Parallelization of Query Processing over Expressive Ontologies

E. Patrick Shironoshita[1], Da Zhang[2], Mansur R. Kabuka[1,2] and Jia Xu[2]
[1]INFOTECH Soft, Inc.
1201 Brickell Avenue, Suite 220
Miami, Florida 33131, USA
patrick@infotechsoft.com
[2]University of Miami
Coral Gables, Florida 33124, USA

Abstract

Efficient query answering over Description Logic (DL) ontologies with very large datasets is becoming increasingly vital. Recent years have seen the development of various approaches to ABox partitioning to enable parallel processing. Instance checking using the enhanced most specific concept (MSC) method is a particularly promising approach. The applicability of these distributed reasoning methods to typical ontologies has been shown mainly through anecdotal observation. In this paper, we present an analysis method that makes use of random graph theory to show that the enhanced MSC method results in very small, tractable concepts provided that the number of role assertions removed from consideration is large enough. We also present execution time and efficiency of a parallel implementation deployed over computing clusters of various sizes, showing the ability of the method to process instance checking for large scale datasets.

1 Introduction

Description Logics (DL) are increasingly being used to model and represent structured and semi-structured data in different applications (Horrocks, 2008). A core task for DL systems is to provide an efficient way to answer queries over the extensional level of the ontology, that is, to compute answers that are not only asserted, but logically implied by the ontology (Calvanese et al., 2013). Considerable efforts have been dedicated to the optimization of algorithms for query answering in recent years (Calvanese et al., 2013; Möller et al., 2007; Motik et al., 2007). One of the challenges faced in this era of increasing data wealth is to produce responsive results for queries over very large data sets (Priya et al., 2014). However, even as reasoners for very expressive DLs have been created, performing reasoning over very large ABoxes is still prohibitive (Calvanese et al., 2013; Donini, 2003; Glimm et al., 2007).

In the last few years, methods for parallel and distributed reasoning over expressive ontologies have been published (Xu et al., 2013, 2015b; Priya et al., 2014; Wandelt and Möller, 2012); these methods generally seek to make use of fast syntactic checks to generate a set of independent partitions that can be processed in parallel. In (Xu et al., 2015a), a method for instance checking is proposed, combining the work in (Xu et al., 2013, 2015b) with the idea of a most specific concept (MSC) (Nebel, 1990; Donini and Era, 1992; Donini et al., 1994) to move the reasoning task from the very large ABox into a much smaller TBox. Empirical evaluation of the method has shown its ability to perform sound and complete instance checking over $\mathcal{SHI}$ DLs within reasonable time. Moreover, the method is inherently parallelizable, lending itself to implementation within clusters of commodity hardware. In this paper, we examine the enhanced MSC method and its parallelization implementation.

2 The Enhanced Most Specific Concept (MSC) method

2.1 Basic MSC Method

A Description Logics (DL) knowledge base, also referred to as an ontology, is typically defined a tuple, denoted $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where the *terminological* component or TBox $\mathcal{T}$ contains definitions of concepts and roles, and the *assertional* component or ABox $\mathcal{A}$ contains assertions about membership of individuals in concepts and about role relations between individuals. The set of roles, concepts, and individual instances in an ontology are denoted respectively by $\mathbf{R}$, $\mathbf{C}$, and $\mathbf{I}$. The discussion

TBox
Headmaster ⊑ Professor
Professor ⊑ Person
MagicCourse ⊑ Course
Muggle ⊑ ¬Wizard
takesCourse.MagicCourse ⊑ Wizard
∃isHeadOf.School ⊓ Person ⊑ Headmaster
ABox
School(hogwarts)
Professor(albus)
Professor(severus)
MagicCourse(potions)
MagicCourse(transfiguration)
Course(math)
Student(harry)
Muggle(dudley)
isHeadOf(hogwarts, albus)
takesCourse(harry, transfiguration)
taughtBy(transfiguration, albus)
taughtBy(potions, severus)

Table 1: Example ABox

in this paper assumes that the reader is familiar with DL concepts and notations; refer to (Baader et al., 2003) for details. For the discussion below, we will use the example illustrated in Table 1.

Definition 1 (Most Specific Concept) *(Nebel, 1990; Donini and Era, 1992) Let $\mathcal{K} = \{\mathcal{T}, \mathcal{A}\}$ be an ontology, and a be an individual in* **I**. *The most specific concept for a w.r.t. $\mathcal{A}$, written $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$, is a concept such that for every concept D where $\mathcal{K} \models D(a)$, $\mathcal{T} \models \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}.a) \sqsubseteq D$.*

If $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$ can be derived, then, to test whether $\mathcal{K} \models Q(a)$ holds for an arbitrary concept Q it suffices to test if $(\mathcal{T} \cup \{Q\}) \models \mathrm{MSC}_{\mathcal{T}\cup\{Q\}}(\mathcal{A}, a) \sqsubseteq Q$. We call the concept Q the *query*. Note that Q needs to be inserted into the TBox in simple form, which may require in turn the insertion of additional named concepts as well as general concept inclusion (GCI) axioms to express the necessary equivalences for these inserted concepts. In the remainder of this paper, we will assume that the query Q has been inserted into the TBox so that $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$ denotes the MSC calculated considering Q.

Computation of the MSC for a given individual a as defined above can be performed using a *rolling up* procedure adapted from the one first introduced in (Horrocks and Tessaris, 2000).

Definition 2 (Basic MSC Rollup Procedure) *Provided that the ABox does not contain assertion cycles, computation of the MSC can be performed recursively as follows:*

1. *for a given individual a, start with an empty* $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$*;*
2. *for every concept assertion $C(a)$,* $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \sqcap C$*;*
3. *for every role assertion $R(a, b)$,* $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \sqcap \exists R.\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}\backslash\{R(a, b)\}, b)$*;*
4. *for every individual equality assertion $a = a'$,* $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \sqcap \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a')$.

So, in the example ABox above, the MSC for `severus` is

$$\texttt{Professor} \sqcap \exists \texttt{taughtBy}^{-}.\texttt{Course} \quad (1)$$

It is important to note that while class assertions generate relatively simple concepts, role assertions are capable of generating very complex concepts. Suppose for example that ABox $\mathcal{A}_n$ consists of role assertions $R_0(a_0, a_1)$, $R_1(a_1, a_2)$, $\ldots$, $R_n(a_n, a_{n+1})$; then:

$$\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}_n, a_0) = \exists R_1.(\exists R_2.(\ldots(\exists R_n.\mathrm{MSC}_{\mathcal{T}}(a_{n+1}))\ldots)) \quad (2)$$

Thus, in the example ABox of Table 1, the MSC for `harry` is

$$\texttt{Student} \sqcap \exists \texttt{takesCourse}^{-}.(\texttt{Course} \sqcap \exists \texttt{taughtBy}.(\texttt{Professor} \sqcap \texttt{isHeadOf}^{-}.\texttt{School})) \quad (3)$$

Two main issues preclude this basic MSC method from proving fully useful with expressive ontologies. First, if assertion cycles are present in the ABox, the method does not terminate. For example, consider what would happen if

$$\texttt{isProtegeOf(albus,harry)} \quad (4)$$

is inserted into the ABox, then a cycle forms among the individuals `harry`, `transfiguration`, and `albus`.

Second, unless the ABox is highly disconnected, the method has the potential to generate very large concepts, of size in the same order of the ABox itself. Consider what happens if the following assertion is added to the ABox:

$$\texttt{takesCourse(harry,potions)} \quad (5)$$

In this case, all individuals become connected with each other, and the MSC of every individual ends up being the same size of the ABox.

Xu et.al. (2015a) define a set of improvements that result in the *Enhanced MSC Method*. This enhancement consists of two parts: a mechanism to address assertion cycles, and a set of syntactic conditions to reduce the size of the MSCs in practical ontologies.

2.2 MSC Computation with Assertion Cycles

To address assertion cycles, Xu et.al. (2015a) use nominals to indicate the joint node of a cycle, as previously suggested in (Donini and Era, 1992; Schaerf, 1994).

Definition 3 (MSC Rollup of Assertion Cycles) *When a cycle is found starting and ending at individual a_c, the individual is represented by its corresponding nominal class $\{a_c\}$, and the conversion of role assertions within the cycle requires modification of the rollup method as follows: If a_c is an individual where a cycle is found, select a direction to go through the cycle and:*

- *for $R(a_c, x)$, $x \neq a_c$*
  $$\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a_c) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a_c) \sqcap (\{a_c\} \sqcap \exists R.\mathrm{MSC}_{\mathcal{T}}(\mathcal{A} \backslash \{R(a_c, x)\}, x))$$
- *for $R(y, a_c)$, $y \neq a_c$*
  $$\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, y) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, y) \sqcap \exists R.\{a_c\}$$
- *for $R(a_c, a_c)$,*
  $$\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a_c) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a_c) \sqcap \{a_c\} \sqcap \exists R.\{a_c\}$$
- *for any other $R(x, y)$ in the cycle, for $x \neq a_c$ and $y \neq a_c$,*
  $$\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, x) \leftarrow \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, x) \sqcap \exists R.\mathrm{MSC}_{\mathcal{T}}(\mathcal{A} \backslash \{R(x, y)\}, y)$$

Thus, if an ABox

$$\mathcal{A}_c = \{R_0(a_0, a_1), R_1(a_1, a_2), \ldots, R_n(a_n, a_0)\}$$

the MSC of an assertion cycle is obtained as follows:

$$\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}_c, a_0) = \{a_0\} \sqcap \exists R_1.(\exists R_2.(\ldots(\exists R_n.\{a_0\})\ldots)) \quad (6)$$

So, suppose that the ABox in Table 1 is augmented with the assertion in equation (4), then the MSC for `harry` becomes

$$\begin{aligned} &\{\texttt{harry}\} \sqcap \texttt{Student} \sqcap \exists \texttt{takesCourse}^{-}. \\ &\quad (\texttt{Course} \sqcap \exists \texttt{taughtBy}. \\ &\qquad (\texttt{Professor} \sqcap \texttt{isHeadOf}^{-}.\texttt{School} \\ &\qquad\quad \sqcap \texttt{isProtegeOf}^{-}.\{\texttt{harry}\})) \end{aligned} \quad (7)$$

Use of the MSC method to perform instance retrieval is straightforward. Suppose it is desired to query an ABox to retrieve all instances of a concept Q. It suffices to generate $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$ for every individual a in the ABox, and then accept every individual where $(\mathcal{T} \cup Q) \models \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a) \sqsubseteq Q$. Note that the query concept Q is added to the TBox being verified.

2.3 Syntactic Conditions

In (Xu et al., 2015a), syntactic conditions verifiable in polynomial time or better were defined, to enable reduction on the size of the MSC and thus permit TBox reasoning in tractable time. These conditions are based on the following:

Lemma 1 *Given two individuals a and b and a role R, a role assertion $R(a, b)$ influences the classification of individual a into concept A if and only if*

$$\mathcal{K} \models \exists R.B \sqcap A_0 \sqsubseteq A \quad (8)$$

where $\mathcal{K} \models B(b)$ and where $A_0 \not\sqsubseteq A$ summarizes information about a not contained in A.

Proposition 1 *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{SHI}$ ontology containing named concept A, concepts A_0 and B, and role R. If equation (8) holds, there must exist some GCIs in $\mathcal{T}$ of the form*

$$\exists R'.C_1 \bowtie C_2 \sqsubseteq C_3 \quad (9)$$

where $R \sqsubseteq R'$, $\bowtie$ is a placeholder for either $\sqcap$ or $\sqcup$, and C_i's are concepts.

Proof of this proposition can be found in (Xu et al., 2013).

Proposition 1 directly leads to a first syntactic condition denoted SYN_COND.

Definition 4 (SYN_COND) *Role assertions of the form $R(a, b)$ are said to be **true** for* SYN_COND *if role R participates in at least one axiom that can be logically converted to the form of equation 9 for some $R \sqsubseteq R'$, **false** otherwise.*

Assertions $R(a, b)$ with SYN_COND = **false** do not affect the classification of a unless either R or some R' such that $R \subseteq R'$ exist in the query, and can be safely removed from the calculation of $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$. By symmetry to the inverse role R^{-}, this condition also applies to b. In practice, the axioms more likely to be found are of the form $C \sqsubseteq \exists R.D$, which is equivalent to $C \sqcap \top \sqsubseteq \exists R.D$. Also note that $(C \sqsubseteq \forall R.D) \equiv (\exists R.\neg C \sqsubseteq \neg D)$. In the example in Table 1, assertions with

role `isHeadOf` have SYN_COND=**true**, while assertions with roles `takesCourse` and `taughtBy` have SYN_COND=**false** and can be safely removed from consideration, unless the roles exist in the query itself. Note that in this case, the MSC for `harry` is reduced to

$$\texttt{Student} \sqcap \exists \texttt{takesCourse}^{-}.\texttt{Course} \tag{10}$$

A second syntactic condition presented in (Xu et al., 2015a) relies on the identification of explicit concept assertions and disjointness axioms that indicate that an individual cannot be classified to an existential restriction:

Definition 5 (SYN_COND_DJ) *Role assertions $R(a, b)$ with* SYN_COND= ***true*** *are said to be* ***true*** *for* SYN_COND_DJ *if there do not exist any of:*

- *an explicit concept assertion $B_0(b)$ such that $\mathcal{K} \models B_0 \sqsubseteq \neg C_1$;*
- *an explicit concept assertion $A_0(a)$ such that $\mathcal{K} \models A_0 \sqsubseteq \neg C_3$; or*
- *an explicit concept assertion $A_0(a)$ such that $\mathcal{K} \models A_0 \sqsubseteq \neg(C_3 \sqcup \neg C_2)$*

for C_1, C_2, an C_3 as in Eq.(9) and $R \sqsubseteq R'$.

Role assertions with SYN_COND_DJ=**false** can be safely removed from the calculation of the MSC of any individual, since they do not affect classification of either a or b. For example, in Table 1, role assertions with role `takesCourse` have SYN_COND=**true** due to the assertion

$$\texttt{takesCourse.MagicCourse} \sqsubseteq \texttt{Wizard}.$$

However, suppose that the following assertion were inserted into the ABox:

$$\texttt{takesCourse(dudley,math)} \tag{11}$$

This assertion has SYN_COND_DJ=**false**, since `dudley` is an instance of `Muggle`, which in turn is disjoint with `Wizard`, the filler concept in the assertion above.

Definition 6 (SYN_COND_SC) *Role assertions $R(a, b)$ with* SYN_COND= ***true*** *are said to be* ***true*** *for* SYN_COND_SC *if, for every GCI of the form in Eq.(9) there exists an explicit concept assertion $A_0(a)$ such that $\mathcal{K} \models A_0 \sqsubseteq C_3$.*

Role assertions with SYN_COND_SC=**true** are redundant for the classification of a, and can therefore be safely removed from the calculation of the MSC of any individual. For example, the role assertion

$$\texttt{Headmaster(albus)} \tag{12}$$

would have SYN_COND_SC=**true**.

Application of these three syntactic conditions to the computation of the MSC of a given individual results in a significant reduction in the size of the MSC for practical ontologies. The reasons for this reduction will be established in more detail in the next section, but first we provide a definition for the parallelization of the algorithm.

Remark 1 *The MSC method is guaranteed to be sound and complete for $\mathcal{SHI}$ DL (Xu et al., 2015a). It can be extended to DL with expressivity $\mathcal{SHIQ}$ with unique name assumption, that is, if it is assumed that two individuals a and b are different if they have different names. Details of such extension are outside the scope of this paper, but they are straightforward, and generally follow the extensions to equality-free module extraction detailed in (Xu et al., 2013).*

2.4 Parallelization of the MSC Method

The MSC method, including the syntactic conditions detailed above, is highly parallelizable, due to the following:

Proposition 2 *For a given knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, computation of $\mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$ for an individual a can be performed independently of the computation of the MSC of any other individual.*

Proof of this proposition follows directly from the definition of the MSC computation in Definitions 2 and 3, as well as in the definition of the syntactic conditions, where it should be clear that MSC computation depends only on the initial state of the knowledge base. This means that the calculation of the MSCs for all individuals can be done in parallel. Instance checking then needs to be performed against $\mathcal{T} \cup \mathrm{MSC}_{\mathcal{T}}(\mathcal{A}, a)$ for every individual a.

Note that naive parallelized processing of individual equality assertions (i.e., `owl:sameAs` expressions) using the procedures outlined in Definition 2 results in redundant computations. Precomputation of the reflexive-transitive closure of these equalities avoids such redundancy. Note also that anonymous individuals, typically presented as blank nodes in OWL, are processed in the same way as named individuals for calculation purposes.

As is shown later in Section 4, parallelization of the MSC method allows for the processing of extremely large ABoxes within clusters of commodity hardware. The resulting computational complexity is sublinear with respect to the size of the ABox, which indicates linear complexity for single-machine implementation. In the next section, we present an analysis of the expected computational complexity of the enhanced MSC method, and show why it has tractable expected performance for practical ontologies.

3 Data Complexity and Random Graphs

3.1 MSC Method Worst-Case Complexity

Typically, the size of the ABox as measured by the number of facts it contains is orders of magnitude greater than the size of the TBox, and also much larger than the size of the query. While it has been shown that, even with exponential time complexity, TBoxes of realistic size can be processed in realistic time (Donini, 2003), practical ABoxes can contain million of individuals. Data complexity for instance checking in DLs with expressivity $\mathcal{SHIQ}$ has an upper bound of 2EXPTIME (Glimm et al., 2007). A family of less expressive languages called *DL-Lite* have been shown to be first-order-logic rewritable and thus with data complexity in AC^0, while even small additions to this language family renders the data complexity for instance checking in co-NP complete or worse (Calvanese et al., 2013).

It is clear that the MSC method for instance checking is EXPTIME-complete for $\mathcal{SHI}$. Since the computation of the MSC is PTIME, as it is essentially depth-first search, complexity of checking depends on TBox reasoning. From Definition 1, it should be clear that for a connected ABox, the size of the MSC is the size of the ABox itself. Since testing for the subsumption of the MSC against a concept in the TBox is reducible to satisfiability testing of the union of the TBox and the MSC, instance checking with the MSC method is equivalent to satisfiability testing for $\mathcal{SHI}$ DLs, which is EXPTIME-complete (Tobies, 2001). Given that the syntactic conditions defined in subsection 2.3 do not guarantee that any role assertions will be actually removed from the calculation of the MSC, the worst-case complexity of the enhanced MSC method is still EXPTIME. This situation is unlikely to occur in practice however, as it would mean that every role in the ABox would have to be involved in a GCI of the form of equation (9).

From a practical standpoint, it seems more interesting to study the performance of the enhanced MSC method when used with *typical* ABoxes. Graph theory, and particularly random graph theory, provide methods and tools for the study of typical graphs.

3.2 Random Graph Theory

Random graph theory is concerned with the study of graphs as probabilistic random variables with a defined probability distribution. Here we provide a short summary of the points of interest to this paper; the reader is referred to (Chung, 2009) for more details.

Many real-world network structures are so-called *scale-free*, that is, they exhibit power-law degree distributions (Mika, 2007; Newman, 2002). A random power-law graph model can be defined as $\mathcal{G}(C, \alpha)$, where the number of nodes n_k with degree k is given by

$$n_k = C \cdot k^{-\alpha} \tag{13}$$

where C is the volume of the graph and α is the log-log rate of growth of the graph.

It is important to note that random power-law graphs do not contain nodes with degree of 0, since at this value the distribution is undefined; therefore, analysis of random power-law graphs discards all zero-degree nodes. Where it is necessary to make the distinction, we will denote the number of non-zero-degree nodes the *effective order* of the graph.

It has been shown that a *phase transition* occurs as the average node degree increases (where the *degree* of a node is the number of edges connected to it.), so that at a critical value a *giant component* forms *with high probability* [1] (Erdős and Rényi, 1960; Molloy and Reed, 1995). A *component* (also called *connected component*) is a maximal subset of the nodes of the graph and all edges between those nodes such that there is a path between any two nodes in the component. A giant component is a connected component with number of nodes in the same order of magnitude as the graph itself. For any random graph, the phase transition occurs at the point given by the solution of:

[1] Since random graph theory is probabilistic, conclusions are always obtained "with high probability". Henceforth, we will abbreviate this as w.h.p. This is also sometimes stated as "asymptotically almost surely", or just "almost surely".

$$\sum_{k=0}^{\infty} k(k-2)p_k = 0 \tag{14}$$

where p_k is the degree distribution. For power-law graphs, this reduces to (Newman, 2002; Chung, 2009):

$$\sum_{k=0}^{\infty} k^{1-\alpha} k^{2-\alpha} = \zeta(\alpha-2) - 2\zeta(\alpha-1) = 0 \tag{15}$$

after integration, where $\zeta(t) = \sum_{n=1}^{\infty} n^{-t}$ is the Riemann zeta function. Thus, the phase transition point occurs at the value of the power-law exponent $\alpha_0 = 3.47875....$ Graphs with exponent $\alpha < \alpha_0$ show a unique giant component of size $O(n)$ w.h.p. It can also be shown that for $\alpha > 2$ the second largest component is in size $\theta(\log n)$, for $1 < \alpha < 2$ the second largest component is in $\theta(1)$, and for $\alpha < 1$ the graph is connected, all w.h.p (Chung, 2009).

3.3 Random Graphs and the MSC method

To apply random graph theory to the study of DL ABoxes, we define the role assertion *graph* as follows:

Definition 7 (ABox Role Assertion Graph) *The set of role assertions in a $\mathcal{SHI}$ ABox define an unlabeled, undirected graph $\mathcal{G}_\mathcal{A}$, where the nodes represent the individuals in the ABox, and where there is an edge between nodes if there is at least one role assertion between the underlying individuals.*

In other words, the ABox graph $\mathcal{G}_\mathcal{A}$ replaces multiple parallel edges between two nodes with a single, unlabeled edge. Self-loops are still allowed. Note that with respect to the MSC method, parallel edges result in only a small increase in MSC size, equivalent to the number of parallel edges. Suppose multiple role assertions $R_1(a,b), R_2(a,b), \ldots, R_n(a,b)$ exist in between individuals a and b in the ABox $\mathcal{A}$; application of the rollup procedure for assertion cycles in Definition 3 results in the following:

$$\begin{aligned}\mathrm{MSC}_\mathcal{T}(\mathcal{A},a) \leftarrow \mathrm{MSC}_\mathcal{T}(\mathcal{A},a) \sqcap \{a\} \\ \sqcap \exists R_1.(\exists R_2^-\{a\} \sqcap \cdots \sqcap \exists R_n^-\{a\} \\ \sqcap \mathrm{MSC}_\mathcal{T}(\mathcal{A}\backslash\{R_1(a,b), R_2(a,b) \\ \ldots R_n(a,b)\}, b))\end{aligned} \tag{16}$$

Thus, collapsing parallel edges into a single unlabeled edge results in minimal reduction in the complexity of MSC calculations.

Certain properties in the role assertion graph $\mathcal{G}_\mathcal{A}$ can be readily related to expected characteristics of the MSCs calculated for the underlying ABox. Specifically:

- the *size* of $\mathrm{MSC}_\mathcal{T}(\mathcal{A},a)$, as measured by number of simple form concepts that it contains, is equivalent to the *order* of the connected component that contains the node that represents the individual a.
- the *number of conjuncts* at the top level of $\mathrm{MSC}_\mathcal{T}(\mathcal{A},a)$ is proportional to the degree of the node representing a.
- the maximum *quantification depth* of $\mathrm{MSC}_\mathcal{T}(\mathcal{A},a)$ is given by the diameter of the connected component containing the node corresponding to a.

It is clear that the appearance of a giant component then means that the MSC for a significant number of individuals is in $O(n)$, and that therefore the complexity of MSC instance checking is exponential in n.

The application of the syntactic conditions outlined in subsection 2.3 can be reflected in the role assertion graph through the removal of assertions that do not satisfy the conditions. An important question to ask is how such removal affects the characteristics of the resulting graph. Since random graphs are generative processes, it is reasonable to assume that removal of edges follows the same degree distribution as the original generation of the graphs themselves. If this is the case, the resulting graphs retain their power-law degree distribution. Thus, if the original ABox role assertion graph had a giant component of size $O(n)$, for n the number of nodes, w.h.p. the modified graph will also show a giant component.

Note however that the reduction in the number of edges necessarily results also in a reduction in the effective order of the graph, that is, in the number of non-zero-degree nodes. It is in fact possible to calculate both the number of nodes n and number of edges E in the graph based on the volume and log growth rate parameters from equation 13. For $\alpha > 2$, the equations are as follows (Chung, 2009):

$$n = \sum_{k=1}^{\infty} C \cdot k^{-\alpha} \approx C \cdot \zeta(\alpha) \tag{17}$$

	Classes		Existential Restrictions	
	MSC	**HermiT**	**MSC**	**HermiT**
Min.	7.82	0.67	7.30	0.76
Max.	19.66	780.20	26.94	2,848.68
Avg.	8.38	19.42	9.14	489.95
Std. Dev.	1.42	90.50	3.24	698.14
Median	8.16	0.79	8.19	135.94

Table 2: Times (in seconds) for execution of class and existential restriction queries

$$E = \frac{1}{2}\sum_{k=1}^{\infty} k \cdot C \cdot k^{-\alpha} \approx \frac{1}{2} \cdot C \cdot \zeta(\alpha - 1) \quad (18)$$

The ratio $\frac{n}{E}$ is then

$$\frac{n}{E} \approx \frac{2\zeta(\alpha)}{\zeta(\alpha - 1)} \quad (19)$$

which depends only on α. Therefore, if α remains constant, **a reduction in the number of edges necessarily results in a proportional reduction in the number of (non-zero-degree) nodes**, that is, in the effective order of the graph. As will be shown in Section 4, it is this property that enables the enhanced MSC method to provide extremely good performance for very large ABoxes.

4 Experimental Evaluation

4.1 Accuracy

To test both the accuracy and to measure parallelization speedup of the enhanced MSC method, we set up clusters of compute-optimized instances through Amazon Web Services (AWS)[2]. The enhanced MSC method with syntactic condition correction was implemented in Java and Scala to work over Apache Spark[3], installed over Hadoop HDFS and YARN.

For the accuracy tests, we used an in-memory version of our enhanced MSC implementation. We set up clusters of two c4.xlarge machines, each containing 4 virtual CPUs and 7.5GB of memory, to run the enhanced MSC method, and compared the results against the HermiT reasoner version 3.8.1[4], running on a single c4.xlarge machine; HermiT was chosen as a comparison standard due to its stability and speed; future tests will be done against other reasoners such as Pellet[5] and Konclude[6]. These tests were performed against a single department dataset for the University Ontology Benchmark (UOBM) (Ma et al., 2006), containing about 150,000 triples. The UOBM TBox was modified to convert cardinality restrictions to existential restrictions, since our current implementation only handles expressivity up to $\mathcal{SHI}$. This modified version can be provided upon request. The UOBM Tbox contains a total of 113 named classes and 35 object properties. We tested our enhanced MSC implementation against all 35 named class queries, and all 3,955 possible single-depth existential restriction queries, and verified 100% agreement between our enhanced MSC method implementation and HermiT. In addition, we recorded the running time for all query executions - the results can be seen in Table 2. It is interesting to note that the enhanced MSC method performs much better than HermiT for existential restrictions. Also note the large standard deviation found when running a hypertableaux-based reasoner like HermiT, where a few queries take a very long time to finish. This result also suggests the possibility of using enhanced MSC in tandem with a traditional reasoner when dealing with smaller datasets.

4.2 Parallelization

To test the parallelization of the MSC method, we used the c3.8xlarge instances,which provide 32 virtual CPUs and 60 GBs of storage. We used the Lehigh University Benchmark (LUBM) (Guo et al., 2005) to generate data sets of up to 500 million triples. LUBM was chosen as an initial test ontology due to its ability to generate datasets of varying size, while providing a reasonably expressive TBox. These data sets were stored using the TitanDB[7] graph database interface over an Apache HBase[8] backend. Both TitanDB and HBase were installed over Hadoop HDFS 2.7. Our prototype application over Spark accesses TitanDB through its standard Java interface. Data distribution and replication are performed by the database and are transparent to our application.

A test was performed to evaluate the scalability of the parallel MSC method over the number of triples in the ABox. This test was performed over

[2] aws.amazon.com
[3] https://spark.apache.org/
[4] http://www.hermit-reasoner.com
[5] https://github.com/stardog-union/pellet
[6] http://derivo.de/produkte/konclude/
[7] http://thinkaurelius.github.io/titan/
[8] http://hbase.apache.org/

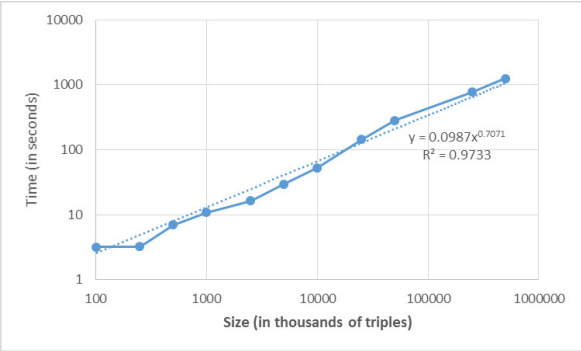
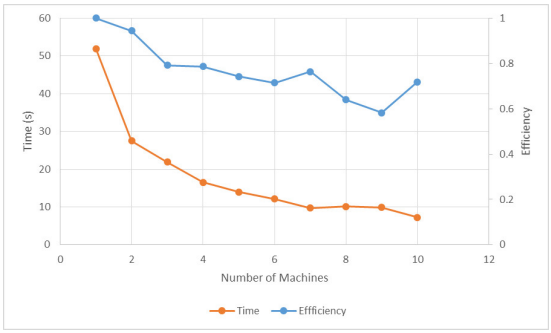


Figure 1: Execution time vs. data set size with LUBM datasets, in AWS, for 10 c3.8xlarge machines with 32 cores each.

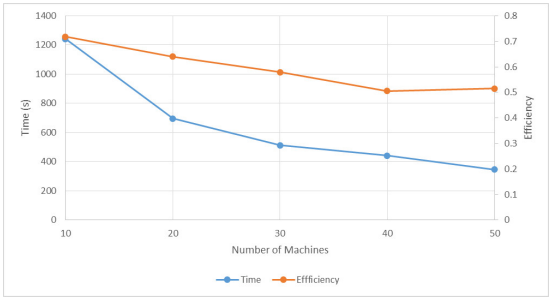
a cluster of 10 c3.8xlarge machines in AWS. The results are shown in the log-log diagram in Figure 1. As can be observed, the method shows clear sub-linear performance with respect to the size of the data set, as expected from an algorithm with linear performance in a sequential machine.

The performance with respect to the number of machines was evaluated in two parts. First, to evaluate under small cluster conditions, a 500,000 triple set was assembled and used to test against 1 to 10 machines. The execution time and the efficiency with respect to single-machine execution are shown in Figure 2a. To obtain evaluation for large numbers of machines, we used the ABox with 500 million triples and ran it against 10 to 50 machines; to provide a more realistic estimation, the efficiency value was corrected against the result for 500,000 triples in 10 machines. These higher scalability results are shown in Figure 2b.

In terms of raw performance, the parallel enhanced MSC algorithm was capable of performing instance checking for a dataset with 500 million triples and over 110 million individual instances in about 1,240 seconds, or around 20 min., using a cluster of 10 machines and a total of 320 execution cores. Using 50 machines, the execution time was 346 seconds, or somewhat less than 6 minutes. As a comparison, although the difference in algorithms means that execution times are not directly comparable, Oracle reports full ABox inference over 869 million triples in 62 minutes, and query performance over this pre-reasoned ABox in about 4.3 min., using specialized hardware (W3C, 2015). It is also important to note that, since tests were performed over an uncontrolled environment, external perturbations could have affected



(a) Efficiency for small number of machines



(b) Efficiency for large number of machines

Figure 2: Execution time and efficiency of parallelization.

some measurements. Nevertheless, it is clear that the MSC method provides performance comparable with top-of-the-line database technologies and very broad scalability.

These results demonstrate that the enhanced MSC method is inherently parallelizable, enabling it to work with large scale ABoxes. They also show that the fact that in the worst case the enhanced MSC method is in EXPTIME does not preclude its usefulness with practical ontologies. In the next section, we evaluate the characteristics of typical ABoxes that support this assertion using random graph models.

4.3 Random Graph Complexity Analysis

For this purpose, we have created ABox role assertion graphs for the same set of ontologies used for the empirical evaluation in (Xu et al., 2015a), and have calculated the least-squares fit for their degree distributions. The data sets used are the following:

1. LUBM1 and LUBM4: Lehigh University Benchmark ontologies about higher education entities generated using the tool provided by (Guo et al., 2005); and
2. Arabidopsis thaliana (AT) and Caenorhabditis elegans (CE), two ABoxes built over the BioPAX TBox about biomedical pathways in

	n	E	C	α	R^2
LUBM1	17,174	49,321	19,608	2.051	0.75
LUBM4	78,579	236,514	162,116	2.276	0.72
AT	42,695	74,680	75,558	2.653	0.92
CE	37,162	68,034	147,066	2.89	0.92

Table 3: Degree distribution of role assertion graphs of common ontologies: The least-squares fit approximation for the degree distributions of the role assertion graphs of selected common ontologies is shown. n is the number of nodes; E the number of edges; C and α are the volume and log-growth of the power law; and R^2 is the correlation coefficient of the approximation.

the respective organisms; the TBox and both ABoxes can be provided upon request.

We have performed the least-squares fit up to a maximum degree of 100, which accounts for over 99.8% of all nodes in every graph considered. Since there usually are discrepancies when the degree is very small or very large (Chung, 2009), this truncation eliminates a significant source of inaccuracy in the models. Table 3 provides the total number of edges and nodes of the role assertion graph (remember that the number of edges differs from the total number of properties in the ABoxes due to the conversion to a graph), values for C and α in Equation 13, and correlation coefficient R^2. Note that the larger graphs approximate a power-law degree distribution to a very high degree. Note also that the exponents of the approximations are all between 2 and 3, which is the typical range for small-world, scale-free networks (Mika, 2007).

We have then calculated the effective order and log-growth rate of the degree distribution of the ABox role assertion graphs for all ontologies after application of SYN_COND, SYN_COND_DJ, and SYN_COND_SC; the results, shown in Table 4, demonstrate that these conditions indeed produce a drastic reduction in the effective order of the role assertion graph and thus in the size of the resulting connected components, even if a giant component still appears. Observe in particular that for the LUBM ontologies, the reduction is so significant that only a very small number of individuals remain connected to others.

It is interesting to examine the relative impact of the various syntactic conditions. Note that SYN_COND must always be applied, since without it, the other conditions do not make sense. Table 5 shows the impact for the application of

	Eff. n	**E**
LUBM1	30	15
LUBM4	142	71
AT	11,364	11,573
CE	9,123	8,240

	#	**Max. Size**	**Avg. Size**	**non-sngltn**
LUBM1	17,160	2	1.00	15
LUBM4	78,508	2	1.00	71
AT	32,911	442	1.30	1,580
CE	29,913	251	1.24	1,874

Table 4: Component sizes after application of syntactic conditions. The 'non-sngltn" column gives the number of components that have more than one node.

the different conditions on the AT ABox; similar trends can be seen with the other data sources. As can be seen, the initial SYN_COND condition accounts for a significant portion of the reduction, but still leaves highly connected components, and is therefore not sufficient on its own to enable processing of instance checking in realistic time. Both the disjointness and subclass conditions contribute to a further reduction, resulting finally in a realistic component size for TBox reasoning purposes. In general, SYN_COND_SC contributes more of the reduction in size, for all studied ontologies. It should be also noted that a very large number of resulting components are singletons, which in the MSC method result in subsumption checking based only on class assertions; this can be performed very fast since the class hierarchy of the TBox can be precomputed.

Up to now, we have assumed that the power law exponent remains constant after application of the various syntactic conditions. It is interesting then to evaluate the behavior of the exponent in the power law distribution of equation 13. Figure 3 shows this for the AT data source on a log-log scale; as can be seen, other than significant deviations at very low degree values, the exponent remains relatively constant.

5 Discussion and Future Work

The enhanced MSC method is inherently parallelizable, since instance checking for every individual in the ABox can be performed independently. Coupled with recent advances in cluster computing such as Apache Spark, large triple

	Graph size		Components			
	Effective n	E	#	Max. Size	Avg. Size	non-sngltn
SYN_COND	33,235	33,015	12,447	2,411	3.43	2,987
SYN_COND_DJ	23,377	22,374	22,161	862	1.93	2,843
SYN_COND_SC	19,197	19,099	25,801	1,695	1.66	2,303
SYN_COND_ALL	11,364	11,573	32,911	442	1.30	1,580

Table 5: Component sizes after application of different syntactic conditions on the AT data set. SYN_COND_ALL refers to the application of all conditions.

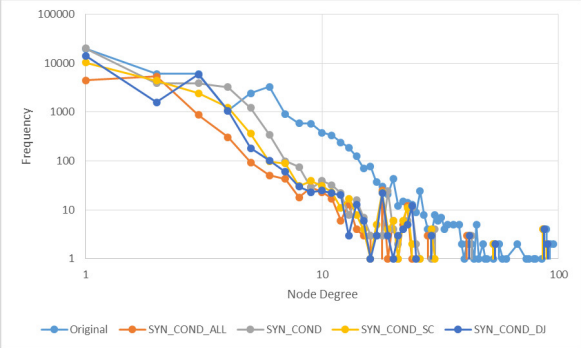


Figure 3: Power law distribution of AT data source after application of different syntactic conditions.

stores can be queried efficiently using commodity hardware clusters or cloud platforms.

Even as worst-case complexity of the method is in exponential time for a single sequential machine, and thus intractable for parallelization, the method performs within realistic time in practical, real-world ontologies, which typically present a power-law degree distribution. As discussed in our random graph theory analysis, this is due to the ability of the method to reduce the size of the connected components formed within the ABox Role Assertion Graph, which in turn means a reduction in the size of the MSCs generated for the individuals in the ABox. Our method is thus most effective when the different syntactic conditions can be applied to effect this reduction, and will possibly prove less beneficial if most of the roles in the TBox are engaged in axioms of the form of equation (9).

We are currently exploring the use of the method to address full conjunctive queries over DL knowledge bases expressed in the SPARQL query language. This requires the expansion of the MSC method to retrieve individuals a and b that can be inferred to be in a relation $R(a, b)$. This extension is being implemented based on Theorem 4.1 in (Xu et al., 2015a). The implementation of SPARQL query answering will also enable us to perform tests with other existing benchmarks such as the DBPedia SPARQL Benchmark (DBPSB). In addition, we are working on improvements in the efficiency of the parallelization of the MSC method. In particular, we are looking into combining multiple individuals that form part of the same connected component in the ABox role assertion graph in the same parallel task, since it can be seen in Definitions 2 and 3 that portions of the MSC computation can be shared among individuals provided that they are connected to each other.

6 Conclusion

In this paper, we have presented a parallel implementation of the enhanced MSC method, and we have evaluated execution time and efficiency as we varied the size of the data and the number of processors used. Since the method performs independent checking for every individual in the ABox, it is inherently parallelizable. The results show sublinear performance with respect to the size of the ABox, which stems from its performance in linear time in the computation of the MSCs, and almost constant time for reasoning due to the small size of the resulting MSC.

Acknowledgements

This work is supported by grant # R44GM097851 from the National Institute of General Medical Sciences (NIGMS), part of the U.S. National Institutes of Health (NIH).

References

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. 2003. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA.

Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.

2013. Data complexity of query answering in description logics. *Artificial Intelligence* 195:335–360. https://doi.org/10.1016/j.artint.2012.10.003.

Fan Chung. 2009. A whirlwind tour of random graphs. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, Springer, pages 7493–7505.

F Donini and A Era. 1992. Most specific concepts for knowledge bases with incomplete information. In *Proceedings of CIKM*. Baltimore, MD, USA, pages 545–551.

Francesco M. Donini. 2003. Complexity of Reasoning. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The description logic handbook: theory, implementation, and applications*, Cambridge University Press, New York, NY, USA, pages 96–136.

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. 1994. Deduction in Concept Languages: from Subsumption to Instance Checking. *J Logic Computation* 4(4):423–452. https://doi.org/10.1093/logcom/4.4.423.

P. Erdős and A. Rényi. 1960. On the Evolution of Random Graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*. pages 17–61.

Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler. 2007. Conjunctive Query Answering for the Description Logic SHIQ. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'07, pages 399–404.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A Benchmark for OWL Knowledge Base Systems. *Web Semant.* 3(2-3):158–182.

Ian Horrocks. 2008. Ontologies and the Semantic Web. *Commun. ACM* 51(12):58–67. https://doi.org/10.1145/1409360.1409377.

Ian Horrocks and Sergio Tessaris. 2000. A conjunctive query language for description logic ABoxes. In *In In AAAI/IAAI*. pages 399–404.

Li Ma, Yang Yang, Zhaoming Qiu, Guotong Xie, Yue Pan, and Shengping Liu. 2006. Towards a Complete OWL Ontology Benchmark. In *The Semantic Web: Research and Applications*. Springer, Berlin, Heidelberg, pages 125–139.

Peter Mika. 2007. *Social Networks and the Semantic Web*, volume 5 of *Semantic Web and Beyond*. Springer US, Boston, MA.

Michael Molloy and Bruce Reed. 1995. A critical point for random graphs with a given degree sequence. *Random Struct. Alg.* 6(2-3):161–180.

Boris Motik, Rob Shearer, and Ian Horrocks. 2007. Optimized Reasoning in Description Logics Using Hypertableaux. In Frank Pfenning, editor, *Automated Deduction – CADE-21*, Springer Berlin Heidelberg, number 4603 in Lecture Notes in Computer Science, pages 67–83.

Ralf Möller, Volker Haarslev, and Michael Wessel. 2007. On the Scalability of Description Logic Instance Retrieval. In Christian Freksa, Michael Kohlhase, and Kerstin Schill, editors, *KI 2006: Advances in Artificial Intelligence*, Springer Berlin Heidelberg, number 4314 in Lecture Notes in Computer Science, pages 188–201.

Bernhard Nebel. 1990. Reasoning and Revision in Hybrid Representation Systems. In *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Mark E. J. Newman. 2002. Random graphs as models of networks. In Stefan Bornholdt and Hans Georg Schuster, editors, *Handbook of Graphs and Networks*, Wiley-VCH Verlag GmbH & Co. KGaA, pages 35–68.

Sambhawa Priya, Yuanbo Guo, Michael Spear, and Jeff Heflin. 2014. Partitioning OWL Knowledge Bases for Parallel Reasoning. IEEE, pages 108–115. https://doi.org/10.1109/ICSC.2014.34.

Andrea Schaerf. 1994. Reasoning with individuals in concept languages. *Data & Knowledge Engineering* 13(2):141–176.

Stephan Tobies. 2001. Complexity Results and Practical Algorithms for Logics in Knowledge Representation. *arXiv:cs/0106031* ArXiv: cs/0106031.

W3C. 2015. Large Triple Stores - W3c Wiki. https://www.w3.org/wiki/LargeTripleStores.

Sebastian Wandelt and Ralf Möller. 2012. Towards ABox Modularization of Semi-expressive Description Logics. *Appl. Ontol.* 7(2):133–167.

Jia Xu, Patrick Shironoshita, Ubbo Visser, Nigel John, and Mansur Kabuka. 2013. Extract ABox Modules for Efficient Ontology Querying. *arXiv:1305.4859 [cs]* ArXiv: 1305.4859.

Jia Xu, Patrick Shironoshita, Ubbo Visser, Nigel John, and Mansur Kabuka. 2015a. Converting Instance Checking to Subsumption: A Rethink for Object Queries over Practical Ontologies. *International Journal of Intelligence Science* 05(01):44–62. ArXiv: 1412.7585. https://doi.org/10.4236/ijis.2015.51005.

Jia Xu, Patrick Shironoshita, Ubbo Visser, Nigel John, and Mansur Kabuka. 2015b. Module Extraction for Efficient Object Queries over Ontologies with Large ABoxes. *AIA* 2(1):8–31. https://doi.org/10.15764/AIA.2015.01002.