Reasoning with Large Numbers of Individuals Moves on: Extending the Instance Store

Lei Li

Information Management Group
Department of Computer Science
University of Manchester
Manchester, UK
lil@cs.man.ac.uk

1 Introduction

The Semantic Web [2] aims at enabling Web *resources* to be accessible to automated processes by adding "semantic annotations"—metadata (data about data) that describes their content. It is envisaged that the semantics in semantic annotations will be given by *ontologies* [8], which will provide a source of precisely defined terms (vocabularies) that are amenable to automated reasoning.

A standard for expressing ontologies in the Semantic Web has already emerged: the ontology language *OWL* [4], which recently became a W3C proposed recommendation. One of the main features of OWL is that there is a direct correspondence between (two of the three "species" of) OWL and Description Logics (DLs) [12]. This means that reasoning procedures from DLs can be used in order to infer useful facts from ontology based annotations in OWL. In particular, since annotations are usually defined as individual instances of ontology classes, it is reasonable then to use *ABox reasoning* (the DL term for reasoning about individuals) to reason about annotations [3]. However, developing a practical ABox reasoner for such annotations is difficult. This difficulty arises not only from the computational complexity of ABox reasoning, but also from the fact that the number of annotations might be extremely large.

On the one hand, the task of reasoning with large amounts of individuals is difficult even for the state-of-art optimised ABox reasoner RACER [9]. On the other hand, efficient management of very large volumes of data is well known to be a stronghold of relational databases. The approach of combining relational databases with Description Logics to deal with large amounts of data was, therefore, proposed in [11] which we call an *Instance Store*.

The Instance Store provides an infrastructure for reasoning with *large numbers of individuals*. It combines a DL *reasoner* with a *database* in order to provide large scale storage of instance data, and sound and complete answers to instance retrieval queries for arbitrary query concepts. However, in order to simplify the integrated architecture, and to provide acceptable query answering performance, the current Instance Store only supports ABox axioms asserting that a given individual is an instance of a given (possibly complex) concept (e.g., John is an instance of the class of persons having at least one female child); but it forbids axiom assertions involving role relationships between pairs of individuals (e.g., an assertion that Mary is a child of John).

Certainly, the idea of the Instance Store is not new, but, to the best of our knowledge, it is the first implementation of a general purposed *role-free*[1] DL reasoner over individuals that is sound and complete, has reasonable response times. This claim is supported by presenting a variety of empirical test results [11] contrasting the Instance Store with the other existing techniques such as RACER.

[1] An ABox is said to be **role-free** if it does not contain role assertions, i.e., all the assertions are of the form $a : C$ rather then the form $\langle a, b \rangle : R$.

2 A Brief Introduction to Description Logics

The family of Description Logics [1] is a knowledge representation system evolved from early *frame systems* [14] and *semantic networks* [15]. DLs are distinguished from their ancestors by having a precise *semantics* which enables the description and justification of automated deduction processes.

A DL knowledge base typically consists of two components—"TBox" and "ABox". The TBox defines the structure of the knowledge domain and consists of a set of asserted axioms, i.e., the definition of a new concept in terms of other previously defined concepts. ABox contains a concrete knowledge domain and asserted axioms about individuals, e.g., an individual is an instance of a concept or an individual is related to another by a role.

The basic inference on concept expressions in a TBox is *subsumption* checking, typically written $C \sqsubseteq D$. Determining *subsumption* is the problem of checking whether the concept denoted by *D* (the subsumer) is more general than the one denoted by *C* (the subsumee). In other words, *subsumption* checks whether the first concept always denotes a subset of the set denoted by the second one. The basic task of constructing a terminology in a TBox is *classification*, which amounts to placing a new concept in the suitable place in a taxonomic hierarchy according to the partial order induced by *subsumption* relationships among the other defined concepts.

The basic reasoning task in an ABox is *instance checking*, which verifies whether a given individual is an instance of (belongs to) a specified concept, written as $(a\colon C)$. Formally, an interpretation $\mathcal{I}$ maps [2] each individual name like *a* to an element of $\Delta^{\mathcal{I}}$, it satisfies $(a\colon C)$ if and only if $a^{\mathcal{I}} \in C^{\mathcal{I}}$. There are other ABox reasoning tasks of practical usage, such as *retrieval*— given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a concept C. It is the problem of finding all individuals a such that $\mathcal{K} \models a\colon C$. It is clear that that tasks such as *retrieval* can be reduced to *instance checking* [6].

3 Instance Store

An ABox $\mathcal{A}$ is role-free if it contains only axioms of the form $x : C$. We can assume, without loss of generality, that there is exactly one such axiom for each individual as $x : C \sqcup \neg C$ holds in all interpretations, and two axioms $x : C$ and $x : D$ are equivalent to a single axiom $x : (C \sqcap D)$. It is well known that, for a role-free ABox, instantiation can be reduced to TBox subsumption [10, 16]; i.e., if $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and $\mathcal{A}$ is role-free, then $\mathcal{K} \models x : D$ iff $x : C \in \mathcal{A}$ and $\mathcal{T} \models C \sqsubseteq D$. Similarly, if $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and $\mathcal{A}$ is a role-free ABox, then the instances of a concept D could be retrieved simply by testing for each individual x in $\mathcal{A}$ if $\mathcal{K} \models x : D$. However, this would clearly be very inefficient if $\mathcal{A}$ contained a large number of individuals.

An alternative approach is to add a new axiom $C_x \sqsubseteq D$ to $\mathcal{T}$ for each axiom $x : D$ in $\mathcal{A}$, where C_x is a new atomic concept; we will call such concepts *pseudo-individuals*. Classifying the resulting TBox is equivalent to performing a complete realisation of the ABox: the most specific atomic concepts that an individual x is an instance of are the most specific atomic concepts that subsume C_x and that are not themselves pseudo-individuals. Moreover, the instances of a concept D can be retrieved by computing the set of pseudo-individuals that are subsumed by D.

The problem with this latter approach is that the number of pseudo-individuals added to the TBox is equal to the number of individuals in the ABox, and if this number is very large, then TBox reasoning may become inefficient or even break down completely (e.g., due to resource limits).

The basic idea behind the Instance Store is to overcome this problem by using a DL reasoner to classify the TBox and a database to store the ABox, with the database also being used to store a complete realisation of the ABox, i.e., for each individual x, the concepts that x realises (the most specific atomic concepts that x instantiates). The realisation of each individual is computed using the DL (TBox) reasoner when an axiom of the form $x : C$ is added to the Instance Store ABox.

A retrieval query Q to the Instance Store (i.e., computing the set of individuals that instantiate a concept Q) can be answered using a combination of database queries and TBox reasoning. Given an

[2] Note that, although not mandatory, this mapping usually has to respect the *unique name assumption* (UNA), i.e., distinct individual names denote distinct objects. Formally, if *a* and *b* are distinct individual names, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Instance Store containing a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ and a query concept Q, retrieval involves the computation of the following sets of individuals for which we introduce a special notation:

- I_1 denotes the set of individuals in $\mathcal{A}$ that realise *some* concept in $Q\downarrow_{\mathcal{T}}$;
- I_2 denotes the set of individuals in $\mathcal{A}$ that realise *every* concept in $\lceil Q \rceil_{\mathcal{T}}$.

The Instance Store algorithm to retrieve the instances of Q can be then described as follows:

1. use the DL reasoner to compute $Q\downarrow_{\mathcal{T}}$;
2. use the database to find the set of individuals I_1;
3. use the reasoner to check whether Q is equivalent to any atomic concept in $\mathcal{T}$; if that is the case then simply return I_1 and *terminate*;
4. otherwise, use the reasoner to compute $\lceil Q \rceil_{\mathcal{T}}$;
5. use the database to compute I_2;
6. use the reasoner and the database to compute I_3, the set of individuals $x \in I_2$ such that $x : C$ is an axiom in $\mathcal{A}$ and C is subsumed by Q;
7. return $I_1 \cup I_3$ and *terminate*.

4 Extending the Instance Store

Although the current Instance Store has already demonstrated its usage in a variety of application domains, examples include the use of ontology based vocabulary to describe documents in "publish and subscribe" applications [17], to annotate data in bioinformatics applications [7] and to annotate web resources such as web pages [5] or web service descriptions [13] in Semantic Web applications. We believe that the existing one could be improved by adding some optimisations, providing extra functionalities and relaxing several assumptions.

A severe restriction made in the Instance Store is that axioms asserting role relationships between pairs of individuals are not allowed. A DL technique, so-called *precompletion* [10], can be used to transform a general ABox (allowing role assertions) into an equivalent *role-free* one. With this pretreatment, the Instance Store will be able to cope with more general ABox datasets.

Precompletions of knowledge bases are built using a set of syntactic rules which extend the ABox of the original knowledge base. It tries to make explicit all the information concerning a single individual by means of relationships which link the individual with other individuals. In other words, additional concept assertions are added into the knowledge base to capture the information carried by role assertions. Because of the possible non-determinism of the precompletion rules, many different[3] precompletions can be derived from a single knowledge base.

4.1 Inverse role

The extension of precompletion for inverse roles is problematic, as an example, let us consider the following ABox

$$\mathcal{A}_1 = \{John\colon \exists hasFriend.(\forall hasFriend^{-1}.(\forall hasChild.\neg female)), Mary\colon female\}$$

Adding role assertion $\langle John, Mary \rangle\colon hasChild$ to ABox $\mathcal{A}_5$ is inconsistent, as the combination of the existential and universal quantification $\exists hasFriend.\forall hasFriend^{-1}.(\ldots)$ pushes extra restrictions on the individual *John* and *Mary* (i.e., the new concept label $\forall hasChild.\neg female$ on *John* and $\neg female$ on *Mary*), and results in a contradiction of $Mary\colon female \sqcap \neg female$.

The example above shows that precompletion might fail in the presence of inverse role because it cannot prevent cases when new concepts might be added to the label of the individuals due to terminological reasoning.

[3] In the worst-case, exponentially many precompletions can be derived from the original knowledge base, in such cases, our approach seems not to be helpful.

4.2 Unique name assumption

As the Instance Store does not respect the Unique Name Assumption (UNA), two separate individuals could be inferred to be identical. When individuals are determined to be the same, information about them (concept assertions and role assertions) should be expanded explicitly in a way such that the consistency of the original knowledge base is preserved—this would clearly require an extension to the original precompletion algorithm.

4.3 Instance retrieval

Another issue one should consider is how to make the *instance retrieval* in the extended Instance Store correct, extra caution has to be taken regarding the predecessors appearing in role assertions. To motivate how the problem could possibly merge, let us consider for example the following ABox and instance retrieval queries:

$$\begin{aligned} \mathcal{A}_2 &= \{John\colon \forall hasChild.female,\ \langle John, Mary\rangle\colon hasChild,\ Mary\colon doctor\} \\ Q_1(x)\ ?-\ &(female \sqcap doctor)(x) \\ Q_2(x)\ ?-\ &(\exists hasChild.doctor)(x) \end{aligned}$$

our precompletion algorithm considers the interaction of role assertions and value restrictions, therefore it must take into account that *Mary* must also be a female (which follows from the fact that she is a child of *John*) and this would make *Mary*'s concept label $female \sqcap doctor$ and hence the answer to Q_1 is $\{Marry\}$. However, it would fail to answer Q_2—although obviously $\{John\}$ should be the answer since he has a child *Mary* who is a *doctor*.

This problem comes directly from the precompletion procedure: when a role assertion is added to the ABox, the precompletion only pushes information to the successor down the role edge but leave the predecessor untouched—clearly some information is lost. The solution is trivial as shown in Definition 2:

Definition 1 (label). *Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, the label of an individual $x \in \mathcal{A}$ is defined as the conjunction of all the concepts in the concept assertions about the individual x:*

$$\mathcal{L}(x) = C_1 \sqcap C_2 \sqcap \cdots \sqcap C_n$$

where $\{x\colon C_i \mid i = 1, \cdots, n\} \subseteq \mathcal{A}$.

Definition 2 (extended label). *Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, the extended label $\mathcal{L}'(x)$ of an individual $x \in \mathcal{A}$ is defined as the conjunction of $\mathcal{L}(x)$ and $\exists R_x.\mathcal{L}'(a_x)$ for each role assertion $\langle a, a_x\rangle\colon R_x$ in $\mathcal{A}$:*

$$\mathcal{L}'(x) := \mathcal{L}(x) \sqcap \bigcap \exists R_x.\mathcal{L}'(a_x) \ \ for\ all\ \langle a, a_x\rangle\colon R_x\ in\ \mathcal{A}$$

Clearly, the information carried by role assertion is therefore made explicit to its predecessor in $\mathcal{L}'(x)$, and $\mathcal{L}'(x)$ instead of $\mathcal{L}(x)$ can be used in the *instance retrieval* operation to fix the problematic situation mentioned above. In the previous $\mathcal{A}_5$'s example, $\mathcal{L}'(John)$ becomes $\{\ \forall hasChild.female \sqcap \exists hasChild.doctor\ \}$, when computing the answer for Q_2 against $\mathcal{L}'(John)$ instead of $\mathcal{L}(John)$ (i.e., $\forall hasChild.female$), it would successfully find out that $\{John\}$ is indeed the answer.

However, when a set of role assertions in the ABox form a cycle (even a role assertion such as $\langle a, a\rangle\colon R$), the simple rolling up procedure is not working anymore—the presence of cycles lead to an endless reference chain.[4]

[4] One possible way of resolving this is to use *fixpoint semantics* to capture the cyclic situation, this approach is still under investigation.

5 Contributions Anticipated

The anticipated main contributions of this research can be summarised in the following points:

- An extension of the *instance store* which relaxes some constraints on the kinds of ABox that can be handled, for example, ABoxes containing role assertions;
- An extended precompletion technique which removes the UNA and adopts optimisations for reducing non-determinism;
- Demonstrate the applicability of the *instance store* in realistic applications;
- An evaluation of the techniques used in the *instance store* against other DL/DB or, if one exists, some other DL+DB applications for supporting Semantic Web annotations reasoning.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. T. Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.
3. G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 316–327, 1996.
4. M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language 1.0 reference, July 2002. Available at `http://www.w3.org/TR/owl-ref/`.
5. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.
6. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.
7. GOA project. European Bioinformatics Institute. `http://www.ebi.ac.uk/GOA/`.
8. T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.
9. V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.
10. B. Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Ann. of Mathematics and Artificial Intelligence*, 18(2–4):133–157, 1996.
11. I. Horrocks, L. Li, D. Turi, and S. Bechhofer. The instance store: Description logic reasoning with large numbers of individuals. Submitted to IJCAR2004, Jan. 2004.
12. I. Horrocks and P. F. Patel-Schneider. Reducing owl entailment to description logic satisfiability. In *Proc. of the 2nd International Semantic Web Conference (ISWC)*, 2003.
13. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 331–339. ACM, 2003.
14. M. Minsky. A framework for representing knowledge. In P. J. Winston, editor, *The psychology of computer visions*, pages 211–277. McGraw-Hill, New York, 1975.
15. M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. The MIT Press, 1968.
16. S. Tessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, University of Manchester, Department of Computer Science, Apr. 2001.
17. M. Uschold, P. Clark, F. Dickey, C. Fung, S. Smith, S. U. M. Wilke, S. Bechhofer, and I. Horrocks. A semantic infosphere. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 882–896. Springer, 2003.