

# The Summary Abox: Cutting Ontologies Down to Size

Achille Fokoue, Aaron Kershenbaum, Edith Schonberg, Kavitha Srinivas  
IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA  
achille, aaronk, ediths, ksrinivs@us.ibm.com

Li Ma  
IBM China Research Lab, Beijing 100094, China  
malli@cn.ibm.com

## Abstract

*Reasoning in descriptive logic (DL) ontologies is known to be intractable in the worst-case. This poses a serious problem for the use of DL-based ontologies such as OWL, because in practice, most ontologies have numerous assertions about individuals in the Abox. We propose a technique to make reasoning scalable for very large Aboxes in secondary storage. The technique operates by producing a dramatically reduced summary Abox for the original Abox. Reasoning on the original Abox is reduced in most cases to reasoning on this reduced summary Abox. We show that our techniques scale to consistency detection on ontologies with 6.5 million role assertions.*

**Keywords:** Semantic web ontologies

## 1 Introduction

Description Logic (DL) provides the theoretical foundation for semantic web ontologies (OWL). A DL ontology can be divided conceptually into three components: the Tbox, the Rbox and the Abox. The Tbox contains assertions about concepts such as subsumption ( $Man \sqsubseteq Person$ ) and equivalence ( $Man \equiv MaleHuman$ ). The Rbox contains assertions about roles and role hierarchies ( $hasSon \sqsubseteq hasChild$ ). The Abox contains role assertions between individuals ( $hasChild(John, Mary)$ ) and membership assertions ( $John : Man$ ).

All common reasoning tasks in expressive DL ontologies, such as query answering [11], reduce to consistency detection. As an example, a standard approach to testing if *John* is a member of the concept *Man* requires testing if the addition of the assertion ( $John : \neg Man$ ) makes the Abox

inconsistent using the tableau algorithm [1]. A challenge is that consistency detection in expressive DL is well known to be intractable in the worst-case [3]. Given that the size of an Abox may be in the order of millions of assertions, this complexity poses a serious problem for the practical use of DL ontologies. Furthermore, large Aboxes often reside in transactional databases, and are prone to frequent changes. Although highly optimized tableau algorithms exist for consistency detection in DL, they cannot be easily adapted to large Aboxes in secondary storage, especially for frequently changing Aboxes. One approach that has been applied to reasoning on Aboxes in secondary storage is to convert DL to disjunctive datalog, and use deductive databases to reason over the Abox [16].

We propose an alternative technique that operates on Aboxes stored in traditional relational database systems. Our technique exploits a key observation about real world Aboxes, namely, similar individuals are related to other individuals in similar ways (e.g. fathers and mothers are related to each other by the *hasSpouse* role, and are related to their children by the *hasChild* role). Specifically, our technique builds a summary Abox  $\mathcal{A}'$  of the original Abox  $\mathcal{A}$ , by aggregating similar individuals and assertions. The advantages of our summary Abox  $\mathcal{A}'$  are: (a)  $\mathcal{A}'$  is dramatically smaller than  $\mathcal{A}$ ; (b) Reasoning on  $\mathcal{A}'$  isolates a small relevant portion of  $\mathcal{A}$  needed to obtain the correct answer; (c)  $\mathcal{A}'$  can be computed efficiently using straightforward relational database queries; (d)  $\mathcal{A}'$  can be maintained as changes occur to  $\mathcal{A}$ , and is thus resilient to change; (e)  $\mathcal{A}'$  only needs to be computed once, and can be reused for answering subsequent queries.

To isolate relevant portions of  $\mathcal{A}$  for a specific reasoning task, we introduce efficient filtering techniques that operate on  $\mathcal{A}'$ . In this paper, we demonstrate the utility of such filtering techniques for the task of Abox consistency detection, although the approach can be generalized to query answering. Our filtering techniques are based on the conser-

vative assumption that any individual in the Abox may be inferred to be a member of any concept in the closure of the Abox, i.e., the *complete reachability assumption*. Informally, the closure of the Abox is the set of concepts, and their sub-expressions that may be present in the Abox. (To generalize this to query answering, the closure would include the negated concept in the query.) With this assumption, we show that in practice, we can filter a large number of role assertions in the summary Abox, because they are in effect *irrelevant* for consistency detection. The effect of such filtering is to produce multiple partitions in the summary Abox, where most partitions consist of a single individual in practice. Consistency check for these partitions reduces to a concept satisfiability test, which can be performed using any existing reasoner. For partitions of  $\mathcal{A}'$  with multiple individuals, we perform a consistency check of the partition. If the partition is consistent, then the image in  $\mathcal{A}$  that corresponds to the partition is also consistent, which means that the consistency check for the original Abox is reduced to a check on the summary. If the partition is inconsistent, this inconsistency could arise either due to the summarization technique, or due a real inconsistency in the original Abox. We retrieve the image in  $\mathcal{A}$  of the inconsistent partition, and compute a consistency check on this image.

Our techniques proved very effective on the 4 large Aboxes that we studied: a vast majority of partitions (95%) had just one individual, and were therefore reduced to concept satisfiability checks. Only one of the 4 ontologies we studied required us to check the image of the the partition in the original Abox. Even in this case, our consistency check had to be performed on 4045 individuals and 2942 role assertions instead of the 1,106,858 individuals and 6,494,950 assertions in the entire Abox.

Our key contributions in this paper are as follows: (a) We present a technique to summarize an Abox in secondary storage into a smaller  $\mathcal{A}'$ . (b) We describe the use of filtering techniques to construct a dramatically reduced version of  $\mathcal{A}'$ . This filtering produces many partitions, which are then exploited in scaling the consistency check. The filtering techniques we describe works for SHIN Aboxes (SHIN is a DL language that is described in the Background). (c) We show the application of these techniques to 4 ontologies (the largest of which had 6.5 million assertions), where we show dramatic reductions in space and time requirements for consistency checking.

## 1.1 Background

The techniques we apply in this paper assume ontologies of SHIN expressiveness. In this section, we briefly introduce the semantics of SHIN, which is equivalent to OWL-DL (<http://www.w3.org/2001/sw/WebOnt>) minus nominals

**Table 1. SHIN Description Logic**

Definitions	Semantics
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\exists R.C$	$\{x \mid \exists y. \langle x, y \rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$
$\forall R.C$	$\{x \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
$\leq nR$	$\{x \mid  \langle x, y \rangle \in R^{\mathcal{I}}  \leq n\}$
$\geq nR$	$\{x \mid  \langle x, y \rangle \in R^{\mathcal{I}}  \geq n\}$
$R^-$	$\{\langle x, y \rangle \mid \langle y, x \rangle \in R^{\mathcal{I}}\}$
Axioms	Satisfiability conditions
$\text{Trans}(R)$	$(R^{\mathcal{I}})^+ = R^{\mathcal{I}}$
$R \sqsubseteq P$	$\langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow \langle x, y \rangle \in P^{\mathcal{I}}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$R(a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

and datatype reasoning, as shown in Table 1 (We assume the reader is familiar with Description Logics). In the definition of the semantics of SHIN,  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  refers to an interpretation where  $\Delta^{\mathcal{I}}$  is a non-empty set (the domain of the interpretation), and  $\cdot^{\mathcal{I}}$ , the interpretation function, maps every atomic concept  $C$  to a set  $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ , every atomic role  $R$  to a binary relation  $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ , and every individual  $a$  to  $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ .  $\text{Trans}(R)$  in the table refers to a transitive role  $R$ . An RBox  $\mathcal{R}$  is a finite set of transitivity axioms of the form  $\text{Trans}(R)$  and role inclusion axioms of the form  $R \sqsubseteq P$  where  $R$  and  $P$  are roles.  $\sqsubseteq^*$  denotes the reflexive transitive closure of the  $\sqsubseteq$  relation on roles. A Tbox  $\mathcal{T}$  is a set of concept inclusion axioms of the form  $C \sqsubseteq D$  where  $C$  and  $D$  are concept expressions. Finally, an Abox  $\mathcal{A}$  is a set of axioms of the form  $a : C$  ( $a$  is a member of the concept  $C$ ),  $R(a, b)$  (there is a  $R$  relationship between  $a$  and  $b$ ) and  $a \neq b$  ( $a$  is different from  $b$ ).

An interpretation  $\mathcal{I}$  is a model of an Abox  $\mathcal{A}$  w.r.t. a Tbox  $\mathcal{T}$  and a Rbox  $\mathcal{R}$  iff it satisfies all the axioms in  $\mathcal{A}$ ,  $\mathcal{R}$ , and  $\mathcal{T}$  (The necessary and sufficient conditions for satisfiability of axioms are given in Table 1). An Abox  $\mathcal{A}$  is said to be consistent w.r.t. a Tbox  $\mathcal{T}$  and a Rbox  $\mathcal{R}$  iff there is a model of  $\mathcal{A}$  w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$ . If there is no ambiguity from the context, we simply say that  $\mathcal{A}$  is consistent.

A standard technique for checking the consistency of a SHIN Abox  $\mathcal{A}$  is to use a tableau algorithm [10], which executes a set of non-deterministic expansion rules to satisfy constraints in  $\mathcal{A}$  until either no rule is applicable or an obvious inconsistency, a clash, is detected. These expansion rules are discussed in a later section.

## 2 Summary Abox

Intuitively, the Abox contains many redundant assertions from the point of view of consistency checking that can be collapsed to create a reduced *summary Abox*. The summary Abox captures this redundancy by collapsing across individuals that are members of the same concepts as shown in Figures 1 and 2 below. As shown in Figure 2, a single node  $a$  represents  $a1$  and  $a2$  because they are both members of  $A$ , and they are not explicitly asserted to be different from each other (similarly for  $b$  and  $d$ ). Any explicit assertions that two individuals are different from each other ( $c1$  and  $c2$ ) are maintained in the summary Abox. Reasoning over such a summary corresponds to reasoning over the original Abox, as shown formally below.

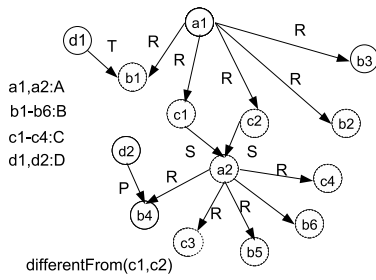


Figure 1. Original Abox

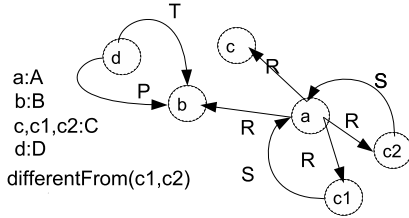


Figure 2. Canonical Summary Abox

**Definition 1:** A summary Abox is an Abox  $\mathcal{A}'$  that is generated from any SHIN Abox  $\mathcal{A}$  using a mapping function  $f$  that satisfies the following constraints:

- (1) if  $a:C \in \mathcal{A}$  then  $f(a):C \in \mathcal{A}'$
- (2) if  $R(a,b) \in \mathcal{A}$  then  $R(f(a), f(b)) \in \mathcal{A}'$
- (3) if  $a \neq b \in \mathcal{A}$  then  $f(a) \neq f(b) \in \mathcal{A}'$

**Theorem 2:** If the summary Abox  $\mathcal{A}'$  obtained by applying the mapping function  $f$  to  $\mathcal{A}$  is consistent w.r.t. a Tbox  $\mathcal{T}$  and a Rbox  $\mathcal{R}$ , then  $\mathcal{A}$  is consistent w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$ . However, the converse of Theorem 2 does not hold.

**Proof:** Let us assume that  $\mathcal{A}'$  is consistent w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$ . Therefore there is a model  $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$  of  $\mathcal{A}'$  w.r.t.  $\mathcal{T}$

and  $\mathcal{R}$ . A model of  $\mathcal{A}$  can easily be built from  $\mathcal{I}'$  by interpreting an individual  $a$  in  $\mathcal{A}$  in the same way as  $f(a)$  is interpreted by  $\mathcal{I}'$ . Formally, let  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  be the interpretation of the  $\mathcal{A}$  w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$  defined as follows:  $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ ; for a concept  $C \in \mathcal{T}$ ,  $C^{\mathcal{I}} = C^{\mathcal{I}'}$ ; for a role  $R$  in  $\mathcal{R}$ ,  $R^{\mathcal{I}} = R^{\mathcal{I}'}$ ; for an individual  $a$  in  $\mathcal{A}$ ,  $a^{\mathcal{I}} = f(a)^{\mathcal{I}'}$ .

$\mathcal{I}$  is a model of  $\mathcal{A}$  w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$  as a direct consequence of the fact that  $\mathcal{I}'$  is a model of  $\mathcal{A}'$  and  $\mathcal{A}'$  satisfies the 3 conditions stated in definition 1. The first 3 types of axioms in Table 1 are obviously satisfied by  $\mathcal{I}$  because there are satisfied by  $\mathcal{I}'$ . Now, let us assume that  $a:C \in \mathcal{A}$ , then, by definition of  $\mathcal{A}'$ ,  $f(a):C \in \mathcal{A}'$ . Since  $\mathcal{I}'$  is a model of  $\mathcal{A}'$ ,  $f(a)^{\mathcal{I}'} \in C^{\mathcal{I}'}$ , which, by definition of  $\mathcal{I}$ , implies that  $a^{\mathcal{I}} \in C^{\mathcal{I}}$ . This establishes that axioms of the forms  $a:C \in \mathcal{A}$  are satisfied by  $\mathcal{I}$ . Likewise,  $\mathcal{I}$  satisfies the last two types of axioms presented in Table 1. ■

Let  $\mathcal{L}$  be a mapping from each individual in  $\mathcal{A}$  to a set of concepts, such that  $a:C \in \mathcal{A}$  iff  $C \in \mathcal{L}(a)$ . We call  $\mathcal{L}(a)$  the *concept set* of  $a$ . In practice, we use a *canonical function*  $f$  to create a summary Abox, which maps non-distinct individuals that have identical concept sets to the same individual in  $\mathcal{A}'$ . More formally, the canonical summary Abox has the following additional properties:

- (4) if  $f(a):C \in \mathcal{A}'$  then  $a:C \in \mathcal{A}$
- (5) if  $R(a', b') \in \mathcal{A}'$  then there are  $a$  and  $b$  in  $\mathcal{A}$  such that  $a' = f(a)$ ,  $b' = f(b)$  and  $R(a, b) \in \mathcal{A}$ .
- (6) if for any individual  $x \in \mathcal{A}$ ,  $a \neq x \notin \mathcal{A}$ ,  $b \neq x \notin \mathcal{A}$ , and  $\mathcal{L}(a) = \mathcal{L}(b)$ , then  $f(a) = f(b)$ .
- (7) if  $f(a) \neq f(b) \in \mathcal{A}'$  then  $a \neq b \in \mathcal{A}$ , and  $a$  is the only individual in  $\mathcal{A}$  mapped to  $f(a)$  (similarly for  $b$ ).

If a summary Abox  $\mathcal{A}'$  is not consistent, either there is a real inconsistency in  $\mathcal{A}$  or the process of collapsing individuals to create  $\mathcal{A}'$  caused an artificial inconsistency. In section 3.4, we explain how filtering and partitioning techniques applied to the summary can provide a scalable consistency check of the original Abox even when its summary is not consistent. Note that the summary graph can be computed efficiently from a relational database. Furthermore, it only needs to be computed once, since it can be maintained incrementally with changes to the Abox.

## 3 Abox Filtering

We perform filtering on the canonical summary Abox described in Section 2, but for the purpose of exposition, we describe filtering techniques on the original Abox first<sup>1</sup>. Our filtering technique assumes that the Tbox  $\mathcal{T}$  is transformed,

<sup>1</sup>If the goal is solely to check consistency, then we could apply filtering directly to the original Abox.

through splitting and absorption [12], into two disjoint sets  $\mathcal{T}_u$  and  $\mathcal{T}_g$  such that  $\mathcal{T}_u$ , the unfoldable part of  $\mathcal{T}$ , only contains axioms of the form  $A \sqsubseteq D$  and  $\neg A \sqsubseteq D$ , where  $A$  is an atomic concept.  $\mathcal{T}_g$  contains general concept inclusions (GCIs) of the form  $C \sqsubseteq D$  that could not be absorbed in  $\mathcal{T}_u$  (where  $C$  is a complex concept).

We assume that for an Abox  $\mathcal{A}$ , an Rbox  $\mathcal{R}$ , and a Tbox  $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ , all concepts appearing in  $\mathcal{T}$  and  $\mathcal{A}$  are in the negation normal form (NNF). For a concept expression  $C$  in NNF,  $\text{clos}(C, \mathcal{T}, \mathcal{R})$  is the smallest set  $X$  containing  $C$ , closed under concept sub-expression, such that, for an atomic concept  $A$ , (1) if  $A \in X$  and  $A \sqsubseteq D \in \mathcal{T}_u$ , then  $D \in X$ , (2) if  $\neg A \in X$  and  $\neg A \sqsubseteq D \in \mathcal{T}_u$ , then  $D \in X$ , and (3) if  $\forall P.C \in X$  and there is a role  $R$  with  $R \sqsubseteq^* P$  and  $\text{Trans}(R)$ , then  $\forall R.C \in X$ . Formally, we define the closure of  $\mathcal{A}$  w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$ , denoted  $\text{clos}(\mathcal{A}, \mathcal{T}, \mathcal{R})$ , as  $\bigcup_{\alpha C \in \mathcal{A}} \text{clos}(C, \mathcal{T}, \mathcal{R}) \cup \bigcup_{C \sqsubseteq D \in \mathcal{T}_g} \text{clos}(\text{NNF}(\neg C) \sqcup D, \mathcal{T}, \mathcal{R})$ . When there is no ambiguity, we use  $\text{clos}(\mathcal{A})$  instead of  $\text{clos}(\mathcal{A}, \mathcal{T}, \mathcal{R})$ .

### 3.1 Motivation

To provide an intuition for our criteria for filtering role assertions, we first describe a set of tableau expansion rules, which correspond to a subset of expansion rules defined in [10]. These rules are simplified for explication purposes (e.g. blocking is not considered and the details of merging are not provided).

We assume that  $a$  and  $b$  are named individuals in  $\mathcal{A}$ ,  $x$  is an unnamed individual,  $C$  is a concept in  $\text{clos}(\mathcal{A})$ , and  $R$  is a role. Named individuals are in  $\mathcal{A}$  before applying any expansion rules, while unnamed individuals are introduced as a result of expansion rules. An individual  $b$  is defined to be an  $R$ -neighbor of  $a$  iff there is an assertion  $Q(a, b)$  or  $Q^-(b, a)$  in  $\mathcal{A}$  where  $Q \sqsubseteq^* R$ .

$\forall$ -rule:  $a : (\forall R.C) \in \mathcal{A}$ ,  $b$  is an  $R$ -neighbor of  $a$ , and  $b : C \notin \mathcal{A} \Rightarrow$  add  $b : C$  to  $\mathcal{A}$ .

$\leq$ -rule:  $a : (\leq nR) \in \mathcal{A}$  and, for  $1 \leq i \leq m$ ,  $m > n$ ,  $b_i$  is an  $R$ -neighbor of  $a$ , and for two of these  $b_k$  and  $b_j$  the assertion  $b_k \neq b_j$  is not in  $\mathcal{A} \Rightarrow$

- (i) Merge (identify)  $b_j$  with  $b_k$  (the precise rules to determine which of  $b_j$  or  $b_k$  is selected is detailed in [10].)
- (ii) add the assertions in  $\mathcal{L}(b_j)$  to  $\mathcal{L}(b_k)$
- (iii) for every role assertion with  $b_j$ , replace  $b_j$  with  $b_k$ . In effect, this step adds new role assertions to the  $\mathcal{A}$ .

$\exists$ -rule:  $a : (\exists R.C) \in \mathcal{A}$  and for all  $R$ -neighbors  $b$  of  $a$ ,  $b : C \notin \mathcal{A} \Rightarrow$  add  $R(a, x)$  and  $x : C$  to  $\mathcal{A}$ .

$\geq$ -rule:  $a : (\geq nR) \in \mathcal{A}$  and  $a$  does not have  $n$  distinct  $R$ -neighbors  $\Rightarrow$  add  $R(a, x_i)$  to  $\mathcal{A}$ ,  $1 \leq i \leq n$  where the  $x_i$  are new distinct unnamed individuals.

$\forall_+$ -rule:  $a : \forall R.C \in \mathcal{A}$ ,  $P \sqsubseteq^* R$  is transitive,  $b$  is a  $P$ -neighbor of  $a$ , and  $b : (\forall P.C) \notin \mathcal{A} \Rightarrow$  add  $b : (\forall P.C)$  to  $\mathcal{A}$ .

We make the key observation that role assertions of the form  $R(a, b)$  may affect the outcome of an execution of the tableau algorithm only if one of the following two conditions holds:

1. They can be used to trigger the application of tableau rules that alter the Abox. As an example of such alteration, a role assertion can be used to add new membership assertions about named individuals (e.g., a new concept  $C$  can be propagated to  $b$ 's concept set through a role assertion  $R(a, b)$  by the application of the  $\forall$  rule on  $a$  if  $a : (\forall R.C) \in \mathcal{A}$ , and  $b$  is an  $R$ -neighbor of  $a$ ).
2. They can be involved in clash detection due to a violation of a maximum cardinality restriction. As an example, if  $a : (\leq nR)$  is in the Abox and  $b$  is one of  $n + 1$  mutually distinct  $R$ -neighbors of  $a$ , then  $R(a, b)$  is important for clash detection.

These conditions can *only* be brought about by either the application of the  $\forall$ ,  $\leq$ , and  $\forall_+$ -rules or the presence of a maximum cardinality constraint. In contrast, the  $\exists$ -rule and  $\geq$ -rule do not use existing role assertions  $R(a, b)$ ; instead, they result in the creation of new role assertions and new unnamed individuals for satisfying the  $\exists R.C$  and  $\geq nR$  constraints.

### 3.2 Criteria for Filtering Role Assertions

Our filtering criteria guarantee that the absence of a role assertion will not affect the outcome of any execution of the non-deterministic tableau algorithm. Our goal is to define criteria which are efficient to evaluate using simple queries against relational databases, while balancing the tradeoff between filtering precision and cost. For instance, by assuming any concept in the  $\text{clos}(\mathcal{A})$  can reach the concept set of any node in the Abox during any execution of the tableau algorithm, we avoid tableau operations which are expensive in relational databases. We will say that a role  $R$  is *part of a universal restriction*  $\forall P.C$  iff  $R \sqsubseteq^* P$ . (Similarly for maximum cardinality restriction).

To filter a role assertion  $R(a, b)$ ,  $\mathcal{A}$  must satisfy either 1) or both 2) and 3):

**1) Absence of universal and maximum cardinality restrictions:** We make the simple observation that if a role  $R$  and its inverse  $R^-$  are not part of any universal or maximum cardinality restrictions, then  $R(a, b)$  can never be used to alter the Abox or detect a clash, so it can be ignored.

**2) Absence of universal rules triggering:** Even if  $R$  is part of a universal restriction, it may never trigger the application of the universal rules ( $\forall_+$ ,  $\forall$ ). We define the conditions under which we can guarantee that the universal rules will never be triggered as follows. If  $R$  (resp.  $R^-$ ) is part of a

universal restriction  $\forall P.C$  in  $clos(\mathcal{A})$ , then  $R(a, b)$  is *irrelevant with respect to  $\forall P.C$*  if  $b : C \in \mathcal{A}$  (resp.  $a : C \in \mathcal{A}$ ) and  $R$  (resp.  $R^-$ ) has no transitive superroles.

To satisfy the filtering condition,  $R(a, b)$  must be irrelevant with respect to all universal restrictions  $\forall P.C$  in  $clos(\mathcal{A})$ , where  $R$  or  $R^-$  is part of  $\forall P.C$ .

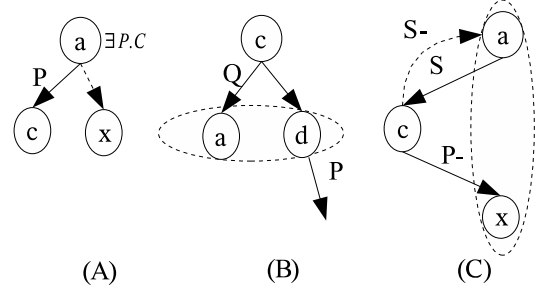
**3) Absence of maximum cardinality restrictions triggering:** For the  $\leq$ -rule to be triggered, there needs to be a violation of a maximum cardinality constraint  $a : \leq nP$ , where the individual  $a$  has more than  $n$   $P$ -neighbors, which then causes a merger between individuals. We introduce a technique to conservatively estimate an upper bound on  $P$ -neighbors, such that at any step of any possible execution of tableau algorithm, the number of  $P$ -neighbors of  $a$  is less than or equal to this upper bound. If  $R$  (resp.  $R^-$ ) is part of a maximum cardinality restriction  $\leq nP$  in  $clos(\mathcal{A})$ , then  $R(a, b)$  is *irrelevant with respect to  $\leq nP$*  if the upper bound on the number of  $P$ -neighbors of  $a$  (resp.  $b$ ) is less than  $n$ . Note that this also guarantees that no clash can occur from the presence of the role assertion.

To satisfy the filtering condition,  $R(a, b)$  must be irrelevant with respect to all maximum cardinality restrictions  $\leq nP$  in  $clos(\mathcal{A})$ , where  $R$  or  $R^-$  is part of  $\leq nP$ .

**Upper bound on the number of  $P$ -neighbors.** Unfortunately, in expressive logics such as SHIN, computing an upper bound does not simply involve counting the number of explicit  $P$ -neighbors of  $a$  that are present in the  $\mathcal{A}$ . Figure 3 shows examples of the three ways that an individual  $a$  can acquire a new  $P$ -neighbor during the execution of the tableau algorithm:

- 3(A) Individual  $a$  acquires a new  $P$ -neighbor  $x$ , where  $x$  is an unnamed individual, to satisfy  $\exists P.C$  in  $a$ 's concept set.
- 3(B) Individual  $a$  is merged with a named individual  $d$  and acquires a new  $P$ -neighbor in order to satisfy a maximum cardinality restriction  $c : \leq nQ$ .
- 3(C) Individual  $a$  is merged with an unnamed individual  $x$  and acquires a new  $P$ -neighbor  $c$ , where  $c$  is either a named or unnamed individual. This occurs in this example because of two conditions: (i) there is a role  $S^-$  that is *attracted* to  $P^-$  because a common super role  $Q$  is part of a maximum cardinality restriction and (ii) a role generator of the form  $\exists T.B$  or  $\geq mT$  is in the concept set of  $c$ , where  $T \sqsubseteq^* P^-$ .

Accounting for  $P$ -neighbors acquired through situations like 3(B) and 3(C) is not obvious. Therefore we define sufficient conditions under which these situations cannot occur, so that an upper bound of  $a$ 's  $P$ -neighbors can be computed safely and efficiently. If any of these conditions are violated, then a merger of  $a$  may result in an increase of its number of  $P$ -neighbors, and hence we do not filter  $R(a, b)$ :



**Figure 3. Acquisition of  $P$ -neighbors**

(C1)  $P$  is safe in  $\mathcal{A}$ .

Intuitively, the notion of safety ensures that a merger of a named individual  $a$  with an unnamed individual  $x$  that would increase the number of  $P$ -neighbors of  $a$ , as illustrated in 3(C), cannot occur. If  $P$  is safe, then either condition (i) or (ii) in 3(C) must be false. More generally, for a given role  $P$ , we say that  $T$  belongs to the set  $attractant(P)$  iff there is a role  $Q$  such that  $P \sqsubseteq^* Q$ ,  $T \sqsubseteq^* Q$ , and  $\leq nQ \in clos(\mathcal{A})$ . A role  $P$  is *safe* if one of the two conditions hold: (a)  $attractant(P) \subseteq \{P\}$  and  $attractant(P^-) \subseteq \{P^-\}$  (b) For all subroles  $Q$  of  $P$  or  $P^-$  there are no  $Q$ -generators (i.e.  $\geq mQ$  or  $\exists Q.C$ ) in  $clos(\mathcal{A})$ .

(C2) For any role  $Q$ , if  $a$  is a  $Q$ -neighbor of some named individual  $c$  then there is no concept of the form  $(\leq nQ)$  in  $clos(\mathcal{A})$ .

(C3) For any role  $S$ , if some named individual  $c$  is a  $S$ -neighbor of  $a$  and  $\leq nS$  is in  $clos(\mathcal{A})$ , then  $S$  is safe in  $\mathcal{A}$ .

Conditions (C2) and (C3) ensure that a merger of  $a$  and a named individual as illustrated in 3(B) is impossible. (C2) by itself is not sufficient because, even if  $Q$  is not part of a maximum cardinality restriction,  $Q^-$  may have an attractant  $T^-$ , where  $\exists T^- B$  is in the concept set of  $a$ . As described in 3(C), these conditions can cause a merger between  $c$  and an unnamed individual  $x$ , so that  $a$  becomes a  $T$ -neighbor of  $c$ . If  $T$  is part of a maximum cardinality restriction,  $a$  itself may become mergable. Condition (C3) prevents mergers between  $c$  and unnamed individuals, thus preventing  $a$  from becoming mergable.

If  $a$  and  $P$  satisfy (C1), (C2) and (C3), an upper bound on the number of  $P$ -neighbors can be computed using the following formula:

$$|P(a)| + |Some(P, a)| + \sum_{\geq mP \in Min(P, a)} m$$

where before the application of any tableau rules,

- $|P(a)|$  denotes the number of  $P$ -neighbors of  $a$
- $Some(P, a) = \{\exists P.C \in clos(\mathcal{A}) \mid \text{there is no } P\text{-neighbor } d \text{ of } a \text{ such that } d : C \in \mathcal{A}\}$
- $Min(P, a) = \{\geq mP \in clos(\mathcal{A}) \mid \text{there are no individuals } d_i \text{ such that, for } 1 \leq i \leq m, d_i \text{ is a } P\text{-neighbors of } a, \text{ and if } j \neq k, \text{ then } d_k \neq d_j \in \mathcal{A}\}$

Intuitively, the upper bound is the sum of the explicit  $P$ -neighbors of  $a$  before the application of tableau rules, plus the maximum number of unnamed individuals that can be generated by the application of the  $\exists$ - and  $\geq$ -rules, excluding any existential or minimum cardinality restrictions that are already satisfied prior to the application of tableau rules.

### 3.3 Correctness of filtering criteria

Since some of the notions introduced in the previous section are defined in the context of the tableau algorithm, we first briefly present some important concepts related to this algorithm that are used throughout this section. As described in [10], the tableau algorithm operates on completion forest  $F = (G, \mathcal{L}, \dot{=}, \dot{\neq})$  where  $G$  is graph;  $\mathcal{L}$  is a mapping from a node  $x$  in  $F$  to a set of concepts,  $\mathcal{L}(x)$ , in  $clos(\mathcal{A})$ , and from an edge  $\langle x, y \rangle$  in  $F$  to a set of roles,  $\mathcal{L}(\langle x, y \rangle)$ , in  $\mathcal{R}$ ;  $\dot{=}$  is an equivalence relation on nodes of  $G$ ; and  $\dot{\neq}$  is the binary relation *distinct from* on nodes of  $G$ . To check the consistency of an ABox  $\mathcal{A}$ , the completion forest is initialized as follows. There is a node  $a$  in  $G$  iff there is an individual  $a$  in  $\mathcal{A}$ .  $\langle x, y \rangle$  is an edge in  $G$  with  $R \in \mathcal{L}(\langle x, y \rangle)$  iff  $R(x, y) \in \mathcal{A}$ . For  $x$  and  $y$  in  $G$ ,  $x \dot{\neq} y$  iff  $x \neq y \in \mathcal{A}$ . Initially, there are no  $x$  and  $y$  in  $G$  such that  $x \dot{=} y$ . A *root* node  $a$  is a node present in the initial completion forest (it corresponds to the named individual with the same name in  $\mathcal{A}$ ).

Next, we show that conditions (C1), (C2) and (C3) in Section 3.2 are sufficient to rule out mergers which can increase the number of  $P$ -neighbors as shown in Figure 3(B) and (C). Lemma 3 below is an important step towards this goal.

**Lemma 3:** Let  $P$  be a role that is safe in  $\mathcal{A}$ . At any step of any execution of the tableau algorithm on  $\mathcal{A}$ , the following holds: if there is an unnamed node  $x$  such that  $P$  or  $P^-$  is in the  $\mathcal{L}(\langle parent(x), x \rangle)$ , then  $|\mathcal{L}(\langle parent(x), x \rangle)| = 1$ , where  $parent(x)$  denotes the parent node of  $x$  in the completion forest (Note that in a SHIN completion forest, unnamed nodes are always in a tree rooted at a root node).

**Proof of Lemma 3:** We prove the following property by induction: at any step  $k$  of an execution of the tableau algorithm on an Abox  $\mathcal{A}$ , if there is a unnamed node  $x$  and a safe role  $P$  in  $\mathcal{A}$  such that  $\{P, P^-\} \cap \mathcal{L}_k(\langle parent(x), x \rangle) \neq \emptyset$ , then  $|\mathcal{L}_k(\langle parent(x), x \rangle)| = 1$ .

At step  $k = 0$  (before the start of the execution), the property is trivially satisfied since there are no unnamed

nodes. We assume that at a step  $k$  of the execution the property holds, and we show that it also hold at the next step  $k + 1$ .

- Let us assume that, in order to satisfy the constraint  $\exists R.C \in \mathcal{L}(y)$  on a node  $y$ , the  $\exists$ -rule is applied at step  $k + 1$ . By definition of the  $\exists$ -rule, a unnamed node  $x$  is created such that  $\mathcal{L}(\langle parent(x), x \rangle) = \{R\}$  and  $\mathcal{L}(x) = \{C\}$ . The property is trivially satisfied at step  $k + 1$  because  $|\mathcal{L}(\langle parent(x), x \rangle)| = |\{R\}| = 1$
- We show as we did for  $\exists$ -rule, that if the  $\geq$ -rule is applied at step  $k + 1$ , the property is still satisfied.
- Let assume that, in order to satisfy the constraint  $\leq nR \in \mathcal{L}(y)$  on a node  $y$ , the  $\leq$ -rule is applied, at step  $k + 1$ , and it merges the unnamed node  $z$  into the node  $x$  ( $x$  could be a root or a unnamed node and  $z$  is not an ancestor of  $y$ ). At step  $k$ , both  $x$  and  $z$  are  $R$ -neighbors of  $y$ . One of the following must have been true :

- case 1:  $x$  is not an ancestor of  $y$ .

In this case  $parent(x) = parent(z) = y$ . By definition of the  $\leq$ -rule,

$$\mathcal{L}_{k+1}(\langle y, x \rangle) = \mathcal{L}_k(\langle y, x \rangle) \cup \mathcal{L}_k(\langle y, z \rangle) \text{ and}$$

$$\mathcal{L}_{k+1}(\langle y, z \rangle) = \emptyset.$$

We assume that  $x$  is a unnamed node (otherwise, the property is trivially satisfied at step  $k + 1$ ).

We also assume that there is a safe role  $P$  such that, after the merger,  $\{P, P^-\} \cap \mathcal{L}_{k+1}(\langle y, x \rangle) \neq \emptyset$ .  $P$  cannot satisfy the following condition: for all roles  $S$  such that either  $S$  or  $S^-$  is a sub role of  $P$ , there are no  $S$ -generators in  $clos(\mathcal{A})$  (Otherwise,  $x$  could never have been generated in the first place).

So, since  $P$  is a safe role in  $\mathcal{A}$ , it must be such that the  $attractant(P) \subseteq \{P\}$  and  $attractant(P^-) \subseteq \{P^-\}$ .

$\mathcal{L}_{k+1}(\langle y, x \rangle) = \mathcal{L}_k(\langle y, x \rangle) \cup \mathcal{L}_k(\langle y, z \rangle)$  implies that one of the following must be true: (a)  $\{P, P^-\} \cap \mathcal{L}_k(\langle y, x \rangle) \neq \emptyset$ , or (b)  $\{P, P^-\} \cap \mathcal{L}_k(\langle y, z \rangle) \neq \emptyset$

If (b) is true, by induction hypothesis, we must have  $|\mathcal{L}_k(\langle y, z \rangle)| = 1$ , which means that either  $\mathcal{L}_k(\langle y, z \rangle) = \{P\}$  or  $\mathcal{L}_k(\langle y, z \rangle) = \{P^-\}$ . If  $\mathcal{L}_k(\langle y, z \rangle) = \{P\}$  (resp.  $\mathcal{L}_k(\langle y, z \rangle) = \{P^-\}$ ), then  $P$  is a sub role of  $R$  (resp.  $P^-$  is a sub-role  $R$ ) (because, at step  $k$ ,  $z$  is a  $R$ -neighbor of  $y$ ). By definition of  $attractant(P)$  and the fact that  $\leq nR \in clos(\mathcal{A})$ , it follows that  $\{P, R\} \subseteq attractant(P)$  (resp.  $\{P^-, R\} \subseteq attractant(P^-)$ ). Since  $attractant(P) \subseteq \{P\}$

and  $\text{attractant}(P^-) \subseteq \{P^-\}$ , we must have  $R = P$  (resp.  $R = P^-$ ).

Therefore, at step  $k$ ,  $x$  is a  $P$ -neighbor (resp. a  $P^-$ -neighbor) of  $y$ , which implies that there is a subrole  $Q$  of  $P$  (resp.  $P^-$ ) such that there  $Q \in \mathcal{L}_k(< y, x >)$ . As explained in the previous paragraph,  $Q$  is a subrole of  $P$ ,  $\leq nR \in \text{clos}(\mathcal{A})$  and  $\text{attractant}(P) \subseteq \{P\}$  (resp.  $\text{attractant}(P^-) \subseteq \{P^-\}$ ) imply that  $Q = P$  (resp.  $Q = P^-$ ). By induction hypothesis, we must have  $\mathcal{L}_k(< y, x >) = \{P\}$  (resp.  $\mathcal{L}_k(< y, x >) = \{P^-\}$ ). It follows that  $|\mathcal{L}_{k+1}(< y, x >)| = 1$ , which establishes the induction property at step  $k + 1$ . The same result is obtained assuming that (a) is true.

- case 2:  $x$  is an ancestor of  $y$ . In this case,  $\text{parent}(y) = x$  and  $\text{parent}(z) = y$ . By definition of the  $\leq$ -rule,

$$\mathcal{L}_{k+1}(< x, y >) = \mathcal{L}_k(< x, y >) \cup \text{Inv}(\mathcal{L}_k(< y, z >)) \text{ and}$$

$$\mathcal{L}_{k+1}(< y, z >) = \emptyset.$$

We assume that  $y$  is a unnamed node (otherwise, the property is trivially satisfied at step  $k + 1$ ).

We also assume that there is a safe role  $P$  such that, after the merger,  $\{P, P^-\} \cap \mathcal{L}_{k+1}(< x, y >) \neq \emptyset$ .  $P$  cannot satisfy the following condition: for all roles  $S$  such that either  $S$  or  $S^-$  is a subrole of  $P$ , there are no  $S$ -generators in  $\text{clos}(\mathcal{A})$  (otherwise,  $y$  could never have been generated in the first place).

So, since  $P$  is a safe role in  $\mathcal{A}$ , it must be such that the  $\text{attractant}(P) \subseteq \{P\}$  and  $\text{attractant}(P^-) \subseteq \{P^-\}$ . The remainder of the proof proceeds in a similar way as case 1.

- All other rule applications do not change  $\mathcal{L}(< \text{parent}(x), x >)$  for an unnamed node  $x$ . ■

The fact that the safety condition (C1) is sufficient to rule out mergers illustrated in Figure 3(C) is a direct consequence of Lemma 3 and the definition of safety. Obviously, if condition (b) of the definition of safety (i.e no generators for subroles and their inverses) holds, mergers illustrated in Figure 3(C) are not possible. If (b) does not hold, then (a) imposes  $\text{attractant}(P) \subseteq \{P\}$  and  $\text{attractant}(P^-) \subseteq \{P^-\}$ . Assume that at a step  $k + 1$  of an execution of the tableau algorithm,  $a$  is merged with an unnamed node  $x$  to satisfy the constraint  $\leq nP^- \in \mathcal{L}(c)$  where, at step  $k$ ,  $a$  and  $x$  are  $P^-$ -neighbors of  $c$ , and  $\text{parent}(x) = c$ . Since  $\leq nP^- \in \text{clos}(\mathcal{A})$ , by definition of  $\text{attractant}(P^-)$ ,  $P^-$  and any of its sub-roles are in  $\text{attractant}(P^-)$ . Because of the constraint  $\text{attractant}(P^-) \subseteq \{P^-\}$ , it follows that the only sub-role of  $P^-$  is itself. Therefore,  $P \in \mathcal{L}(< a, c >)$

and  $P^- \in \mathcal{L}(< c, x >)$ , which, by Lemma 3, implies  $\mathcal{L}(< c, x >) = \{P^-\}$ . Thus, after the merger,  $\mathcal{L}(< a, c >)$  remains unchanged. This means that a merger between  $a$  and an unnamed node cannot increase of the number of  $P$ -neighbors of  $a$  if  $P$  is safe.

Now, we need to prove that if (C2) and (C3) are satisfied for a named individual  $a$ ,  $a$  cannot be merged with another named individual. First, we formally define the notion of *mergeability with a named individual*.

**Definition 4:** A named individual  $a$  in  $\mathcal{A}$  is *mergeable with named individuals* in  $\mathcal{A}$  iff there is at least one execution of the tableau algorithm on  $\mathcal{A}$  such that, at some step, the root node  $a$  is merged with another root node  $b$ . When there is no ambiguity, we simply say that  $a$  is mergeable.

**Theorem 5:** If a named individual  $a$  in  $\mathcal{A}$  satisfies conditions (C2) and (C3), then  $a$  is *not mergeable with named individuals* in  $\mathcal{A}$ .

**Proof of Theorem 5:** We assume that  $a$  in  $\mathcal{A}$  satisfies the two conditions of the theorem. We show by induction the following using Lemma 3: at any step  $k$  of an execution of the tableau algorithm, (a) there is no root node  $b$  such that the root node  $a$  is  $P$ -neighbor of  $b$  where  $(\leq nP) \in \text{clos}(\mathcal{A})$ ; and (b) if there is a root node  $b$  that is a  $P$ -neighbor of  $a$  where  $(\leq nP) \in \text{clos}(\mathcal{A})$ , then  $P$  is safe in  $\mathcal{A}$ ; and (c) the root node  $a$  has not been merged with another root node.

Before the start of the execution, (a), (b), and (c) are obviously satisfied ((a) and (b) are satisfied because (C2) and (C3) hold). Next, we assume that (a), (b) and (c) hold at a step  $k$ , and we prove that they still hold at step  $k + 1$ .

If, at step  $k + 1$ , (c) does not hold, then at this step  $a$  must have been merged with a root node  $c$  (through the application of  $\leq_r$ -rule, i.e., the rule responsible for mergers between named individuals). Therefore, by definition of the  $\leq_r$ -rule, there must be a root node  $b$  s.t., at step  $k$ ,  $a$  and  $c$  were  $P$ -neighbors of  $b$  and  $b : \leq nP$ . This contradicts the induction hypothesis (a), so (c) must still hold at step  $k + 1$ .

The only rules that can invalidate (a) and (b) are  $\leq_r$  and  $\leq$  rules (the other rules cannot add new edges or change labels of edges relating  $a$  to another root node).

A  $\leq_r$ -rule application at step  $k + 1$  that would merge  $a$  and another root node has already been ruled out.

Because, by the induction hypothesis, (a) and (b) are satisfied at step  $k$ , an application of the  $\leq_r$ -rule at step  $k + 1$  that merges two root nodes neighbors of  $a$  to satisfy a maximum cardinality constraint in  $a$  cannot make (a) or (b) unsatisfied.

Assume that, in order to satisfy the constraint  $\leq nP \in \mathcal{L}(b)$  on a root node  $b$ , the  $\leq$ -rule is applied, at step  $k + 1$ , and merges  $a$  with an unnamed node  $x$  s.t  $x$  and  $a$  are neighbors of  $b$ . By definition of  $\leq$ -rule, at step  $k$ ,  $x$  and  $a$  must be  $P$ -neighbors  $b$ . This directly contradicts the induction hypothesis (a).

Assume that, in order to satisfy the constraint  $\leq nP \in$

$\mathcal{L}(a)$ , the  $\leq$ -rule is applied, at step  $k+1$ , and merges a root node  $b$  and an unnamed node  $x$  both neighbors of  $a$ . Therefore, at step  $k$ ,  $b$  and  $x$  must be  $P$ -neighbors of  $a$ . Since the root node  $b$  is a  $P$ -neighbor  $a$  and  $\leq nP \in \text{clos}(\mathcal{A})$ , by induction hypothesis (b),  $P$  is safe in  $\mathcal{A}$ .

Since the unnamed node  $x$  is a  $P$ -neighbor of  $a$ , the condition (b) of the definition of safety (i.e. no generators for sub-roles  $P$  or their inverses) cannot be satisfied for  $P$ . The only way for  $P$  to be safe in  $\mathcal{A}$  is then to satisfy:  $\text{attractant}(P) \subseteq \{P\}$  and  $\text{attractant}(P^-) \subseteq \{P^-\}$ . Since  $P \in \text{attractant}(P)$  (because of the existence of the constraint  $a : \leq nP$ ), it follows  $\text{attractant}(P) = \{P\}$ , which, as a direct consequence of the definition of  $\text{attractant}(P)$ , implies that the only subrole of  $P$  is  $P$ . So,  $x$  is a  $P$ -neighbor of  $a$  implies that, at step  $k+1$ ,  $P \in \mathcal{L}(\langle \text{parent}(x), x \rangle)$  with  $\text{parent}(x) = a$ . By Lemma 3,  $\mathcal{L}(\langle a, x \rangle) = \{P\}$ . Therefore, the merger between  $b$  and  $x$  has not changed the set of roles  $Q$  such that  $b$  is a  $Q$ -neighbor of  $a$ . Thus, (a) and (b) must still hold.

No other application of  $\leq$  and  $\leq_r$  rules can invalidate (a) or (b) because they do not add new edges or change labels of edges relating  $a$  to another root node. ■

Finally, the correctness of our filtering criteria relies on the following theorem:

**Theorem 6:** A role assertion  $R(a, b)$  can safely be ignored in an Abox  $\mathcal{A}$  if it is irrelevant with respect to universal restrictions and irrelevant with respect to maximum cardinality restrictions, as defined in section 3.2.

**Proof of Theorem 6:** Let  $R(a, b)$  be a role assertion irrelevant w.r.t. maximum cardinality and universal restrictions in an Abox  $\mathcal{A}$ . Let  $\mathcal{A}'$  be the Abox defined as  $\mathcal{A}' = \mathcal{A} - \{R(a, b), R^-(b, a)\}$ . If  $\mathcal{A}$  is consistent,  $\mathcal{A}'$  is obviously consistent. We show that if  $\mathcal{A}'$  is consistent,  $\mathcal{A}$  is also consistent.

It would be more elegant to provide a direct model-theoretic proof. Unfortunately, because of the presence of maximum cardinalities with values greater than 1 and generator concepts, we cannot easily extend any arbitrary model of  $\mathcal{A}'$  into a model of  $\mathcal{A}$ . For example, consider the following knowledge base  $(\mathcal{A}, \mathcal{R}, \mathcal{T})$ ,

$\mathcal{R} = \{R \sqsubseteq S\}$ ,  $\mathcal{T} = \emptyset$ , and  $\mathcal{A} = \{R(a, b), R(c, d), a : (\leq 2R), a : \exists R.\neg A, a : \exists S.A, b : A, c : \top, d : A, d : \forall S^-. (\exists R.\neg A)\}$ ,

According to our criteria,  $R(a, b)$  is irrelevant w.r.t. universal restrictions and maximum cardinality restrictions.

Let  $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$  be an interpretation of  $\mathcal{A}'$  w.r.t.  $\mathcal{R}$  and  $\mathcal{T}$  defined as follows:  $\Delta^{\mathcal{I}'} = \{s, t, u, x_1\}$ ,  $a^{\mathcal{I}'} = c^{\mathcal{I}'} = s$ ,  $b^{\mathcal{I}'} = t$ ,  $d^{\mathcal{I}'} = u$ ,  $R^{\mathcal{I}'} = S^{\mathcal{I}'} = \{\langle s, x_1 \rangle, \langle s, u \rangle\}$ ,  $R^{-\mathcal{I}'} = S^{-\mathcal{I}'} = \{\langle x_1, s \rangle, \langle u, s \rangle\}$ , and  $A^{\mathcal{I}'} = \{t, u\}$ .  $\mathcal{I}'$  is obviously a model of  $\mathcal{A}'$  w.r.t.  $\mathcal{T}$  and  $\mathcal{R}$ , but it cannot easily be extended to create a model of  $\mathcal{A}$ : adding  $\langle s, t \rangle$  in  $R^{\mathcal{I}'}$  would result in a violation of the maximum cardinality constraint  $a : \leq 2R$ ; removing  $\langle s, x_1 \rangle$  from

$R^{\mathcal{I}'}$  would result in the violation of the existential restriction  $a : \exists R.\neg A$ ; removing  $\langle s, u \rangle$  from  $R^{\mathcal{I}'}$  would not result in a model of  $\mathcal{A}$  because  $\langle c^{\mathcal{I}'}, d^{\mathcal{I}'} \rangle$  would not be in  $R^{\mathcal{I}'}$ . So it is not obvious to extend  $\mathcal{I}'$  into a model of  $\mathcal{A}$ . In this example, the fundamental problem is that  $a$  and  $c$  have unnecessarily been "merged" in  $\mathcal{I}'$ , which makes it harder to satisfy existential, minimum and maximum cardinality restrictions in an extended model of  $\mathcal{A}$ .

Instead of trying to extend any arbitrary model of  $\mathcal{A}'$  into a model of  $\mathcal{A}$ , we show that if  $\mathcal{A}'$  is consistent, a model of  $\mathcal{A}$  can be constructed by applying the tableau algorithm rules in a particular way.

First, for a root node  $c$  in the completion forest  $F$ , the root node  $\alpha(c)$  is defined as follows (informally,  $\alpha(c)$  corresponds to the node in which  $c$  has been directly or indirectly merged):

$$\alpha(c) = \begin{cases} c & \text{if } \mathcal{L}(c) \neq \emptyset \\ d & \text{if } \mathcal{L}(c) = \emptyset, d \text{ is the unique root node in } F \\ & \text{with } \mathcal{L}(d) \neq \emptyset \text{ and } d \doteq c \end{cases}$$

Since  $\mathcal{A}'$  is consistent, we can apply the tableau expansion rules on  $\mathcal{A}'$  without creating a clash in such a way that:

- $\exists$ -rule is never triggered to satisfy a constraint  $\exists P.C \in \mathcal{L}(\alpha(a))$  (resp.  $\mathcal{L}(\alpha(b))$ ) where  $\leq nP \in \text{clos}(\mathcal{A})$ ,  $R$  (resp.  $R^-$ ) is part of  $\leq nP$ , and  $b : C \in \mathcal{A}$  (resp.  $a : C \in \mathcal{A}$ ), and
- $\geq$ -rule is never triggered to satisfy a constraint  $\geq nP \in \mathcal{L}(\alpha(a))$  (resp.  $\mathcal{L}(\alpha(b))$ ) where  $\leq nP \in \text{clos}(\mathcal{A})$ ,  $R$  (resp.  $R^-$ ) is part of  $\leq nP$ , and, in the Abox  $\mathcal{A}$ ,  $b$  (resp.  $a$ ) is one of  $n$   $P$ -neighbors of  $a$  (resp.  $P$ -neighbors of  $b$ ) explicitly asserted to be distinct.

Such a rule application yields a clash-free completion forest  $F$ , and the only nodes on which expansion rules may be applicable are  $\alpha(a)$  and  $\alpha(b)$  (the only applicable rules are  $\exists$ -rule and  $\geq$ -rule).

Next, we modify  $F$  to create a completion forest  $F'$  by adding to  $F$  the edge  $\langle \alpha(a), \alpha(b) \rangle$  if it was not already in  $F$ , and by adding  $R$  to  $\mathcal{L}(\langle \alpha(a), \alpha(b) \rangle)$ , if it was not already there. We show that  $F'$  is complete (i.e. no rules are applicable) and clash-free.

The fact that, in  $F'$ ,  $R \in \mathcal{L}(\langle \alpha(a), \alpha(b) \rangle)$  ensures that the  $\exists$  and  $\geq$  rules, which may have been applicable on  $\alpha(a)$  or  $\alpha(b)$  in  $F$ , are not applicable on  $\alpha(a)$  and  $\alpha(b)$  in  $F'$ . However, the same fact may now make the  $\forall$ ,  $\forall_+$ ,  $\leq$ , and  $\leq_r$  rules applicable on  $\alpha(a)$  or  $\alpha(b)$  in  $F'$ . We show that this cannot be the case.

The definition of irrelevance w.r.t. universal restrictions given in section 3.2 obviously ensures that  $\forall$  and  $\forall_+$  rules are not applicable on  $\alpha(a)$  or  $\alpha(b)$  in  $F'$ .  $\leq$ , and  $\leq_r$  rules are not applicable on  $\alpha(a)$  or  $\alpha(b)$  in  $F'$  as a direct consequence of the following claim :

**Claim:** if  $R(a, b)$  is irrelevant w.r.t  $\leq nP \in \text{clos}(\mathcal{A})$  and  $R$  (resp.  $R^-$ ) is part  $\leq nP$ , then the number of  $P$ -neighbors of  $a$  (resp.  $P$ -neighbors of  $b$ ) in  $F$  is less than or equal to  $n$ . Furthermore, if it is equal to  $n$ , then, in  $F$ ,  $\alpha(b)$  is a  $P$ -neighbor of  $\alpha(a)$  (resp.  $\alpha(a)$  is a  $P$ -neighbor of  $\alpha(b)$ ).

The proof of this claim is a direct consequence of Lemma 3, Theorem 5 and the fact that the upper-bound (defined in section 3.2) of  $P$ -neighbors of  $a$  is less than or equal to  $n$  (the proof of this claims is given at the end of the proof of Theorem 6).

The addition of  $R \in \mathcal{L}(\langle \alpha(a), \alpha(b) \rangle)$  to  $F$  cannot create a clash of the form  $\{C, \neg C\}$  in  $F'$ , and the previous claim implies that a clash in  $F'$  due to a violation of a maximum cardinality constraint on  $\alpha(a)$  or  $\alpha(b)$  is not possible.

Thus,  $F'$  is a complete clash-free completion forest such that  $R \in \mathcal{L}(\langle \alpha(a), \alpha(b) \rangle)$ . Therefore, a tableau for  $\mathcal{A}$  can be built from  $F'$  as in [10], which establishes that  $\mathcal{A}$  has a model. ■

To formally established Claim A, we first introduce the following notation:

**Notation:** For an unnamed node  $x$  in a completion forest,  $g(x)$  denotes the role generator concept  $\exists Q.C$  or  $\geq nQ$  that triggered the application of the  $\exists$ -rule or  $\geq$ -rule resulting in the creation of  $x$ . For a root node  $a$ ,  $g(a)$  is not defined.

Claim A stated in the proof of theorem 6 follows immediately from the following Lemma 7 (3) and (4):

**Lemma 7** Let  $R(a, b)$  be a role assertion irrelevant w.r.t. maximum cardinality and universal restrictions in an Abox  $\mathcal{A}$ . Let  $\mathcal{A}'$  the Abox defined as  $\mathcal{A}' = \mathcal{A} - \{R(a, b), R^-(b, a)\}$ . Let  $P$  be a role such that  $\leq nP \in \text{clos}(\mathcal{A})$  and  $R$  is part of  $\leq nP$ . Let  $F$  denote the completion forest obtained from the first step of the application of tableau rules on  $\mathcal{A}'$  as described in the proof of Theorem 6 (i.e. without applying some  $\exists$ -rule and  $\geq$ -rule on  $\alpha(a)$  and  $\alpha(b)$ ). Let  $P(a)$  denote the set of  $P$ -neighbors of  $a$  in  $F$ , the following hold:

- 1.  $\alpha(a) = a$
- 2.  $P(a) = X \cup A \cup B$  and  $A, B$ , and  $X$  are mutually disjoint sets defined as follows:
 
$$A = \{x \in F \mid x \text{ is a } P\text{-neighbor of } a \text{ and } g(x) = \exists P.C \text{ s.t. } b : C \notin \mathcal{A}\}, \text{ and}$$

$$B = \{x \in F \mid x \text{ is a } P\text{-neighbor of } a \text{ and } g(x) = \geq mP \text{ s.t., in } \mathcal{A}, b \text{ is not one of } m \text{ } P\text{-neighbors of } a \text{ explicitly asserted to be mutually distinct}\}$$

$$X = \{c \in F \mid c \text{ in } \mathcal{A}' \text{ and } c \text{ is } P\text{-neighbor of } a\}$$
- 3. if  $\text{attractant}(P) = \{P\}$ , then  $|P(a)| \leq n - 1$
- 4. if  $\text{attractant}(P) \neq \{P\}$ , then  $|P(a)| \leq n, |A| = |B| = 0$  and  $(|P(a)| = n \text{ implies } b \text{ is a } P\text{-neighbor of } a \text{ in } \mathcal{A}')$ .

Informally, elements of  $A$  are unnamed nodes  $P$ -neighbors of  $a$  created through an application of an  $\exists$ -rule; elements of  $B$  are unnamed nodes  $P$ -neighbors of  $a$  created through an application of an  $\geq$ -rule ; and elements of  $X$  are root nodes  $P$ -neighbors of  $a$ .

**Proof of Lemma 7** Consequence of the definition of role assertion irrelevant w.r.t. maximum cardinality and universal restrictions in an Abox, lemma 3 and theorem 5.

(1)  $R(a, b)$  irrelevant in  $\mathcal{A}$  implies that  $a$  is not mergeable in  $\mathcal{A}$  and  $\mathcal{A}'$  (direct consequence of the fact that conditions (C2) and (C3) must be satisfied and theorem 5). Therefore  $\mathcal{L}(a) \neq \emptyset$ , which implies that  $\alpha(a) = a$ .

(2) Since  $R(a, b)$  is irrelevant with w.r.t.  $\leq nP$ , by definition,  $P$  must be safe in  $\mathcal{A}$ . By definition of safety, one of the following must be true

- case 1: for all roles  $S$  such that either  $S$  or  $S^-$  is a sub-role of  $P$ , there are no  $S$ -generators in  $\text{clos}(\mathcal{A})$ . There cannot be any unnamed node  $x$  such that  $x$  is a  $P$ -neighbor of  $a$ , so  $P(a) = X$  and  $A = B = \emptyset$
- case 2: The  $\text{attractant}(P) \subseteq \{P\}$  and  $\text{attractant}(P^-) \subseteq \{P^-\}$ . As we have already established, in this case, the only subroles of  $P$  is  $P$ . Therefore, Lemma 3 ensures that the only unnamed nodes  $x$   $P$ -neighbors of  $a$  must be such that  $g(x) = \exists P.C$  or  $g(x) = \geq mP$ . Furthermore, due to the specific way tableau rules are applied to construct  $F$ , we have the following: (a) all unnamed nodes  $x$   $P$ -neighbors of  $a$  with  $g(x) = \exists P.C$  are such that  $b : C \notin \mathcal{A}$ , and (b) all unnamed nodes  $x$   $P$ -neighbors of  $a$  with  $g(x) = \geq mP$  are such that  $b$  is not one of  $m$   $P$ -neighbors of  $a$  explicitly asserted to be distinct. So all unnamed nodes  $P$ -neighbors of  $a$  are in  $A \cup B$ , and  $A$  and  $B$  are obviously disjoint. All root nodes are obviously in  $X$ . By definition of  $A, B$  and  $X$ , they contain only  $P$ -neighbors of  $a$ .

(3) If  $\text{attractant}(P) = \{P\}$ , then, as already established, the only subrole of  $P$  is  $P$ . Since  $R$  is part of  $\leq nP$ ,  $R$  must be a subrole of  $P$ . So  $R = P$ .

By definition of  $A$ , an element  $x$  of  $A$  with  $g(x) = \exists P.C$  must be such that there is no named individual  $c$  in  $\mathcal{A}'$   $P$ -neighbor of  $a$  before the start of the execution of tableau rules and such that  $c : C \in \mathcal{A}'$  (otherwise,  $x$  would not have been generated in the first place). Furthermore, by definition of  $A, b : C \notin \mathcal{A}$ . Therefore,  $|A| \leq |\text{Some}^{\mathcal{A}}(P, a)|$  with  $\text{Some}^{\mathcal{A}}(P, a) = \{\exists P.C \in \text{clos}(\mathcal{A}) \mid \text{there is no named individual } c \text{ } P\text{-neighbor of } a \text{ (before the start of the tableau algorithm) such that } c : C \in \mathcal{A}\}$ .

By definition of  $B$ , an element  $x$  of  $B$  with  $g(x) = \geq mP$  must be such that there are no  $m$  named individuals in  $\mathcal{A}'$   $P$ -neighbors of  $a$  before the start of the execution of tableau rules and all asserted to be mutually distinct (otherwise,  $x$  would not have been generated in the

first place). Furthermore, by definition of  $B$ ,  $b$  cannot be one of  $m$  distinct  $P$ -neighbors of  $a$  explicitly asserted to be mutually distinct in  $\mathcal{A}$  before the start of the application of tableau rules. This implies that if  $x$  is an element of  $B$  with  $g(x) \geq mP$ , then, in  $\mathcal{A}$ , there are no  $m$   $P$ -neighbors of  $a$  explicitly asserted to be mutually distinct. Therefore  $|B| \leq \sum_{\geq mP \in \text{Min}^{\mathcal{A}}(P,a)} m$  with  $\text{Min}^{\mathcal{A}}(P,a) = \{\geq mP \in \text{clos}(\mathcal{A})\}$  in  $\mathcal{A}$ , before the start of the execution of tableau rules, there are no individuals  $d_i$  such that, for  $1 \leq i \leq m$ ,  $d_i$  is a  $P$ -neighbors of  $a$ , and if  $j \neq k$ , then  $d_k \neq d_j \in \mathcal{A}$

$R(a,b)$  is irrelevant w.r.t  $\geq nP$  in  $\mathcal{A}$  implies  $a$  is not mergeable in  $\mathcal{A}$  and  $\mathcal{A}'$  and  $P$  is safe in both  $\mathcal{A}$  and  $\mathcal{A}'$ . From the non-mergeability of  $a$ , the safety of  $P$  and Lemma 3, it follows that  $|X| \leq |P_0|$  where  $|P_0|$  denotes the number of  $R$ -neighbors of  $a$  in  $\mathcal{A}'$  before the start of the tableau algorithm on  $\mathcal{A}'$ . Because  $P$  has no subrole besides itself, the number of  $P$ -neighbors of  $a$  in  $\mathcal{A}$  is  $|P_0| + 1$ . Since the definition of irrelevant w.r.t. to  $\leq nP$  requires  $(|P_0| + 1) + |\text{Some}^{\mathcal{A}}(P,a)| + \sum_{\geq mP \in \text{Min}^{\mathcal{A}}(P,a)} m \leq n$ . It follows that  $|A| + |B| + |X| \leq n - 1$

(4) Let us assume that  $\text{attractant}(P) \neq \{P\}$ . Since  $R(a,b)$  is irrelevant w.r.t.  $\leq nP$ ,  $P$  must be safe in  $\mathcal{A}$ . Since  $\text{attractant}(P) \neq \{P\}$  and  $\text{attractant}(P) \neq \emptyset$  (because  $R$  is part of  $\leq nP \in \text{clos}(\mathcal{A})$  implies  $\{R,P\} \in \text{attractant}(P)$ ),  $P$  can be safe only if itself, its inverse, its subroles and their inverses have no generators in  $\text{clos}(\mathcal{A})$ . Therefore,  $|A| = |B| = 0$ . As in the proof of (3), from the non-mergeability of  $a$ , the safety of  $P$  and Lemma 3, it follows that  $|X| \leq |P_0|$  where  $|P_0|$  denotes the number of  $P$ -neighbors of  $a$  in  $\mathcal{A}'$  before the start of the tableau algorithm on  $\mathcal{A}'$ . The fact that  $R(a,b)$  is irrelevant w.r.t. to  $\leq nP$  implies  $|P_0| \leq n$ , which establishes  $|X| \leq n$ . If  $|X| = n$ , then  $|P_0| = n$ . This means that  $b$  is a  $P$ -neighbor of  $a$  in  $\mathcal{A}'$  (otherwise, the number of  $P$ -neighbors of  $a$  in  $\mathcal{A}$  would be  $n + 1$ , which is an obvious violation of the definition of the irrelevance of  $R(a,b)$  w.r.t.  $\leq nP$ ). ■

### 3.4 Summary Abox Filtering

We now discuss how to apply the filtering techniques described in Section 3.2 to the summary Abox  $\mathcal{A}'$ . Although filtering can be applied to the original Abox  $\mathcal{A}$  before applying the tableau algorithm, we do not filter  $\mathcal{A}$  directly. Rather, we first build a canonical summary Abox  $\mathcal{A}'$ . Next we apply filtering to  $\mathcal{A}'$ , and then run the tableau algorithm. For correctness with respect to cardinality restrictions, we need to augment the summary Abox with role assertion statistics, since role assertions are merged by the summary Abox transformation. For each role  $R$  that is part of a cardinality restriction, we associated with  $R$  the maximum number of  $R$ -neighbors that any individual  $a$  has in  $\mathcal{A}$ . With this augmentation, it is clear that the proofs in Section 3.3 apply

to the canonical summary Abox.

Typically, filtering  $\mathcal{A}'$  creates distinct partitions, and we apply the tableau algorithm to each partition separately. If all of the partitions are consistent, then we are done. Otherwise, we need to check  $\mathcal{A}$ . However, even when  $\mathcal{A}'$  itself is inconsistent, some of its partitions may be consistent, and we only have to check portions of  $\mathcal{A}$  which correspond to the filtered inconsistent partitions of  $\mathcal{A}'$ . Thus, partitioning a summary Abox is an effective way of isolating a potential inconsistency in  $\mathcal{A}$ . Furthermore, filtering  $\mathcal{A}'$  is very efficient since  $\mathcal{A}'$  is relatively small. For partitions consisting of a single individual, checking consistency is just checking concept satisfiability.

More precisely, let  $\mathcal{A}'_p$  be a partition of individuals and assertions in  $\mathcal{A}'$ . The *image* of  $\mathcal{A}'_p$  in  $\mathcal{A}$  is defined to be the maximum subset of the individuals and assertions in  $\mathcal{A}$  which map to  $\mathcal{A}'_p$  via the summary Abox function  $\mathbf{f}$ . If a role assertion  $R(a,b)$  is irrelevant in  $\mathcal{A}'$ , then all role assertions in its image in  $\mathcal{A}$  are also irrelevant. By theorem 2, if a partition  $\mathcal{A}'_p$  is consistent, then its entire image in  $\mathcal{A}$  can be ignored. Finally, retrieving the image in  $\mathcal{A}$  of an inconsistent partition  $\mathcal{A}'_p$  is a simple database operation.

For example, consider the summary Abox  $\mathcal{A}'$  in Figure 2, and suppose that we filter all  $R$ -role assertions. The resulting summary Abox shown in Figure 4 consists of three partitions: X, Y, and Z. We run the consistency check on each partition (partition Z is a singleton, so we only need to check concept satisfiability). Assuming only partition X is inconsistent, we need to check only the consistency of its image in  $\mathcal{A}$ , shown in Figure 5, which is  $d1, d2 : D; b1 - b6 : B; T(d1, b1)$  and  $P(d2, b4)$ . Since  $b2, b3, b5$  and  $b6$  are isolated individuals, checking the consistency is handled by a concept satisfiability test. We run a consistency check on the remaining elements of the image of X.

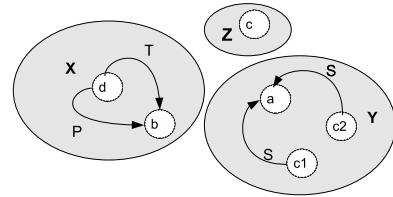


Figure 4. Filtered summary Abox

## 4 Computational Experience

We tested our approach on the four ontologies shown in Table 2. Their expressiveness is given in the first column (Exp) of Table 2. The number of concepts (C) and roles (R) reported in the table reflect concepts and roles actually used in the Abox rather than the Tbox. In the tables, R.A. stands for role assertions, and I for individuals. In all

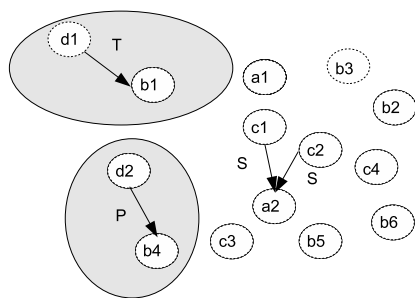


Figure 5. Filtered Abox image

the experiments reported here, the Aboxes for the ontologies were stored in a relational database (DB2) on a 64 bit AMD 997 Mhz dual processor machine with 8 G RAM. Our program ran as a client to the database server, connecting through JDBC to compute and filter the summary. We tested our program both on a 32 bit single 1.8 Ghz processor machine with 1.5 G RAM, and on the 64 bit database server described above. Running times on the 32 bit machine were about 2 times slower than the times reported in the tables, but the program ran with minimal space requirements on both machines (default heap size was 512 M).

Biopax is an ontology that describes biochemical pathways. The Aboxes for Biopax (<http://www.biopax.org/Documents.html>) consist of pathway data for 11 organisms publicly available from the Biocyc website (<http://biocyc.org>). LUBM [5] is a benchmark ontology of universities, and it can be scaled to different numbers of universities to test the performance of OWL reasoners. We used a version of the Lehigh University Benchmark which was augmented to include the expressiveness of OWL-DL [13], but with nominals removed. The next two ontologies were internally developed. The NIMD ontology expresses relationships between persons, places and events (<http://ksl.stanford.edu/projects/NIMD/Kanid1-v1.owl>). Its Abox was generated from text analysis programs run over a large number of unstructured documents. The semantic traceability (ST) ontology specifies the relationships among software artifacts. Its Abox was generated from a program that extracted relationships between software artifacts of a large middleware application. The sizes of the last 4 Aboxes shown in Table 2 are beyond the capabilities of in-memory reasoners such as Pellet and KAON2, when tested on a 64-bit machine with a 4G heap size.

Table 3 shows the size of the corresponding summary Aboxes prior to any filtering. As noted in earlier sections, the summary Abox can be computed once, and maintained with changes to the Abox. Times to compute the summary Abox is given in seconds.

Table 4 shows the effectiveness of filtering the summary

Table 2. Experimental Aboxes.

Ontology	Exp	C	R	I	R.A.
Biopax	ALCHF	31	40	261,149	582,655
LUBM-1	SHIN	91	27	42,585	214,177
LUBM-5	SHIN	91	27	179,871	927,854
LUBM-10	SHIN	91	27	351,422	1,816,153
LUBM-30	SHIN	91	27	1,106,858	6,494,950
NIMD	SHIF	19	27	1,278,540	1,999,787
ST	SHI	16	11	874,319	3,595,132

Table 3. Summary Aboxes prior to filtering.

Ontology	C	R	I	R.A	Time
Biopax	31	40	81	583	46
LUBM-1	91	27	410	16,233	12
LUBM-5	91	27	598	35,375	60
LUBM-10	91	27	673	49,176	128
LUBM-30	91	27	765	79,845	485
NIMD	19	27	19	55	77
ST	16	11	21	183	197

Abox for the consistency detection test. Note that the filtering step is dynamic, i.e., it must be computed on the summary box for each incoming query. The filtering step filters irrelevant role assertions, and as a result, can create partitions. In Table 4, the first number in the first column (Sin.+Mult) indicates the number of partitions with single individuals, and the second number indicates the number of partitions with multiple individuals. The rest of the columns show the size of the Abox that is left after removing all partitions with single individuals. Time to perform the filtering is presented in seconds.

Table 5 shows the size of the Abox on which we had to perform the consistency check. All times for the consistency check were measured using the Pellet OWL reasoner. For those Aboxes where the filtered summary Abox in Table 4 was consistent, the size of the Abox was simply that in Table 4. For some ontologies, however (e.g., all LUBM ontologies marked with an asterisk), the filtered Abox was inconsistent because of our summarization techniques. For these ontologies, we had to retrieve the image of the inconsistent partition from the original Abox. In these cases, the

Table 4. Summary Aboxes after filtering

Ontology	Sin.+Mult.	C	R	I	R.A.	Time
Biopax	42+1	13	1	38	98	1.6
LUBM-1	130+2	28	5	280	284	1.4
LUBM-5	172+2	28	5	426	444	2.1
LUBM-10	199+2	28	5	474	492	2.5
LUBM-30	220+2	28	5	545	574	2.8
NIMD	17+1	2	1	2	1	0.6
ST	3+1	15	2	18	50	0.3

**Table 5. Abox sizes for consistency check.**

Ontology	I	R.A.	Time	Consistent
Biopax	38	98	0.7	Yes
LUBM-1*	140	102	1	Yes
LUBM-5*	644	466	1.5	Yes
LUBM-10*	1283	938	2	Yes
LUBM-30*	4045	2942	3.5	Yes
NIMD	2	1	0.2	Yes
ST	-	-	0.1	No

size shown in Table 5 is the image of the partition in the original Abox. Time for consistency check is provided in seconds. This includes the time for the concept satisfiability check for partitions with single individuals, the time for the consistency check on the filtered summary, and the time for retrieving and checking the image of the inconsistent partition on the original Abox. As shown in Table 5, ST was an inconsistent ontology, but we determined this purely based on a concept satisfiability check for partitions with single individuals. We also deliberately injected an inconsistency for one of the Biopax databases (agrocyc), to check if we could detect an inconsistency that could not simply be detected by a concept satisfiability check. We were able to detect that the Abox was inconsistent using our algorithm.

## 5 Related Work

Consistency checking of various expressive DLs is a rich research area [1], and there are many highly optimized reasoners such as Fact [9], Pellet [14], Racer [8], InstanceStore [2], and Kaon2 [16] designed for consistency checking. All but InstanceStore are in-memory reasoners in their current implementations, and therefore cannot scale to very large ABoxes, as demonstrated in our results. We focus our discussion on Kaon2 and InstanceStore, because they do apply to ABoxes in secondary storage.

InstanceStore is limited to role-free ABoxes. In theory, InstanceStore can handle ABoxes with role assertions through a technique called precompletion [15]. Precompletion applies a set of non-deterministic rules to make explicit all consequences of role assertions, so that the role assertions can subsequently be removed from the resulting ABoxes. For large ABoxes stored in databases, the precompletion approach may not be practical, since it can generate an exponential number of precompleted ABoxes. Furthermore, to the best of our knowledge, precompletion has only been demonstrated for SHF description logic (i.e no inverse roles and only maximum cardinality restrictions of 1). As acknowledged in [15], its extension to SHIN is not obvious.

Kaon2 reduces a SHIQ(D) ontology to a disjunctive datalog program, which makes it naturally applicable to

Aboxes stored in deductive databases. Our approach is an alternative to this approach, and works with ABoxes stored in traditional relational databases.

Our techniques of summarization and filtering can be contrasted with tableau algorithm optimization techniques such as model caching and Abox contraction. Model caching [7][3] is one of the most effective optimization techniques used in tableau reasoners. However, it may not work effectively in the presence of role assertions. Abox contraction [6] aims at improving the efficiency of model caching in the presence of role assertions by replacing acyclic, tree-like role paths between individuals by the appropriate existential restrictions. Although these are effective optimization techniques for ABoxes that fit in memory, it is unclear how such techniques can be applied to ABoxes in databases. In our approach, we accrue the same benefits as model caching by summarization and filtering of irrelevant role assertions.

Abox partitioning plays a key role in our approach since we discover partitions in the original Abox by filtering role assertions in the summary. Other partitioning techniques have been proposed for OWL ontologies [4] but again, these techniques apply efficiently only to ABoxes that fit in memory, not to ABoxes in databases.

## 6 Conclusion

We have demonstrated a technique to scale consistency detection to large ABoxes in secondary storage by extracting a representative summary Abox. Further, we have shown that, in practice, this technique works efficiently on four large ontologies. In all cases, the time and space requirements for the consistency check was on the order of seconds or minutes, even for ABoxes of 6 million assertions. Our plan is to extend this approach to more expressive languages, and refine these techniques for more efficient query processing.

## References

- [1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
- [2] S. Bechhofer, I. Horrocks, and D. Turi. The owl instance store: System description. *Proc. of 20th Int. Conf. on Automated Deduction*, pages 177–181, 2005.
- [3] F. Donini. Complexity of reasoning. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors, *Description Logic Handbook*, pages 101–141. Cambridge University Press, 2002.
- [4] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Automatic partitioning of owl ontologies using e-connections. In *Description Logics*, 2005.

- [5] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large owl datasets. *Third International Semantic Web Conference*, pages 274–288, 2004.
- [6] V. Haarslev and R. Moller. An empirical evaluation of optimization strategies for abox reasoning in expressive description logics. *Proc. of the International Workshop on Description Logics*, pages 115–199, 1999.
- [7] V. Haarslev and R. Möller. Consistency testing: The race experience. In *TABLEAUX*, pages 57–61, 2000.
- [8] V. Haarslev and R. Moller. Racer system description. *Conf. on Automated Reasoning (IJCAR 2001)*, pages 701–705, 2001.
- [9] I. Horrocks. Using an expressive descriptive logic: fact or fiction? *Proceedings of the 6th Int. Conf. on principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
- [10] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ\*. *Proc. of 17th Int. Conf. on Automated Deduction*, pages 482–496, 2000.
- [11] I. Horrocks and S. Tessaris. Querying the semantic web: a formal approach. In I. Horrocks and J. Hendler, editors, *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science, pages 177–191. Springer-Verlag, 2002.
- [12] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *KR*, pages 285–296, 2000.
- [13] L. Ma, Y. Yang, Z. Qiu, G. Xie, and Y. Pan. Towards a complete owl ontology benchmark. In *Proc. of the third European Semantic Web Conf. (ESWC 2006)*, 2006.
- [14] E. Sirin and B. Parsia. Pellet: An owl dl reasoner. In *Description Logics*, 2004.
- [15] S. Tessaris and I. Horrocks. Abox satisfiability reduced to terminological reasoning in expressive description logics. In *LPAR*, pages 435–449, 2002.
- [16] U. Hustadt, B. Motik, and U. Sattler. Reducing shiq description logic to disjunctive datalog programs. *Proc. of 9th Intl. Conf. on Knowledge Representation and Reasoning (KR2004)*, pages 152–162.