioontology.org/ontologies/1086`

Table 19. 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$\backslash D_{+100}$	without rules	0.2	0.6	1.3
family$\backslash D_{+100}$	with rules	2.5	1.3	mem
ODGI$\backslash D_{+100}$	without rules	11.4	367.6	13.3
ODGI$\backslash D_{+100}$	with rules	13.6	467.4	17.8

Table 20. 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$\backslash$D$_{+100}$	without rules	147	473	771
family$\backslash$D$_{+100}$	with rules	954	744	mem
ODGI$\backslash$D$_{+100}$	without rules	704	1302	1658
ODGI$\backslash$D$_{+100}$	with rules	884	1456	1852

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 UOBM$_1\backslash$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 UOBM$_1\backslash$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

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

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

HasInterestAndAllFriendsSameInterest $\sqcup \exists$*isFriendOf*.$(\neg\{z\} \sqcup \neg\forall$*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:

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

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 Post-graduate Scholarships Act of the Land of Baden-Wuerttemberg (LGFG).

References

1. 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)
2. Blackburn, P., Tzakova, M.: Hybridizing concept languages. Annals of Mathematics and Artificial Intelligence 24(1–4), 23–49 (1998)
3. 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)
4. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19(1), 17–37 (1982)
5. 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 (July 2012)
6. 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)
7. 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/`
8. 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)
9. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In: McAllester, D.A. (ed.) Proc. 17th Int. Conf. on Automated Deduction (CADE'00). Lecture Notes in Computer Science, vol. 1831, pp. 482–496. Springer (2000)
10. 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)
11. 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)
12. Kifer, M., Boley, H. (eds.): RIF Overview. W3C Working Group Note (22 June 2010), available at `http://www.w3.org/TR/rif-overview/`
13. Krisnadhi, A., Hitzler, P.: A tableau algorithm for description logics with nominal schema. In: Krötzsch, M., Straccia, U. (eds.) Proc. 6th Int. Conf. on Web Reasoning and Rule Systems (RR'12). LNCS, vol. 7497, pp. 234–237. Springer (2012)
14. 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)

15. 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)
16. 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)
17. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-overview/
18. 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)
19. 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). Lecture Notes in Computer Science, vol. 7364, pp. 514–529. Springer (2012)
20. 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)

Liste der bisher erschienenen Ulmer Informatik-Berichte

Einige davon sind per FTP von `ftp.informatik.uni-ulm.de` erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm

Some of them are available by FTP from `ftp.informatik.uni-ulm.de`
Reports marked with * are out of print

91-01 *Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe*
Instance Complexity

*91-02** *K. Gladitz, H. Fassbender, H. Vogler*
Compiler-Based Implementation of Syntax-Directed Functional Programming

*91-03** *Alfons Geser*
Relative Termination

*91-04** *J. Köbler, U. Schöning, J. Toran*
Graph Isomorphism is low for PP

91-05 *Johannes Köbler, Thomas Thierauf*
Complexity Restricted Advice Functions

*91-06** *Uwe Schöning*
Recent Highlights in Structural Complexity Theory

*91-07** *F. Green, J. Köbler, J. Toran*
The Power of Middle Bit

*91-08** *V.Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf*
Reductions for Sets of Low Information Content

*92-01** *Vikraman Arvind, Johannes Köbler, Martin Mundhenk*
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

*92-02** *Thomas Noll, Heiko Vogler*
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 *Fakultät für Informatik*
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

*92-04** *V. Arvind, J. Köbler, M. Mundhenk*
Lowness and the Complexity of Sparse and Tally Descriptions

*92-05** *Johannes Köbler*
Locating P/poly Optimally in the Extended Low Hierarchy

*92-06** *Armin Kühnemann, Heiko Vogler*
Synthesized and inherited functions -a new computational model for syntax-directed semantics

*92-07** *Heinz Fassbender, Heiko Vogler*
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing

*92-08** *Uwe Schöning*
On Random Reductions from Sparse Sets to Tally Sets

*92-09** *Hermann von Hasseln, Laura Martignon*
Consistency in Stochastic Network

92-10 *Michael Schmitt*
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 *Johannes Köbler, Seinosuke Toda*
On the Power of Generalized MOD-Classes

92-12 *V. Arvind, J. Köbler, M. Mundhenk*
Reliable Reductions, High Sets and Low Sets

92-13 *Alfons Geser*
On a monotonic semantic path ordering

*92-14** *Joost Engelfriet, Heiko Vogler*
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 *Alfred Lupper, Konrad Froitzheim*
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

93-02 *M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch*
The COCOON Object Model

93-03 *Thomas Thierauf, Seinosuke Toda, Osamu Watanabe*
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 *Jin-Yi Cai, Frederic Green, Thomas Thierauf*
On the Correlation of Symmetric Functions

93-05 *K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam*
A Conceptual Approach to an Open Hospital Information System

93-06 *Klaus Gaßner*
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 *Ullrich Keßler, Peter Dadam*
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 *Michael Schmitt*
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 *Armin Kühnemann, Heiko Vogler*
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 *Harry Buhrman, Jim Kadin, Thomas Thierauf*
On Functions Computable with Nonadaptive Queries to NP

94-04 *Heinz Faßbender, Heiko Vogler, Andrea Wedel*
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman,Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-11 *Thomas Beuter, Peter Dadam*:
Prinzipien der Replikationskontrolle in verteilten Systemen

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12 *Gerhard Schellhorn, Wolfgang Reif*
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers

97-13 *Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn*
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 *Wolfgang Reif, Gerhard Schellhorn*
Theorem Proving in Large Theories

97-15 *Thomas Wennekers*
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 *Peter Dadam, Klaus Kuhn, Manfred Reichert*
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 *Mohammad Ali Livani, Jörg Kaiser*
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

97-18 *Johannes Köbler,Rainer Schuler*
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

98-01 *Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf*
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 *Thomas Bauer, Peter Dadam*
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 *Marko Luther, Martin Strecker*
A guided tour through *Typelab*

98-04 *Heiko Neumann, Luiz Pessoa*
Visual Filling-in and Surface Property Reconstruction

98-05 *Ercüment Canver*
Formal Verification of a Coordinated Atomic Action Based Design

98-06 *Andreas Küchler*
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 *Heiko Neumann, Thorsten Hansen, Luiz Pessoa*
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 *Thomas Wennekers*
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 *Thomas Bauer, Peter Dadam*
Variable Migration von Workflows in *ADEPT*

98-10 *Heiko Neumann, Wolfgang Sepp*
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 *Frank Houdek, Dietmar Ernst, Thilo Schwinn*
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12 *Gerhard Schellhorn*
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 *Gerhard Schellhorn, Wolfgang Reif*
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 *Mohammad Ali Livani*
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 *Mohammad Ali Livani, Jörg Kaiser*
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01 *Susanne Boll, Wolfgang Klas, Utz Westermann*
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02 *Thomas Bauer, Peter Dadam*
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03 *Uwe Schöning*
On the Complexity of Constraint Satisfaction

99-04 *Ercument Canver*
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05 *Johannes Köbler, Wolfgang Lindner, Rainer Schuler*
Derandomizing RP if Boolean Circuits are not Learnable

99-06 *Utz Westermann, Wolfgang Klas*
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07 *Peter Dadam, Manfred Reichert*
Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999, GI–Workshop Proceedings, Informatik ’99

99-08 *Vikraman Arvind, Johannes Köbler*
Graph Isomorphism is Low for ZPP^{NP} and other Lowness results

99-09 *Thomas Bauer, Peter Dadam*
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02 *Thomas Bauer, Peter Dadam*
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03 *Gregory Baratoff, Christian Toepfer, Heiko Neumann*
Combined space-variant maps for optical flow based navigation

2000-04 *Wolfgang Gehring*
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05 *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06 *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07 *Gerhard Schellhorn, Wolfgang Reif (eds.)*
FM-Tools 2000: The 4th Workshop on Tools for System Design and Verification

2000-08 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09 *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11 *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01 *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02 *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03 *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04 *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02 *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03 *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01 *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values

2003-04 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
On Dealing With Semantically Conflicting Business Process Changes.

2003-05 *Christian Heinlein*
Dynamic Class Methods in Java

2003-06 *Christian Heinlein*
Vertical, Horizontal, and Behavioural Extensibility of Software Systems

2003-07 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values
(Corrected Version)

2003-08 *Changling Liu, Jörg Kaiser*
Survey of Mobile Ad Hoc Network Routing Protocols)

2004-01 *Thom Frühwirth, Marc Meister (eds.)*
First Workshop on Constraint Handling Rules

2004-02 *Christian Heinlein*
Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures

2004-03 *Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)*
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence

2005-01 *Armin Wolf, Thom Frühwirth, Marc Meister (eds.)*
19th Workshop on (Constraint) Logic Programming

2005-02 *Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven*
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm

2005-03 *Walter Guttmann, Markus Maucher*
Constrained Ordering

2006-01 *Stefan Sarstedt*
Model-Driven Development with ACTIVECHARTS, Tutorial

2006-02 *Alexander Raschke, Ramin Tavakoli Kolagari*
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen

2006-03 *Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari*
Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg

2006-04 *Thorsten Liebig*
Reasoning with OWL - System Support and Insights –

2008-01 *H.A. Kestler, J. Messner, A. Müller, R. Schuler*
On the complexity of intersecting multiple circles for graphical display

2008-02 *Manfred Reichert, Peter Dadam, Martin Jurisch,l Ulrich Kreher, Kevin Göser, Markus Lauer*
Architectural Design of Flexible Process Management Technology

2008-03 *Frank Raiser*
Semi-Automatic Generation of CHR Solvers from Global Constraint Automata

2008-04 *Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander*
Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse

2008-05 *Markus Kalb, Claudia Dittrich, Peter Dadam*
Support of Relationships Among Moving Objects on Networks

2008-06 *Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)*
WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke

2008-07 *M. Maucher, U. Schöning, H.A. Kestler*
An empirical assessment of local and population based search methods with different degrees of pseudorandomness

2008-08 *Henning Wunderlich*
Covers have structure

2008-09 *Karl-Heinz Niggl, Henning Wunderlich*
Implicit characterization of FPTIME and NC revisited

2008-10 *Henning Wunderlich*
On span-P^{cc} and related classes in structural communication complexity

2008-11 *M. Maucher, U. Schöning, H.A. Kestler*
On the different notions of pseudorandomness

2008-12 *Henning Wunderlich*
On Toda's Theorem in structural communication complexity

2008-13 *Manfred Reichert, Peter Dadam*
Realizing Adaptive Process-aware Information Systems with ADEPT2

2009-01 *Peter Dadam, Manfred Reichert*
The ADEPT Project: A Decade of Research and Development for Robust and Fexible Process Support
Challenges and Achievements

2009-02 *Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, Martin Jurisch*
Von ADEPT zur AristaFlow® BPM Suite – Eine Vision wird Realität "Correctness by Construction" und flexible, robuste Ausführung von Unternehmensprozessen

2009-03 *Alena Hallerbach, Thomas Bauer, Manfred Reichert*
Correct Configuration of Process Variants in Provop

2009-04 *Martin Bader*
On Reversal and Transposition Medians

2009-05 *Barbara Weber, Andreas Lanz, Manfred Reichert*
Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

2009-06 *Stefanie Rinderle-Ma, Manfred Reichert*
Adjustment Strategies for Non-Compliant Process Instances

2009-07 *H.A. Kestler, B. Lausen, H. Binder H.-P. Klenk. F. Leisch, M. Schmid*
Statistical Computing 2009 – Abstracts der 41. Arbeitstagung

2009-08 *Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam*
Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen

2009-09 *Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch*
Progressive Point-Light-Based Global Illumination

2009-10 *Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler*
Boolean networks for modeling and analysis of gene regulation

2009-11 *J. Hanika, H.P.A. Lensch, A. Keller*
Two-Level Ray Tracing with Recordering for Highly Complex Scenes

2009-12 *Stephan Buchwald, Thomas Bauer, Manfred Reichert*
Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

2010-01 *Hariolf Betz, Frank Raiser, Thom Frühwirth*
A Complete and Terminating Execution Model for Constraint Handling Rules

2010-02 *Ulrich Kreher, Manfred Reichert*
Speichereffiziente Repräsentation instanzspezifischer Änderungen in Prozess-Management-Systemen

2010-03 *Patrick Frey*
Case Study: Engine Control Application

2010-04 *Matthias Lohrmann und Manfred Reichert*
Basic Considerations on Business Process Quality

2010-05 *HA Kestler, H Binder, B Lausen, H-P Klenk, M Schmid, F Leisch (eds):*
Statistical Computing 2010 - Abstracts der 42. Arbeitstagung

2010-06 *Vera Künzle, Barbara Weber, Manfred Reichert*
Object-aware Business Processes: Properties, Requirements, Existing Approaches

2011-01 *Stephan Buchwald, Thomas Bauer, Manfred Reichert*
Flexibilisierung Service-orientierter Architekturen

2011-02 *Johannes Hanika, Holger Dammertz, Hendrik Lensch*
Edge-Optimized À-Trous Wavelets for Local Contrast Enhancement with Robust Denoising

2011-03 *Stefanie Kaiser, Manfred Reichert*
Datenflussvarianten in Prozessmodellen: Szenarien, Herausforderungen, Ansätze

2011-04 *Hans A. Kestler, Harald Binder, Matthias Schmid, Friedrich Leisch, Johann M. Kraus (eds):*
Statistical Computing 2011 - Abstracts der 43. Arbeitstagung

2011-05 *Vera Künzle, Manfred Reichert*
PHILharmonicFlows: Research and Design Methodology

2011-06 *David Knuplesch, Manfred Reichert*
Ensuring Business Process Compliance Along the Process Life Cycle

2011-07 *Marcel Dausend*
Towards a UML Profile on Formal Semantics for Modeling Multimodal Interactive Systems

2011-08 *Dominik Gessenharter*
Model-Driven Software Development with ACTIVECHARTS - A Case Study

2012-01 *Andreas Steigmiller, Thorsten Liebig, Birte Glimm*
Extended Caching, Backjumping and Merging for Expressive Description Logics

2012-02 *Hans A. Kestler, Harald Binder, Matthias Schmid, Johann M. Kraus (eds):*
Statistical Computing 2012 - Abstracts der 44. Arbeitstagung

2012-03 *Felix Schüssel, Frank Honold, Michael Weber*
Influencing Factors on Multimodal Interaction at Selection Tasks

2012-04 *Jens Kolb, Paul Hübner, Manfred Reichert*
Model-Driven User Interface Generation and Adaption in Process-Aware Information Systems

2012-05 *Matthias Lohrmann, Manfred Reichert*
Formalizing Concepts for Efficacy-aware Business Process Modeling*

2012-06 *David Knuplesch, Rüdiger Pryss, Manfred Reichert*
A Formal Framework for Data-Aware Process Interaction Models

2013-01 *Frank Kargl*
Abstract Proceedings of the 7th Workshop on Wireless and Mobile Ad-Hoc Networks (WMAN 2013)

2013-02 *Andreas Lanz, Manfred Reichert, Barbara Weber*
A Formal Semantics of Time Patterns for Process-aware Information Systems

2013-03 *Matthias Lohrmann, Manfred Reichert*
Demonstrating the Effectiveness of Process Improvement Patterns with Mining Results

2013-04 *Semra Catalkaya, David Knuplesch, Manfred Reichert*
Bringing More Semantics to XOR-Split Gateways in Business Process Models Based on Decision Rules

2013-05 *David Knuplesch, Manfred Reichert, Linh Thao Ly, Akhil Kumar, Stefanie Rinderle-Ma*
On the Formal Semantics of the Extended Compliance Rule Graph

2013-06 *Andreas Steigmiller, Birte Glimm, Thorsten Liebig*
Nominal Schema Absorption

Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm