Published in final edited form as:

Artif Intell Appl. 2015 February ; 2(1): 8–31. doi:10.15764/AIA.2015.01002.

Module Extraction for Efficient Object Queries over Ontologies with Large ABoxes

Jia Xu^{1,*}, Patrick Shironoshita¹, Ubbo Visser², Nigel John¹, and Mansur Kabuka¹

¹Electrical and Computer Engineering Department, University of Miami, Coral Gables, FL 33146

²Department of Computer Science, University of Miami, Coral Gables, FL 33146

Abstract

The extraction of logically-independent fragments out of an ontology ABox can be useful for solving the tractability problem of querying ontologies with large ABoxes. In this paper, we propose a formal definition of an ABox module, such that it guarantees complete preservation of facts about a given set of individuals, and thus can be reasoned independently w.r.t. the ontology TBox. With ABox modules of this type, isolated or distributed (parallel) ABox reasoning becomes feasible, and more efficient data retrieval from ontology ABoxes can be attained. To compute such an ABox module, we present a theoretical approach and also an approximation for *SHIQ* ontologies. Evaluation of the module approximation on different types of ontologies shows that, on average, extracted ABox modules are significantly smaller than the entire ABox, and the time for ontology reasoning based on ABox modules can be improved significantly.

Keywords

Ontology Querying; ABox Module; *SHIQ*

1. Introduction

Description logics (DLs), as a decidable fragment of first-order logic, are a family of logic based formalisms for knowledge representation, and the mathematical underpinning for modern ontology languages such as OWL [1, 2] and OWL 2 [3]. A DL ontology (or knowledge base) consists of a terminological part (TBox) T that defines concepts and roles of a given domain, and an assertional part (ABox) A that describes instances of the conceptual knowledge. Similar to a database, the ontology TBox usually represents the data schema, and the ontology ABox corresponds to the actual data set.

In recent years, DL ontologies have been increasingly applied in the development of DL-based information systems in diverse areas, including biomedical research [4, 5], health care [6, 7], decision support [8, 9], and many others; see [10] for a review of existing applications. Most of these DL applications involve intensive querying of the underlying knowledge that requires reasoning over ontologies.

*Corresponding author: j.xu11@umiami.edu.

Standard DL reasoning services include *subsumption testing* (i.e. testing if one concept is more general than the other), and *instance checking* (i.e. checking if an individual is one instance of a given concept). The former is considered TBox reasoning, and the latter is considered ABox reasoning as well as the central reasoning task for information retrieval from ontology ABoxes [11]. These reasoning services are highly complex tasks, especially for ontologies with an expressive DL [12, 13]. For example, instance checking in a *SHIQ* ontology is in EXPTIME [13]. Consequently, query-answering over ontologies that rely on instance checking [14] can have high computational complexity [15–17].

Existing systems that provide standard DL reasoning services include HermiT [18], Pellet [19], FaCT++ [20], and Racer [21]. They are all based on the (*hyper*)*tableau* algorithms that are proven sound and complete. Despite highly optimized implementations of reasoning algorithms in these systems, they are still confronted with the scalability problem in practical DL applications, where the TBox could be relatively small and manageable, the ABox tends to be extremely large (e.g. in semantic web setting), which can result in severe tractability problems [22, 23].

Due to the central role that is played by ontology querying in data-intensive DL-applications, various approaches have been proposed to cope with this ABox reasoning problem. For example, there are optimization strategies proposed by [24] for instance retrieval, and also techniques developed for ABox reduction [25] and partition [26, 27]. [28] proposed to reduce DL ontology into a disjunctive datalog program, so that existing techniques for querying deductive databases can be reused; and [29], [30], and [31] have suggested the combination of ontology reasoning with inference rules.

In this paper, conceiving that a large ABox may consist of data with great diversity and isolation, we explore the modularity of an ontology ABox and expect ABox reasoning to be optimized by utilizing logical properties of ABox modules. Analogous to modularity defined for the ontology TBox [32, 33], the notion of modularity for the ABox proposed in this paper is also based on the semantics and implications of ontologies; and an ABox module should be a *closure* or *encapsulation* of logical implications for a given set of individuals (that is, it should capture all facts, both explicit and implicit, of the given entities), such that each module can be reasoned *independently* w.r.t. T .

This property of an ABox module is desired for efficient ABox reasoning under situations, either when querying information of a particular individual, such as instance checking (i.e. test if $K \models C(a)$) or arbitrary property assertion checking (i.e. test if $K \models R(a,b)$), or when performing instance retrieval and answering (conjunctive) queries over ontologies [14, 16]. For the first case, an independent reasoning on the ABox module for that particular individual will be sufficient, while for instance retrieval, the property of ABox modules would allow the ABox reasoning to be parallelized, and thus being able to take advantage of existing parallel-processing frameworks, such as MapReduce [34].

For illustration, consider a real-world ontology that models and stores massive biomedical data in an ABox, and a query-answering system that is based on this ontology. A biomedical researcher may want to obtain information of a particular gene instance, say *CHRNA4*, to see

if it is involved in any case of diseases, and in what manner. To answer such queries submitted to this ontology-based system, a reasoning procedure is normally invoked and applied to the ontology. Note however, given the fact that in this case the entire ontology ABox is extremely large, complete reasoning can be prohibitively expensive [16, 17], and it may take a lengthy period to reach any conclusions. Therefore, it would be preferable to perform reasoning on a smaller subject-related (in this case, *CHRNA4*-related) module. In particular, this *CHRNA4*-related ABox module should be a closure of all facts about the individual *CHRNA4*, so that reasoning on this module w.r.t. to T can achieve the same conclusions about *CHRNA4* as if the reasoning would be applied to the entire ontology. In addition, the reasoning time should be significantly decreased, provided the module is precise and much smaller than the entire ABox. Another case of querying over this biomedical ontology could be to retrieve all genes in the ABox that belong to the same pathological class of a certain disease. Answering such queries requires to perform instance retrieval, and a simple strategy based on the ABox module here is to partition the ABox based on groups of individuals and to distribute each independent reasoning task into a cluster of computers.

Our main contributions in this paper are summarized as follows:

1. To capture the notion of ABox modularity, we provide a set of formal definitions about the ABox module, specifying logical properties of an ABox module such that it guarantees preservation of sound and complete facts of a given set of individuals, and such that it can be independently reasoned over w.r.t. the TBox;
2. To extract such an ABox module, we develop a theoretical approach for DL *SHIQ* based on the idea of *module-essential* assertions. This approach gives an exposition for the problem of ABox-module extraction from a theoretical perspective, allowing a better understanding of the problem, and providing strategies that can be built into a general framework for computation of ABox modules;
3. To cope with the complexity for checking module-essentiality using a DL-reasoner and to make the techniques applicable to practical DL applications, we also present a simple and tractable syntactic approximation.

Additionally, we present an evaluation of our approximation on different ontologies with large ABoxes, including those generated by existing benchmark tools and realistic ones that are used in some biomedical projects. We also present and compare several simple optimization techniques for our approximation. And finally, we show the efficiency of ontology reasoning that can be achieved when using the ABox modules.

This paper comes with a technical report [35] for proofs of propositions here.

2. Preliminaries

2.1 Description Logic *SHIQ*

The Description Logic *SHIQ* is extended from the well-known logic *ALC* [36], with added supports for role hierarchies, inverse roles, transitive roles, and qualified number restrictions

[37]. A *SHIQ* ontology defines a set $\mathbf{R}$ of role names, a set $\mathbf{I}$ of individual names, and a set $\mathbf{C}$ of concept names.

Definition 2.1 (SHIQ-Role)—A *SHIQ*-role can be an atomic (named) role $R \in \mathbf{R}$, or an inverse role R^- with $R \in \mathbf{R}$. The complete role set in a *SHIQ* ontology is denoted $\mathbf{R}^* = \mathbf{R} \cup \{R^- | R \in \mathbf{R}\}$. To avoid R^{--} , the function $\text{Inv}(\cdot)$ is defined, such that $\text{Inv}(R) = R^-$ and $\text{Inv}(R^-) = R$.

A role hierarchy H_R can be defined in an ontology by a set of role inclusion axioms, each of which is expressed in the form of $R_1 \sqsubseteq R_2$, with $R_1, R_2 \in \mathbf{R}^*$. We call R_1 a *subrole* of R_2 , if $R_1 \sqsubseteq R_2 \in H_R$ or if there exist $S_1, \dots, S_n \in \mathbf{R}^*$ with $R_1 \sqsubseteq S_1, S_1 \sqsubseteq S_2, \dots, S_n \sqsubseteq R_2 \in H_R$. Here, $\sqsubseteq$ is reflexive and transitive.

A role $R \in \mathbf{R}$ is *transitive*, denoted $\text{Trans}(R)$, if $R \circ R \sqsubseteq R$ or $\text{Inv}(R) \circ \text{Inv}(R) \sqsubseteq \text{Inv}(R)$. Finally, a role is called *simple* if it is neither transitive nor has any transitive subroles [12, 37].

Definition 2.2 (SHIQ-Concept)—A *SHIQ*-concept is either an atomic (named) one or a complex one that can be defined using the following constructors recursively

$$C, D ::= A | \neg C | C \sqcap D | C \sqcup D | \forall R.C | \exists R.C | \leq nS.C | \geq nS.C$$

where A is an atomic concept, $R, S \in \mathbf{R}^*$ with S being simple, and n is a non-negative integer. The universal concept $\top$ and the bottom concept $\perp$ can be viewed as $A \sqcup \neg A$ and $\perp$ respectively.

Definition 2.3 (SHIQ Ontology)—A *SHIQ* ontology is a tuple, denoted $K = (T, A)$, where T is the terminology representing general knowledge of a specific domain, and A is the assertional knowledge representing a particular state of the terminology.

The terminology T of ontology K is the disjoint union of a finite set of role inclusion axioms (i.e. $R_1 \sqsubseteq R_2$) and a set of concept inclusion axioms in the form of $C \equiv D$ and $C \sqsubseteq D$, where C and D are arbitrary *SHIQ*-concepts. Statements $C \sqsubseteq D$ are called *general concept inclusion* axioms (GCIs), and $C \equiv D$ can be trivially converted into two GCIs as $C \sqsubseteq D$ and $D \sqsubseteq C$.

The assertional part A of K is also known as a *SHIQ*-ABox, consisting of a set of assertions (facts) about individuals, in the form of

$$C(a) \text{ class assertion}$$

$$R(a, b) \text{ role or property assertion}$$

$a \approx b$ inequality assertion

where C is a *SHIQ* concept, $R \in \mathbf{R}^*$, and $a, b \in \mathbf{I}$.

Note that explicit assertion of $a \not\approx b$ is supported in *SHIQ*, while, conversely, explicit assertion of *equality*, i.e. $a \approx b$, is not supported, since its realization relies on equivalence between nominals [12, 38], i.e. $\{a\} \equiv \{b\}$, which is illegal in *SHIQ*.

Definition 2.4 (SHIQ Semantics)—The meaning of an ontology is given by an interpretation denoted $I = (\mathbf{I}, \cdot^I)$, where $\mathbf{I}$ is a non-empty domain and $\cdot^I$ is an *interpretation function*. This interpretation function $\cdot^I$ maps:

1. every atomic concept $A \in \mathbf{C}$ to a set $A^I \subseteq \mathbf{I}$,
2. every individual $a \in \mathbf{I}$ to an element $a^I \in \mathbf{I}$ and,
3. every role $R \in \mathbf{R}$ to a binary relation on the domain, i.e. $R^I \subseteq \mathbf{I} \times \mathbf{I}$.

Interpretation for other concepts and inverse roles are given below:

$$\top^I = \Delta^I$$

$$\perp^I = \emptyset$$

$$\neg C^I = \Delta^I \setminus C^I$$

$$(R^-)^I = \{(y, x) \mid (x, y) \in R^I\}$$

$$(C \sqcap D)^I = C^I \cap D^I$$

$$(C \sqcup D)^I = C^I \cup D^I$$

$$(\exists R.C)^I = \{x \mid \exists y. (x, y) \in R^I \wedge y \in C^I\}$$

$$(\forall R.C)^I = \{x \mid \forall y. (x, y) \in R^I \wedge y \in C^I\}$$

$$(\leq nR.C)^I = \{x \mid |\{y \mid (x, y) \in R^I \wedge y \in C^I\}| \leq n\}$$

$$(\geq nR.C)^I = \{x \mid |\{y \mid (x, y) \in R^I \wedge y \in C^I\}| \geq n\}$$

where $|\cdot|$ represents the cardinality of a given set.

An interpretation I satisfies an axiom $\alpha : C \sqsubseteq D$, if $C^I \subseteq D^I$. Such interpretation is called a *model* of axiom α . An interpretation I satisfies an axiom or assertion α :

$$R_1 \sqsubseteq R_2 \text{ iff } R_1^I \subseteq R_2^I$$

$$C(a) \text{ iff } a^I \in C^I$$

$$R(a, b) \text{ iff } (a^I, b^I) \in R^I$$

$$a \neq b \text{ iff } a^I \neq b^I$$

For an ontology K , if an interpretation I satisfies every axiom in K , I is a *model* of K , written $I \models K$. In turn, ontology K is said *satisfiable* or *consistent* if it has at least one model; otherwise, it is *unsatisfiable* or *inconsistent*, and there exists at least one contradiction in K .

Definition 2.5 (Logical Entailment)—Given an ontology K and an axiom α , α is called a logical entailment of K , denoted $K \models \alpha$, if α is satisfied in every model of K .

Definition 2.6 (Instance checking)—Given an ontology K , a *SHIQ*-concept C and an individual $a \in \mathbf{I}$, instance checking is defined as testing whether $K \models C(a)$ holds.

Notice that instance checking is considered the central reasoning task for information retrieval from ontology ABoxes and a basic tool for more complex reasoning services [11]. Instance checking can also be viewed as “*classification*” of an individual, that is, checking if an individual can be classified as a member of some defined DL concept.

In most logic-based approaches, realizations of reasoning services are based on a so-called *refutation-style proof* [12, 39], that is, to convert a inference problem to the test of ontology satisfiability. For example, to decide if $K \models C(a)$, one can check if $K \cup \{\neg C(a)\} \not\models \perp$ instead, and the answer is *yes* iff $K \cup \{\neg C(a)\}$ is unsatisfiable, and the answer is *no* otherwise.

Ontology reasoning algorithm in current well-known systems (e.g. Pellet, HermiT, FaCT++, and Racer.) are based on (hyper)tableau algorithms [18–21] that try to build a universal

model for the given ontology based on a set of tableau expansion rules. For details of tableau expansion rules and description of a standard tableau algorithm for *SHIQ*, we refer readers to the work in [37].

2.2 Other Definitions

We adopt notations from tableaux [37] for referring to individuals in $R(a,b)$, such that a is called an R -predecessor of b , and b is an R -successor (or R^- -predecessor) of a . If b is an R -successor of a , b is also an R -neighbor of a .

Definition 2.7 (Signature)—Given an assertion γ in ABox A , the *signature* of γ , denoted $Sig(\gamma)$, is defined as a set of named individuals occurring in γ . This function is trivially extensible to $Sig(A)$ for the set of all individuals in ABox A .

Definition 2.8 (Role path)—We say there is a *role path* between individual a_1 and a_n , if for individuals $a_1, \dots, a_n \in \mathbf{I}$ and $R_1, \dots, R_{n-1} \in \mathbf{R}$, there exist either $R_i(a_i, a_{i+1})$ or $R_i^-(a_{i+1}, a_i)$ in A for all $i = 1, \dots, n-1$.

The role path from a_1 to a_n may involve inverse roles. For example, given $R_1(a_1, a_2)$, $R_2(a_3, a_2)$, and $R_3(a_3, a_4)$, the role path from a_1 to a_4 is $\{R_1, R_2^-, R_3\}$, while the opposite from a_4 to a_1 is $\{R_3^-, R_2, R_1^-\}$.

Definition 2.9 (Simple-Form Concept)—A concept is said to be in *simple form*, if the maximum level of nested quantifiers in the concept is less than 2.

For example, given an atomic concept A , both A and $\exists R.A$ are simple-form concepts, while $\exists R_1.(A \sqcap \exists R_2.A)$ is not, since its maximum level of nested quantifiers is two. Notice however, an arbitrary concept can be *linearly* reduced to the simple form by assigning new concept names for fillers of the quantifiers, for example, $\exists R_1.\exists R_2.C$ can be converted to $\exists R_1.D$ by letting $D \equiv \exists R_2.C$ where D is a new concept name.

3. Definition of an ABox Module

The notion of ABox module here is to be formalized w.r.t. ontology semantics and entailments that are only meaningful when the ontology is consistent. In this study, however, instead of restricting ontologies to be consistent, we aim to discuss the problem in a broader sense such that the theoretical conclusions hold regardless the consistency state of an ontology. For this purpose, we introduce the notion of *justifiable entailment* as defined below.

Definition 3.1 (Justifiable Entailment)—Let K be an ontology, α an axiom, and $K \models \alpha$. α is called a justifiable entailment of K , iff there exists a consistent fragment $K' \subseteq K$ entailing α , i.e. $K' \models \alpha$ and $K' \not\models \perp$.

It is not difficult to see that justifiable entailments of an ontology simply make up a subset of its logical entailments. More precisely, for a consistent ontology, the set of its justifiable entailments is exactly the set of its logical ones according to the definition above; while for

an inconsistent ontology, justifiable entailments are those *sound* logical ones that have consistent bases [40]. For example, given an inconsistent ontology $K=\{C(a), \neg C(a)\}$, both $C(a)$ and $\neg C(a)$ are justifiable entailments of K , but $R(a,b)$ is not.

Remark 3.1: Unless otherwise stated, we take every entailment mentioned in this paper, denoted $K \models \alpha$, to mean a justifiable entailment.

Though complete reasoning on a large ABox may cause intractabilities, in realistic applications, a large ABox may consist of data with great diversity and isolation, and there are situations where a complete reasoning may not be necessary. For example, when performing instance checking of a given individual, say instance *CHRNA4* in the biomedical ontology example in Section 1, the ABox contains a great portion of other unrelated matters.

An ideal solution to this ABox reasoning problem is thus to extract a subject-related module and to have the reasoning applied to the module instead. Particularly, to fulfill soundness and completeness, this subject-related module should be a closure of entailments about the given entities, which in DL, should include class and property assertions. This leads to our formal definition of an ABox module as follows:

Definition 3.2 (ABox Module)—Let $K = (T, A)$ be an ontology, and a set S of individuals be a signature. M_S with $M_S \subseteq A$ is called an ABox module for signature S , iff for any assertion γ (either a class or a property assertion) with $Sig(\gamma) \cap S = \emptyset$, $(T, M_S) \models \gamma$ iff $K \models \gamma$.

This definition provides a necessary and sufficient condition for being an ABox module. It guarantees that sound and complete entailments (represented by γ) of individuals in signature S can be achieved by independent reasoning on the ABox module w.r.t T . Class assertions preserved by an ABox module here are limited to atomic concepts defined in T .

Remark 3.2: Limiting the preserved class assertions to only atomic concepts simplifies the problems to deal with in this paper. While on the other hand, it should not be an obstacle in principle to allow this notion of ABox module to be applied for efficient instance retrieval or answering conjunctive queries with arbitrary concepts. This is because we can always assign a new name A for an arbitrary query (possibly complex) concept C by adding axiom $C \equiv A$ into T , and re-extract the ABox modules. Our empirical evaluation shows that the overhead (time for module extraction) could be negligible when comparing with the efficiency gained for ABox reasoning.

Definition 3.2 does not guarantee, however, uniqueness of an ABox module for signature S , since any super set of M_S is also a module for S , due to the *monotonicity* of DLs [12]. For example, given any signature $S \subseteq I$, the whole ABox A is always a module for S .

Note however, the objective of this paper is to extract a precise ABox module and to select only *module-essential* assertions for a signature, so that the resulting module ensures completeness of entailments while keeping a relatively small size by excluding unrelated assertions. Intuitively, for assertions to be module-essential for a signature S , they must be able to affect logical consequences of any individual in S , so that by having all these assertions included, the resulting ABox module can preserve all facts of the given entities.

This criteria for being a module-essential assertion can be formalized based on the notion of *justification* [41] as given below.

Definition 3.3 (Justification)—Let K be an ontology, α an axiom, and $K \models \alpha$. We say a fragment $K' \subseteq K$ is some justification for axiom α , denoted $Just(\alpha, K)$, iff $K' \models \alpha$ and $K'' \not\models \alpha$ for any $K'' \subset K'$.

Definition 3.4 (Module-essentiality)—Let K be an ontology, a be an individual name, and γ an ABox assertion. γ is called module-essential for $\{a\}$, iff

$$\gamma \in Just(\alpha, K) \wedge K \models \alpha,$$

for any assertion α of a (either $C(a)$ or $R(a, b)$, with $C \in \mathbf{C}$ and $R \in \mathbf{R}^*$) that can be derived from K .

A justification $Just(\alpha, K)$ for axiom α is in fact a *minimum* fragment of the knowledge base that implies α , and every axiom (or assertion) in $Just(\alpha, K)$ is thus essential for this implication. Following from this point, an assertion γ is considered able to affect logical consequences of some individual in signature $\mathbf{S}$, if and only if it appears in some justification for either (i) property assertion or (ii) class assertion of that individual. If so, γ is considered module-essential for $\mathbf{S}$. Having all such module-essential assertions for $\mathbf{S}$ included in $M_{\mathbf{S}}$, the ABox module obviously preserves all facts including both class and property assertions of the individuals in $\mathbf{S}$. On the other hand, with this notion of module-essentiality, the ABox module can be kept as *precise* (and as small) as possible by excluding those assertions that are not module-essential for $\mathbf{S}$. In this paper, we aim to compute such a precise and small ABox module for a given signature $\mathbf{S}$, which would ideally consist of only module-essential assertions.

In the following sections, a theoretical approach and an approximation are presented for the computation of an ABox module. Without loss of generality, we assume all ontology concepts are in simple form as defined previously, and concept terms in all class assertions are atomic.

We will show how to compute an ABox module in two steps: we begin by showing module extraction in an *equality-free SHIQ* ontology, and later we show how the basic technique can be extended to deal with equality. Notice that, this division is only for presentation purpose, and does not have to be considered when using our method for module extraction in practice.

4. ABox Modules in Equality-Free Ontologies

An ontology is called equality-free, if reasoning over this ontology does not invoke any procedure for individual identification between named individuals or named and algorithm-generated individuals.

In this section, we concentrate on a method that computes ABox modules in ontologies of this type. To further simplify the problem, we will consider module extraction for a single

individual instead of an arbitrary signature $\mathbf{S}$, since the *union* of modules of individuals in $\mathbf{S}$ yields a module for $\mathbf{S}$, as indicated by the following proposition.

Proposition 4.1—*Let $\mathbf{S}$ be a signature, $M_{\{i\}}$ be an ABox module for each individual $i \in \mathbf{S}$, and*

$$M_{\mathbf{S}} = \bigcup_{i \in \mathbf{S}} M_{\{i\}}.$$

$M_{\mathbf{S}}$ is an ABox module for $\mathbf{S}$.

4.1 Strategy

Basically, to compute an ABox module as precise as possible for a given individual a , we have to test every assertion in A to see if it is module-essential for a . That is, for every assertion we have to test whether it contributes in deducing any property assertion or class assertion of individual a .

As a fact of *SHIQ*, deducing class assertions (classification) of an individual usually depends on both of its class and property assertions. While conversely, the tableau-expansion procedure for *SHIQ* indicates that deduction of a property assertion between different named individuals in A should not be affected by their class assertions, except via *individual equality* [37] as discussed in Section 5. This is consistent with the tree-model property of the description logic that, if nominals are not involved in the knowledge base, no tableau rules can derive connection (i.e. property assertion) from individual a to an arbitrary named individual, except to itself in the presence of (local) reflexivity [42].

Therefore, given an equality-free ontology, the above observation allows us to deduce property assertions from the ABox w.r.t. role hierarchy and transitive roles defined in T [37], and a strategy for extracting an ABox module can then be devised.

Proposition 4.2—*Given an equality-free *SHIQ* ontology, computation of an ABox module for individual a can be divided into two steps:*

1. *compute a set of assertions (denoted $M_{\{a\}}^P$) that preserves any property assertion $R(a, b)$ of a , with $a \not\approx b$ and $R \in \mathbf{R}^*$,*
2. *compute a set of assertions (denoted $M_{\{a\}}^C$) that preserves any class assertion $C(a)$ of a , with $C \in \mathbf{C}$.*

For simplicity, we call $M_{\{a\}}^P$ a *property-preserved module* and $M_{\{a\}}^C$ a *classification-preserved module* for $\{a\}$.

4.2 Property-Preserved Module

A property-preserved module of individual a is essentially a set of assertions in ontology K , which affect the deduction of a 's property assertions. This set is denoted $M_{\{a\}}^P$ such that

$$M_{\{a\}}^P = \{\gamma \mid \gamma \in \text{Just}(R(a, b), K) \wedge K \models R(a, b)\},$$

where γ is an ABox assertion, and $\text{Just}(R(a, b), K)$ is any justification for $R(a, b)$, with $a \not\approx b$.

In an equality-free *SHIQ* ontology, since property assertions between different individuals can be deduced from the ABox A w.r.t. role hierarchy and transitive roles, the computation for $M_{\{a\}}^P$ is then straightforward based on the following fact: for any $R \in \mathbf{R}^*$, if $K \models R(a, b)$ with $a \not\approx b$, there are two possibilities on an equality-free *SHIQ* ontology, according to [37]:

1. assertion $R_0(a, b)$ or $\text{Inv}(R_0)(b, a) \in A$ with $R_0 \sqsubseteq R$,
2. assertions involved in a role path from a to b , with all roles having a common transitive parent R_0 and $R_0 \sqsubseteq R$. e.g. $R_1(a, a_1), R_2(a_2, a_1), R_3(a_2, b) \in A$, with $R_1, R_2^-, R_3 \sqsubseteq R_0$ and R_0 is transitive.

Abstracting from a particular R_0 , these two possibilities can be generalized into a formal criteria to select assertions to include in $M_{\{a\}}^P$ for individual a :

- C1: property assertions in A that have a as either subject or object, or
- C2: property assertions in A that are involved in a role path from a to some b , with all roles in the path having a common transitive parent.

Proposition 4.3—*On an equality-free SHIQ ontology, the set of property assertions satisfying criteria C1 or C2 forms a property-preserved module $M_{\{a\}}^P$ for individual a .*

4.3 Classification-Preserved Module

To compute a precise ABox module $M_{\{a\}}$, we need to further decide a set of assertions that affect classifications of the individual, and this set is denoted $M_{\{a\}}^C$ such that

$$M_{\{a\}}^C = \{\gamma \mid \gamma \in \text{Just}(A(a), K) \wedge K \models A(a)\},$$

where γ is an ABox assertion, A is an atomic concept, and $\text{Just}(A(a), K)$ is any justification for $A(a)$.

As previously stated, in *SHIQ* an individual is usually classified based on both its class and property assertions in A . It is obvious that explicit class assertions of a form an indispensable part of $M_{\{a\}}^C$. Then, to decide any property assertion of a that affects its classification, we examine each assertion captured in $M_{\{a\}}^P$.

The decision procedure here is based on the idea that instance checking is reducible to concept subsumption [43–45], i.e.

Given an ontology $K = (T, A)$, an individual a and a $SHIQ$ -concept C , a can be classified into C , if the concept behind a 's assertions in the ABox is subsumed by C w.r.t. T .

This idea automatically lends itself as a methodology, such that to determine any assertion of an individual that contributes to its classification, we have to decide if the concept behind this assertion is subsumed by some concept w.r.t. T .

Consider the following example: Let an ontology $K = (T, A)$ be

$$(\{\exists R_0. B \sqsubseteq A\}, \{R_0(a, b), B(b)\}),$$

and let us ask whether $R_0(a, b)$ is essential for individual a 's classification or not. To answer that, we need to decide if the concept behind this property assertion is subsumed by some named concept w.r.t. T , i.e. to test if

$$K \models \exists R. C_1 \sqsubseteq C_2 \quad (1)$$

for some named concept C_2 , with $R_0 \sqsubseteq R$ and $C_1(b)$ entailed by the ontology. It is easy to see (1) is satisfied in this example by substituting B for C_1 and A for C_2 . We can thus determine $R_0(a, b)$ is one of the causes for the entailment $C_2(a)$ (i.e. it is in some justification for $C_2(a)$), and should be an element of $M_{\{a\}}^C$. Moreover, assertions in $Just(C_1(b), K)$ should also be elements of $M_{\{a\}}^C$, since $C_1(b)$ is another important factor to the classification $C_2(a)$.

The above example illustrates a simple case of a single property assertion affecting classification of an individual. Classification of an individual can also be caused by multiple assertions. For example, let K be

$$(\{\exists R_0. B \sqcap \exists R_1. C \sqsubseteq A\},$$

$$\{R_0(a, b), R_1(a, c), B(b), C(c)\}).$$

Here, $R_0(a, b)$ is still essential for the deduction $A(a)$. But when testing the subsumption in (1) for $R_0(a, b)$, it will be found unsatisfied.

Thus, in order to comprehensively and completely include other information about the individual, condition (1) should be generalized into:

$$K \models \exists R. C_1 \sqcap C_3 \sqsubseteq C_2 \quad (2)$$

where all other information of individual a is summarized and incorporated into a concept C_3 with $C_3 \not\sqsubseteq C_2$.

Moreover, taking the number restrictions in $SHIQ$ into consideration, condition (2) can be further generalized as:

$$K \models \geq nR.C_1 \sqcap C_3 \sqsubseteq C_2 \wedge |R^K(a, C_1)| \geq n, \quad (3)$$

where C_2 is a named concept, $C_1(b)$ and $C_3(a)$ are entailed by K , $\exists R.C_1$ is only a special case of $\geq nR.C_1$, and $R^K(a, C_1) = \{b_i \in \mathbf{I} \mid K \models R(a, b_i) \wedge C_1(b_i)\}$ denotes the set of distinct R -neighbors of individual a in C_1 .

In general, condition (3) indicates that in an equality-free *SHIQ*-ontology, for any property assertion $R(a, b)$ to affect classification of individual a , the corresponding concept must be subsumed by some named one, and for any (qualified) number restrictions, the number of distinct R -neighbors of a should be no less than the cardinality required. With this condition derived, we are now in a position to present a procedure for computation of $M_{\{a\}}^C$ and also an ABox module for individual a , which is summarized in Figure 1.

4.4 An Approximation for Module Extraction

Computation of $M_{\{a\}}^P$ depends on the complete role hierarchy, which should be computable using a DL-reasoner, since roles in *SHIQ* are atomic and, most importantly, the size of T should be much smaller than A in realistic applications [23]. On the other hand, it is difficult to compute $M_{\{a\}}^C$, since it demands computation of both concept subsumption (i.e. condition (3)) and justifications for class assertions (i.e. $Just(C_1(b), K)$). Simple approximations for both are given in this section as follows.

Definition 4.1 (Approximation of (3))—A syntactic approximation for condition (3) for $R_0(a, b)$ is that: to test if K contains any formula in the form as listed below:

$$\exists R.C_1 \bowtie C_3 \sqsubseteq C_2$$

$$|C_1 \sqsubseteq \forall R^-.C_2 \bowtie C_3| \quad (4)$$

$$|\geq nR.C_1 \bowtie C_3 \sqsubseteq C_2 \wedge |\{b_i \mid R(a, b_i) \in A\}| \geq n|$$

where $R_0 \sqsubseteq R$, $C_i \in \mathbf{C}$, $\bowtie$ is a place holder for $\sqcup$ and $\sqcap$. Also note the following equivalences:

$$\exists R.C \sqsubseteq D \equiv \neg D \sqsubseteq \forall R.\neg C$$

$$\geq nR.C \sqsubseteq D \equiv \neg D \sqsubseteq \leq (n-1)R.C.$$

For assertion $R_0(a, b)$, the approximation for condition (3) is to check if any formula in K is in the form of any listed axioms in (4). If it is yes, $R_0(a, b)$ may *potentially* affect some logical entailment of individual a , and related assertions will be added into a 's ABox

module to ensure preservation of this potential entailment. Validity of this approximation is shown by the following proposition.

Proposition 4.4: *Let $K = (T, A)$ be a SHIQ ontology with simple-form concepts only, $nR_0.B$ and C be SHIQ concepts, and D a named concept. If*

$$K \models \geq nR_0.B \sqcap C \sqsubseteq D \quad (5)$$

with $C \not\sqsubseteq D$, there must exist some formula in T in the form as listed in (4) for some R with $R_0 \sqsubseteq R$.

Proposition 4.4 shows the completeness of the approximation (4). We can thus conclude that an ABox module resulting from this approximation is still able to capture complete classifications (w.r.t. T) of the given individual, which are derivable from its property assertions. The following statement is an immediate consequence of this conclusion: if $C_1 \sqsubseteq \top$, an approximation for the set of assertions in some $Just(C_1(b), K)$ is $M_{\{b\}}$, an ABox module for b , which is computed using the same strategies described here.

Procedure for this approximation is then summarized in Figure 2.

5. Module Extraction with Equality

In this section, we show how the outcome from the previous section can be utilized to tackle module extraction with individual equality.

In SHIQ, individual equality stems from the at-most number restriction (i.e. $< n.R.C$) [37], such that, if individual x belongs to a concept $n.R.C$ while it has more than n entailed R -neighbors in C , then at least two of these R -neighbors could be identical.

This means that, the determination of individual equality requires computation of both property and class assertions of related individuals. Besides, the strategy proposed in Proposition 4.2 becomes infeasible, since property assertions can be derived from equalities (e.g. given $y \approx z$, $R(y, w)$ simply implies $R(z, w)$). In other words, with equality, the computation of $M_{\{a\}}^P$ may be dependent on that of $M_{\{a\}}^C$.

To address this, we present a procedure for extraction of ABox modules, which first modularizes the ABox as if it were equality-free, and then resolves equality through post-processing.

Proposition 5.1—*Let individual $x \in nR.C$ have more than n entailed R -neighbors in C , of which two, y and z , can be determined to be equal (i.e. $y \approx z$). Let signature $\mathbf{S}$ consist of x and all its R -neighbors in C , and*

$$M_S = \bigcup_{i \in \mathbf{S}} M_{\{i\}}$$

where $M_{\{i\}}$ is an ABox module for each individual $i \in \mathbf{S}$ in the “equality-free” ABox. M_S preserves the equality $y \approx z$.

Proposition 5.1 suggests a strategy to retain equality between y and z , by combining “modules” of related individuals. With $y \approx z$ preserved, M_S automatically preserves all facts of y and z that are derived from the equality. Subsequently, for neighbors of y and z (i.e. individuals in $Sig(M_{\{y\}}^P)$ and $Sig(M_{\{z\}}^P)$), modules of these entities should be combined with the M_S obtained above, so that their facts derivable from $y \approx z$ can also be captured. This strategy to retain equality in ABox modules should be applied *recursively* for all identities.

Notice, however, that the strategy in Proposition 5.1 is based on the condition that individuals y and z be known to be equal in the first place, which cannot be assured without ABox reasoning. Nevertheless, conceiving that equality in *SHIQ* stems from number restrictions [37], a simple approximation for it is given in the definition below.

Definition 5.1: Let x, y and z be named individuals, y, z be R -neighbors of x , and $y \not\approx z$ not hold explicitly¹, y and z are considered **potential equivalents**, if their R -predecessor x :

1. has no potential equivalents, and has m R -neighbors with $m \geq n$, or
2. has a set of potential equivalents, denoted X (which includes x), and there exists a

$$\text{set } S \in \binom{X}{m' - n' + 1}, \text{ such that}$$

$$\max_S |\{y_i | R(x_i, y_i) \in A \wedge x_i \in S\}| = m \geq n. \quad (6)$$

where R is used in number restrictions as in the axiom listed in (4), and n is the minimum of the set $\{k | (k+1)R.C \text{ in l.h.s. of GCIs}\}$. $\binom{X}{m' - n' + 1}$ ² denotes the set of all $(m' - n' + 1)$ -combinations of set X , and variables m' and n' are for identification of x that correspond to m and n above, respectively.

In this definition, for $K \models nR.C(x)$ to be possible it is required that $nR.C$ or its isoform occur in the r.h.s. of GCIs in T (or $(n+1)R.C$ in l.h.s. as in (4)), proof for which is similar to the one given for Proposition 4.4.

Moreover, if x itself has potential equivalents, individuals y and z should be elements of $\{y_i | R(x_i, y_i) \in A \wedge x_i \in X\}$. For the counting of potential R -neighbors of x , instead of taking the entire set X , only the $(m' - n' + 1)$ -combination of X that maximizes the counting is considered (see (6)), since given m' R' -neighbors, the maximum possible number of decidable identical entities is $(m' - n' + 1)$ out of m' .

Proposition 5.2—Applying the strategy in Proposition 5.1 to potential equivalents generates modules that preserve individual equality.

¹Either $y \not\approx z$ is not explicitly given, or assertions $C(y)$ and $\neg C(z)$ do not occur simultaneously in A .

²If $(m' - n' + 1) \leq |X|$, it denotes $\{X\}$.

6. Related Work

For ABox reduction or partition, the problem has been addressed mainly by [25], [26], [46] and [27].

The idea of [25] is to reduce large ABoxes by combining similar individuals (w.r.t. individual description) to develop a summary ABox as the proxy, which is small enough for scalable reasoning. This ABox summary can be useful in some scenarios that involve general testing of consistency of an ontology, but it has a limited capability to support ontology queries such as instance retrieval.

In [26], the authors proposed a method for computing independent ABox partitions for *SHIF* ontologies, such that complete reasoning can be achieved if combining results of independent reasoning on each partition. This method for ABox partitioning is based on a set of inference rules in [31], and it encapsulates assertions that are antecedents of a inference rule as an ABox partition. This technique is also used by [47] to handle reasoning on large data sets.

ABox partition is also used in [46], where the authors proposed an algorithm for *SHIQ* ontologies. Based on the technique in [28] that converts DL ontologies to disjunctive datalog programs, their algorithm further converts the disjunctive datalog program into a plain datalog program, to generate rules that can be employed as guidelines for ABox partition.

Both approaches above for ABox partition, however, did not impose any logical restrictions on a single partition. An immediate consequence is that every single assertion can turn out to be a partition if the ontology is too simple, i.e. has no transitive roles nor concepts defined upon restrictions. Besides, to get complete entailments of an individual, one still has to reason over all partitions of the ontology.

The technique presented in [27] is focused ABox modularization for *SHI* ontologies. In their paper, instead of imposing any specification on a single ABox module, the authors provided a formal definition directly for ABox modularization that extends the notion of ABox partition from [26]. A briefly summarized version of their definition is given below.

Definition 6.1 (ABox Modularization [27])—Given ontology $K = (T, A)$, an ABox modularization M is defined as a set of ABoxes $\{A_1, \dots, A_n\}$, where each $A_i \subseteq A$ is an ABox module. M is said sound and complete for instance retrieval if, given any class assertion $C(a)$ (C is atomic), $\exists A_i \in M$ that $(T, A_i) \models C(a)$ iff $K \models C(a)$.

In [27], the general idea for ABox modularization is based on connected components in an ABox graph. The authors presented an *ABox-Split* based approximation, such that after applying ABox-split on each $R(a, b) \in A$, every connected component in the resulted ABox graph forms an ABox module, and the set of all connected components forms the modularization defined above.

More precisely, this ABox-split technique checks every property assertion $R(a,b)$ in an ABox and will replace it with two generated ones, $R(a',b)$ and $R(a,b')$, if all the conditions below are satisfied for $R(a,b)$ [27]:

1. R is neither transitive nor has any transitive parent.
2. For every $C \in \text{extinfo}_T^\forall(R)$, it satisfies that: (i) $C \equiv \perp$, or (ii) there exists $D(b) \in A$ such that $K \models D \sqsubseteq C$, or (iii) there exists $D(b) \in A$ such that $K \models D \sqcap C \sqsubseteq \perp$.
3. For every $C \in \text{extinfo}_T^\forall(R^-)$, it satisfies that: (i) $C \equiv \perp$, or (ii) there exists $D(a) \in A$ such that $K \models D \sqsubseteq C$, or (iii) there exists $D(a) \in A$ such that $K \models D \sqcap C \sqsubseteq \perp$.

where $C \in \text{extinfo}_T^\forall(R)$ is the set of concepts in T that are used as fillers of R and are able to propagate through the $\forall$ -rule [37] in the tableau algorithm.

In general, this ABox-split based approximation will replace an original property assertion $R(a,b)$ with two generated ones, if it can be determined that, either (i) $R(a,b)$ has no influence on hidden implications of a nor b , or (ii) its influence relies on individual's classification that has already been explicitly given. This approximation hence “splits” assertion $R(a,b)$ and separates a and b into different connected components (ABox modules) in ABox graph. For example, given ontology $K = \{R(a,b), \forall R.B(a), B(b)\}$, $B(b)$ is entailed by K by either the explicitly given assertion or $\{R(a,b), \forall R.B(a)\}$. Nevertheless, since $B(b)$ is explicitly given, this approximation can separate individual a and b into two ABox modules.

To sum up, the method by [27] aims at computing ABox modules as small as possible for *SHI*-ontologies; and it makes use of information from the concept hierarchy (including concept subsumption and concept disjointness, where complex concepts may be involved) to rule out assertions for duplicate or impossible implications, which thus requires invocation of a DL reasoner. Moreover, this approach requires a consistent ontology as the prerequisite.

7. Empirical Evaluation

We implemented our approximation using Manchester's OWL API³, and evaluate it on a lab PC with Intel(R) Xeon(R) 3.07 GHz CPU, Windows 7, and 1.5 GB Java heap. For test data, we collected a set of ontologies with large ABoxes:

1. VICODI⁴ is an ontology that models European history with a simple TBox;
2. LUBM(s) are well-known benchmark ontologies generated using tools provided by [48];
3. Arabidopsis thaliana (AT) and Caenorhabditis elegans (CE) are two biomedical ontologies⁵ based on a complex TBox⁶ that models biological pathways; and

³<http://sourceforge.net/projects/owlapi>

⁴<http://www.vicodi.org>

⁵<http://www.reactome.org/download>

⁶<http://www.biopax.org>

4. DBpedia* ontologies are acquired from the original DBpedia ontology⁷ [49]. They have a common terminological part T , DL expressivity of which is extended from ALF to $SHIQ$ by adding transitive roles, role hierarchy, and concepts defined on role restrictions, and their ABoxes are generated by randomly sampling the original ABox.

Details of these ontologies are summarized in Table 1, in terms of expressiveness (Exp.), number of assertions (#Ast.), and number of individuals (#Ind.).

7. 1 Evaluation of Extracted Modules

These collected ontologies were modularized using the approximation for module extraction for every named individual. As discussed previously, extracting ABox module for a single individual results in a module whose signature is constituted by a set of individuals because of module combination.

In Table 1, we show the statistics of maximum and average module size in terms of number of assertions (#Ast.) and size of signature (#Ind.). It can be observed that, in average, modules of all these ontologies are significantly smaller as compared with the entire ABox. In some ontologies, the maximum module size is relatively large, either because there is a great number of assertions of a single individual, or because there are indeed intricate relationships between individuals that may affect classification of each other.

VICODI is a simple ontology that has no property assertions in the ABox with condition (4) satisfied, and thus, the signature of every ABox module is constituted by only a single entity. However, its maximum ABox module consists of more than 8000 assertions for a single individual.

For the biomedical ontologies AT and CE, their maximum ABox modules are large and complex, mainly because: (i) the terminological part of these ontologies is highly complex, with 33 out of 55 object properties either functional/inverse functional or used as restrictions for concept definition; and (ii) these ontologies also have single individuals that each has a great number of property assertions (e.g. AT has one individual with 8520 assertions). Thus, connections between these individuals by any property assertions satisfying condition (4) result in large ABox modules.

For LUBM-1 and LUBM-2, the sizes of their maximum modules are between of these two categories above, due to moderate complexity of the ontologies and the fact that only 4 out of 25 object properties are used for concept definition.

For DBpedia* ontologies, though their terminological part T is extended from the original ALF to $SHIQ$, the terminology is still simple, since the extension to $SHIQ$ is limited and most of its concepts are not defined on restrictions. Nonetheless, maximum ABox modules of these ontologies are even larger, and it is mainly because of those individuals that each has huge number of property assertions (e.g. DBpedia*_1 has one individual with 40773 assertions, and DBpedia*_2 has one individual with 60935 assertions, etc.), and some of

⁷<http://wiki.dbpedia.org/Ontology?v=194q>

their property assertions happen to satisfy the condition (4) that cause a great number of module combinations.

As shown in Table 2, most of the ABox modules (more than 90%) in these ontologies, are small and simple with no greater than ten assertions or with a single individual in signature. For modules with more than one individual in signature, we plot distributions of signature sizes of these ABox modules in Figure 4 for LUBM-1, LUBM-2, AT, CE, DBpedia*_1 and DBpedia*_3. The X-axis gives the size range and the Y-axis gives the number of modules in each size range. Because of a large span and uneven distribution of module sizes, we use non-uniform size ranges, so that we can have a relatively simple while detailed view of distributions of small, medium, and large modules. It can be observed from the figure that: (i) for all these ontologies, the majority of ABox modules are still small with no more than five individuals in signature, (ii) LUBMs have more medium modules that have signature size between 50 and 600 due to moderate ontology complexity, and (iii) the biomedical ontologies have almost all modules with signature size below 200 but one with very large signature (more than 1,000 individuals), and similar situations are also found in DBpedia* ontologies. Those large ABox modules in both biomedical and DBpedia* ontologies could be caused by complexity of the ontology as discussed above.

7.2 Optimization and Comparison

In this section, we discuss several simple optimization techniques that can be applied to further reduce the size of an ABox module.

As discussed previously, a precise ABox module for signature $\mathbf{S}$ should consist of assertions in justifications for entailments about individuals in $\mathbf{S}$, i.e. module-essential assertions. Considering there may be more than one justification for a single entailment [41], an intuition for the strategy to reduce sizes of ABox modules is to exclude redundant justifications for the same entailments.

For example, consider an ontology K that entails

$$\exists R.B \sqsubseteq A, B(b_1) \text{ and } B(b_2),$$

and the ABox contains

$$R(a, b_1) \text{ and } R(a, b_2).$$

To preserve the fact $A(a)$ while excluding redundant justifications for a single entailment, an ABox module for individual a should include either $R(a, b_1)$ with $\text{Just}(B(b_1), K)$ or $R(a, b_2)$ with $\text{Just}(B(b_2), K)$, but not both. Additionally, instead of considering all $\text{Just}(B(b_1), K)$ s (resp. $\text{Just}(B(b_2), K)$ s), taking a single justification for $B(b_1)$ (resp. for $B(b_2)$) should be sufficient.

Therefore, to exclude redundant justifications for the same entailments from an ABox module, beyond testing the condition (3), i.e.

$$K \models_{\geq} nR.C_1 \sqcap C_3 \sqsubseteq C_2 \wedge |R^K(a, C_1)| \geq n$$

for every property assertion $R_0(a, b)$ when computing $M_{\{a\}}^C$, it is also necessary to consider:

Case1. if any justification for $C_2(a)$ has already been added into the module, or

Case2. if there is any justification for $C_1(b)$ that can be easily obtained.

In *Case1*, if the module already preserves the fact $C_2(a)$, redundant assertions for the same entailment will not be included; in *Case2*, instead of adding all $Just(C_1(b), K)$ s to a 's module, we should first consider if there is any one that is easy to compute. Based on these two cases, we implemented three simple optimized approximations for evaluation and comparison:

Opt1. Based on *Case1*, if $C_2(a)$ can be entailed from explicit class assertions of a , modules of a and b will not be merged.

Opt2. Based on *Case2*, if $B(b)$ can be entailed from explicit class assertions of b , instead of merging modules of a and b , only explicit class assertions of b will be added into a 's module. If we preprocess every axiom in T to the form of $nR.C_1 \sqcap C_3 \sqsubseteq C_2$ as described in the proof for Proposition 4.4, this optimization coincides with the one that can be obtained by extending the method in [27].

Opt3. The combination of Opt1 and Opt2.

It is obvious that the first two strategies, Opt1 and Opt2, should have no advantage over each other, and their performances will mostly depend on the actual ontology data to which they are applied. More precisely, if *Case1* prevails in the ontology, Opt1 is expected to generate smaller ABox modules than that of Opt2, and otherwise if *Case2* prevails. Nonetheless, Opt3 should always have better or no-worse performance than that of Opt1 and Opt2, as it takes advantage of both of them.

In general, these three optimized approximations are expected to outperform the original approximation and produce smaller ABox modules, since they all employ a DL-reasoner to rule out redundant assertions for the same entailments. Indeed, when an ontology is simple and contains considerable redundant implications, these methods are able to efficiently reduce sizes of ABox modules; while on the other hand, when ontologies are complex or with less explicit redundant information, these methods may not provide significant size reductions.

Evaluation and comparison for these three optimized approximations are shown in Table 3. For LUBM ontologies, all these optimized approximations efficiently reduce the size of ABox modules (especially the maximum one), and particularly, Opt3 reduces signature of every ABox module to include only a single individual. For DBpedia* ontologies, Opt1 or Opt2 achieve only limited size reduction for the maximum ABox modules, while their combination, i.e. Opt3, is able to decrease the size of maximum ABox modules significantly. Nevertheless, for the biomedical ontologies, which are complex and may

contain few explicit duplicate entailments, none of the three optimization strategies produced reductions in module size.

7.3 Reasoning with ABox Modules

In this section, we show the efficiency of ontology reasoning gained when reasoning is based on ABox modules (*modular reasoning*), and compare it with that of the complete ABox reasoning. Notice however, the purpose here is not to compare modular reasoning with those developed optimization techniques (e.g. lazy unfolding, and satisfiability reuse etc. [18, 50]) in existing reasoners, since they are in totally different categories and modular reasoning still relies on the reasoning services provided by current state-of-art reasoners. Instead, what we aim to show here is in fact the improvement in time efficiency that can be achieved when using the modular reasoning for instance checking or instance retrieval on top of existing reasoning technologies. The reasoner we used here is HermiT [18, 42], which is one of the fully-fledged OWL 2 reasoners.

For evaluation, we extract ABox modules using the Opt3 discussed in previous section and run the reasoning on each collected ontology: we randomly select 10 concepts that are defined on role restrictions from the testing ontology, and for each one of them we perform the instance checking (on every individual in the ontology) and retrieval using modular reasoning and complete reasoning, respectively. Table 4 details the reasoning time for both instance checking (IC) and instance retrieval (IR). Particularly, for instance checking using modular reasoning, we show both maximum and average reasoning time over the 10 test concepts, since sizes of ABox modules may vary greatly and affect the reasoning efficiency. For instance retrieval, we simply report the average time. Besides, we also set a threshold (24 hours) for the time-out, and if it happens to any one of the test, we simply put a "-" in the corresponding table entry.

Notice that, for fairness the modular reasoning for instance retrieval performed here is not parallelized, but instead running in an arbitrary sequential order of ABox modules. Nevertheless, we can still see the great improvement on time efficiency when using the modular reasoning for instance retrieval as shown in Table 4. For example, the average time for instance checking has been reduced significantly from seconds down to several milliseconds when using the modular reasoning on LUBM(s) and DBpedia*(s), and instance retrieval time on LUBM(s) (respectively on DBpedia*(s)) has been reduced from several hours (respectively several days) down to seconds (respectively less than two hours); even for those biomedical ontologies, the time for instance retrieval is also reduced from more than 130 hours down to less than 6 hours. The reason behind all these improvements is simply that the complexity of the tableau-based reasoning algorithms is up to exponential time w.r.t. the data size for *SHIQ* [13, 51]; once the data size is cut down, the reasoning time could be reduced significantly.

One point to note here is that, when using the modular reasoning for answering object queries, the time for instance retrieval in fact should plus the time that is spent for modularizing the ABox. Nevertheless, as we can see from Table 1, the overhead on average is only around one millisecond for instance checking of each named individual.

8. Discussion and Outlook

In this paper, we have proposed a formal definition of an ABox module, such that each module ensures complete preservation of logical entailments for a give set of individuals. Utilizing this property, scalable object queries over big ontologies can be accomplished, by either conducting an isolated reasoning on a single ABox module when querying information about a particular group of individuals, or by distributing independent reasoning tasks on ABox modules into a cluster of computers when performing instance retrieval or answering conjunctive queries. To put more restrictions on an ABox module, we further defined the notion of module-essentiality, which gives a formal criteria for ABox assertions to be semantically related (or unrelated) to a given signature and could be useful for computation of a precise ABox module.

To extract an ABox module, we presented a theoretical approach, which is only aimed to give an exposition of the problem from a theoretical perspective and to provide strategies that are provable theoretically sound. For applicability in realistic ontologies, we provided a simple and tractable approximation, which is straightforward and easy to implement, but may include many unrelated assertions when the ontology is complex, thus resulting in modules that are larger than desired. The undesired bulk of ABox modules could be caused by: (i) the syntactic approximation (i.e. (4)) for semantic conditions (i.e. (3)) that could result in many false positives and hence cause the unnecessary combination of ABox modules; (ii) the simple approximation of individual equalities, which only checks syntactically the possibility of an individual to be an instance of some concept $nR.C$ and the approximated number of its R -neighbors that should be no less than the associated cardinality required for individual identification; and (iii) last but not least, the intrinsic complexity of ontologies that requires assertions to be grouped to preserve complete logical entailments.

Strategies for optimization can provide significant reduction in module size under some conditions, as has been shown in the results section. However, for highly complex ontologies like those biomedical ones, more advanced optimization techniques are demanded. One of the directions for future optimization is obviously to provide more rigorous approximations for condition (3) and individual equality, and a possible solution is to add affordable semantical verification of individual classifications, which may not only prevent unnecessary module combination but also rule out false individual equalities. Progress in this direction has already been made in our current projects.

Acknowledgements

This work is partly supported by grant # R44GM097851 from the National Institute of General Medical Sciences, part of the U.S. National Institutes of Health (NIH). We'd like to thank Saminda Abeyruwan and Hande Küçük for their contributions to the project, and anonymous reviewers for their helpful comments.

References

1. Bechhofer S, van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. Owl web ontology language. 2004 <http://www.w3.org/TR/owl-ref/>.

2. Horrocks I, Patel-Schneider P, Van Harmelen F. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*. 2003; 1(1):7–26.
3. Cuenca Grau B, Horrocks I, Motik B, Parsia B, Patel-Schneider P, Sattler U. OWL 2: The next step for owl. *Journal of Web Semantics*. 2008; 6(4):309–322.
4. Demir E, Cary M, Paley S, Fukuda K, Lemer C, Vastrik I, Wu G, D'Eustachio P, Schaefer C, Luciano J, et al. The biopax community standard for pathway data sharing. *Nature biotechnology*. 2010; 28(9):935–942.
5. Visser U, Abeyruwan S, Vempati U, Smith R, Lemmon V, Schürer S. BioAssay Ontology (BAO): a semantic description of bioassays and high-throughput screening results. *BMC bioinformatics*. 2011; 12(1):257. [PubMed: 21702939]
6. Bhatt M, Rahayu W, Soni S, Wouters C. Ontology driven semantic profiling and retrieval in medical information systems. *Journal of Web semantics*. 2009; 7(4):317–331.
7. Iqbal, A.; Shepherd, M.; Abidi, S. Proceedings of Hawaii International Conference on System Sciences (HICSS). IEEE; 2011. An ontology-based electronic medical record for chronic disease management; p. 1-10.
8. Haghighi P, Burstein F, Zaslavsky A, Arbon P. Development and evaluation of ontology for intelligent decision support in medical emergency management for mass gatherings. *Decision Support Systems*. 2012
9. Lee C, Wang M, Chen J. Ontology-based intelligent decision support agent for cmmi project monitoring and control. *International Journal of Approximate Reasoning*. 2008; 48(1):62–76.
10. Horrocks I. Ontologies and the semantic web. *Communications of the ACM*. 2008; 51(12):58–67.
11. Schaerf A. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*. 1994; 13(2):141–176.
12. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P., editors. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press; 2007.
13. Tobies, S. PhD thesis. RWTH Aachen; 2001. Complexity Results and Practical Algorithms for Logics in Knowledge Representation.
14. Horrocks I, Tessaris S. A conjunctive query language for description logic aboxes. *Proceedings of AAAI*. 2000:399–404.
15. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated reasoning*. 2007; 39(3):385–429.
16. Glimm B, Horrocks I, Lutz C, Sattler U. Conjunctive query answering for the description logic. *Journal Artificial Intelligence Research*. 2008; 31:157–204.
17. Ortiz M, Calvanese D, Eiter T. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*. 2008; 41(1):61–98.
18. Motik, B.; Shearer, R.; Horrocks, I. Optimized Reasoning in Description Logics using Hypertableaux. In: Pfenning, F., editor. *Proceedings of Conference on Automated Deduction (CADE)*. Springer; 2007 Jul 17–20. p. 67-83.vol. 4603 of LNAI, (Bremen, Germany)
19. Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*. 2007; 5(2):51–53.
20. Tsarkov D, Horrocks I. FaCT++ description logic reasoner: System description. *Automated Reasoning*. 2006:292–297.
21. Haarslev, V.; Möller, R. *Proceedings of the First International Joint Conference on Automated Reasoning*. Springer; 2001. RACER system description; p. 701-705.
22. Horrocks I, Li L, Turi D, Bechhofer S. The instance store: DL reasoning with large numbers of individuals. *Proceedings of the Description Logic Workshop (DL-2004)*. 2004:31–40.
23. Motik B, Sattler U. A comparison of reasoning techniques for querying large description logic aboxes. *Proceedings of LPAR'06*. 2006:227–241.
24. Haarslev V, Möller R. Optimization strategies for instance retrieval. *Proc. International Workshop on Description Logics (DL)*. 2002; 2

25. Fokoue, A.; Kershenbaum, A.; Ma, L.; Schonberg, E.; Srinivas, K. Proceedings of ISWC-06. Springer; 2006. The summary abox: Cutting ontologies down to size; p. 343-356.
26. Guo, Y.; Heflin, J. International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS). Springer; 2006. A scalable approach for partitioning owl knowledge bases; p. 636-641.
27. Wandelt S, Möller R. Towards abox modularization of semi-expressive description logics. *Applied Ontology*. 2012; 7(2):133–167.
28. Hustadt U, Motik B, Sattler U. Reducing SHIQ- description logic to disjunctive datalog programs. *Proceedings of KR*. 2004; 4:152–162.
29. Groszof, B.; Horrocks, I.; Volz, R.; Decker, S. Proceedings of WWW. ACM; 2003. Description logic programs: Combining logic programs with description logic; p. 48-57.
30. Motik B, Sattler U, Studer R. Query answering for OWL-DL with rules. *Journal of Web Semantics*. 2005; 3(1):41–60.
31. Royer, V.; Quantz, J. tech. rep. Germany: Technische Universitaet Berlin; 1993. Deriving inference rules for description logics: a rewriting approach into sequent calculi.
32. Cuenca Grau B, Horrocks I, Kazakov Y, Sattler U. Just the right amount: Extracting modules from ontologies. *Proceedings of WWW'07*. 2007:717–726.
33. Cuenca Grau, B.; Horrocks, I.; Kazakov, Y.; Sattler, U. Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). AAAI; 2007. A logical framework for modularity of ontologies; p. 298-304.
34. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*. 2008; 51(1):107–113.
35. Xu J, Shironoshita P, Visser U, John N, Kabuka M. Extract ABox Modules for Efficient Ontology Querying. *ArXiv e-prints*. 2013 May. vol. arXiv:1305.4859 [cs.AI].
36. Schmidt-Schauß M, Smolka G. Attributive concept descriptions with complements. *Artificial intelligence*. 1991; 48(1):1–26.
37. Horrocks, I.; Sattler, U.; Tobies, S. Proceedings of Conference on Automated Deduction (CADE). Springer; 2000. Reasoning with individuals for the description logic SHIQ; p. 482-496.
38. Hitzler, P.; Krötzsch, M.; Rudolph, S. Foundations of Semantic Web Technologies. Chapman Hall CRC; 2009.
39. Horrocks I, Patel-Schneider P. Reducing owl entailment to description logic satisfiability. *Journal of Web Semantics*. 2004; 1(4):345–357.
40. Huang, Z.; Van Harmelen, F.; Teije, A. Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). Vol. 19. AAAI; 2005. Reasoning with inconsistent ontologies; p. 454
41. Kalyanpur, A.; Parsia, B.; Horridge, M.; Sirin, E. Finding all justifications of OWL DL entailments; Proceedings of International Semantic Web Conference (ISWC); 2007. p. 267-280.
42. Motik B, Shearer R, Horrocks I. Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*. 2009; 36(1):165–228.
43. Donini, F.; Era, A. Proceedings of CIKM. Baltimor, MD: 1992. Most specific concepts for knowledge bases with incomplete information; p. 545-551.
44. Donini F, Lenzerini M, Nardi D, Schaerf A. Deduction in concept languages: From subsumption to instance checking. *Journal of logic and computation*. 1994; 4(4):423–452.
45. Nebel, B. Reasoning and revision in hybrid representation systems. Vol. 422. Germany: Springer-Verlag; 1990.
46. Du, J.; Shen, Y. Proceedings of Description Logics (DL-07). Citeseer; 2007. Partitioning aboxes based on converting DL to plain datalog; p. 251-258.
47. Williams G, Weaver J, Atre M, Hendler J. Scalable reduction of large datasets to interesting subsets. *Journal of Web Semantics*. 2010; 8(4):365–373.
48. Guo Y, Pan Z, Heflin J. LUBM: A benchmark for owl knowledge base systems. *Journal of Web Semantics*. 2005; 3(2–3):158–182.
49. Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; Ives, Z. Proceedings of ISWC. Springer; 2007. Dbpedia: A nucleus for a web of open data; p. 722-735.
50. Horrocks, I. The Description Logic Handbook: Theory, Implementation, and Applications. Vol. ch. 9. Cambridge University Press; 2007. Implementation and Optimization Techniques.

51. Donini, F. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press; 2007. ch. Complexity of Reasoning.

-
1. Compute a property-preserved module $M_{\{a\}}^P$ for the given individual a , by following the criteria $C1$ and $C2$ given in Section 4.2.
 2. Add all explicit class assertions of a into $M_{\{a\}}^C$.
 3. For every $R_0(a, b)$ ($R_0 \in \mathbf{R}^*$) captured in $M_{\{a\}}^P$ and any R with $R_0 \sqsubseteq R$, test if the corresponding condition (3) is satisfied. If it is yes, add $R_0(a, b)$, assertions in some $Just(C_1(b), K)$, and any inequality assertions between individuals in $R^K(a, C_1)$ into $M_{\{a\}}^C$.
 4. Unite the sets, $M_{\{a\}}^P$ and $M_{\{a\}}^C$, to form an ABox module for a .
-

Figure 1.

Steps for computation of an ABox module for individual a .

-
1. Compute a property-preserved module $M_{\{a\}}^P$ for the given individual a , by following the criteria $C1$ and $C2$ given in Section 4.2.
 2. Add all explicit class assertions of a into $M_{\{a\}}^C$.
 3. For every $R_0(a, b)$ captured in $M_{\{a\}}^P$ and any R with $R_0 \sqsubseteq R$, test if K contains any formula in the form as listed in (4). If it is yes, add $R_0(a, b)$, all assertions in $M_{\{b\}}$, and any inequality assertions between a 's R -neighbors into $M_{\{a\}}^C$.
 4. Unite the sets, $M_{\{a\}}^P$ and $M_{\{a\}}^C$, to form an ABox module for a .
-

Figure 2.

Steps of the approximation for module extraction for individual a .

Figure 3.

Examples for two cases for individual x in Definition 5.1. In figure (a), y_1 and y_2 are inferred to be equal, if $y_1 \not\approx y_2$ doesn't hold. In (b), all $x_i(s)$ are considered potential equivalents if none of them are explicitly asserted different from each other; on the other hand, only two of them can be inferred to be equal, when sufficient information is provided. If x_1 is equated with any one of the others, its R -neighbors (i.e. $y_i(s)$) may be further equated.

Figure 4.

Distributions of signature sizes of ABox modules. X-axis is the size range, and Y-axis is the number of modules.

Table 1

ABox modules in different ontologies

Ontology	Exp.	#Ast.	#Ind.	Max. #Ast./#Ind.	Avg. #Ast./#Ind.	Avg. Extraction Time
LUBM-1	SHI	67,465	17,175	2,921/593	13.1/2.4	1.12 ms
LUBM-2	SHI	319,714	78,579	2,921/593	14.4/2.5	1.22 ms
VICODI	ALH	53,653	16,942	8,591/1	5.3/1	0.28 ms
AT	SHIN	117,405	42,695	54,561/10,870	6.9/1.7	1.06 ms
CE	SHIN	105,238	37,162	49,914/9,315	7.1/1.7	0.60 ms
DBpedia*_1	SHIQ	402,062	273,663	94,862/18,671	2.9/1.1	0.62 ms
DBpedia*_2	SHIQ	419,505	298,103	160,436/17,949	2.8/1.1	0.59 ms
DBpedia*_3	SHIQ	388,324	255,909	140,050/35,720	3.1/1.2	1.80 ms
DBpedia*_4	SHIQ	398,579	273,917	139,422/18,208	2.9/1.1	0.51 ms

Table 2

Small and simple ABox modules in ontologies

Ontology	Total #Modules	#Module with #Ast 10 (%)	#Module with Signature Size = 1 (%)
LUBM-1	7,264	7,210 (99.3)	7,209 (99.2)
LUBM-2	31,361	30,148 (96.1)	31,103 (99.2)
VICODI	16,942	16,801 (99.2)	16,942 (100)
AT	24,606	23,464 (95.4)	23,114 (93.9)
CE	21,305	20,010 (93.9)	19,455 (91.3)
DBpedia*_1	239,758	233,231 (97.3)	237,676 (99.1)
DBpedia*_2	264,079	257,555 (97.5)	261,973 (99.2)
DBpedia*_3	208,401	201,905 (96.9)	206,377 (99.0)
DBpedia*_4	241,451	234,903 (97.3)	239,346 (99.1)

Table 3

Evaluation and Comparison of Optimized Approximations

Ontology	Opt1			Opt2 [27]			Opt3		
	Max. #Ast./#Ind.	Avg. #Ast./#Ind.		Max. #Ast./#Ind.	Avg. #Ast./#Ind.		Max. #Ast./#Ind.	Avg. #Ast./#Ind.	
LUBM-1	773/2	6.8/1.0		773/2	6.8/1.0		732/1	6.8/1.0	
LUBM-2	773/2	7.1/1.0		773/2	7.1/1.0		732/1	7.1/1.0	
AT	54,561/10,870	6.9/1.7		54,561/10,870	6.9/1.7		54,561/10,870	6.9/1.7	
CE	49,914/9,315	7.1/1.7		49,914/9,315	7.1/1.7		49,914/9,315	7.1/1.7	
DBpedia*_1	52,539/13,977	2.9/1.1		87,721/15,258	2.9/1.1		228/25	2.7/1.0	
DBpedia*_2	108,772/10,568	2.8/1.1		76,386/12,058	2.7/1.1		150/18	2.6/1.0	
DBpedia*_3	106,344/32,063	3.1/1.2		63,092/10,385	2.9/1.1		113/29	2.8/1.0	
DBpedia*_4	40,760/8,182	2.9/1.1		85,152/11,366	2.8/1.1		171/18	2.7/1.0	

Table 4

Modular v.s. Complete ABox reasoning

Ontology	Modular		Complete	
	IC Time (ms) Max./Avg.	IR Time Avg.	IC Time (ms) Avg.	IR Time Avg.
LUBM-1	17.00 / 1.93	33.08 s	733.21	3.50 h
LUBM-2	17.00 / 1.91	150.36 s	9,839.22	–
AT	3,120.00 / 344.53	4.09 h	11,378.63	–
CE	3,151.00 / 542.60	5.60 h	10,336.61	–
DBpedia*_1	1,326.00 / 19.10	1.45 h	6,187.01	–
DBpedia*_2	1,497.00 / 20.20	1.67 h	6,948.20	–
DBpedia*_3	3,189.00 / 19.89	1.41 h	6,087.23	–
DBpedia*_4	1,154.00 / 20.00	1.52 h	6,305.41	–