Making an Abox Persistent

Quentin Elhaik and **Marie-Christine Rousset**
L.R.I U.R.A C.N.R.S
University of Paris-Sud
Building 490, 91405 Orsay Cedex, France
{quentin,mcr}@lri.lri.fr

1 Introduction

Much of the research in Description Logics (DL) has concentrated so far on the study of computational complexity of reasoning problems and on the design of reasoning algorithms (see e.g. [3] for a recent survey). However, the implementation of DL systems and their use in applications involve many other aspects. In particular, in several domains of applications (e.g configuration [4]), it would be very useful to manage large and persistent Aboxes (relative to a fixed Tbox). On the other hand, several database applications would benefit from being offered the expressive and reasoning power of DLs.

In this paper, we address the problem of encoding and storing an Abox (relative to a fixed Tbox) as a database. Our approach consists of relying on database management systems for storing, efficiently manipulating and querying large amount of data, while providing the reasoning services usually offered by a DL system.

We consider Aboxes that are too large to be loaded in main memory. The solution that we propose is to store them in a database so that, at each update, it is logically saturated: all the facts which can be logically entailed from the database, the new fact and the Tbox are explicitly added and stored. By doing so, at query time, answering queries is straightforward[1], efficient and not requiring any further inference or reasoning.

The core issue is to determine, at each update, the subpart of the database encoding the Abox which is *relevant* to the update. Once the relevant facts have been identified, they can be loaded in the main memory and standard DL inference procedures can be applied to them in order to determine all the new facts that can be logically entailed (and thus have to be added). Speaking informally, a subset A of the Abox is relevant to an update f if it is sufficient to consider A together with f to infer all the facts that can be logically entailed from f, the whole Abox and the Tbox.

The problem is to characterize the set of relevant facts for f, given the schema of the Abox, and not the Abox itself. The reason is that the Abox that has to be updated and saturated is not directly available. It is stored as a database and we can just query it to search and load facts from it. The only way to access the set of facts that are relevant to an update is by queries over the database encoding the Abox. Those queries depends on the encoding choices (i.e the relations that have been chosen to encode the different facts of the Abox) and also on the DL language that is considered because they must account for the different constructors that are available for building facts and concepts.

In this paper, we consider Aboxes composed of instances of concepts and roles in the setting of the DL language (referred to as core-CLASSIC) having the constructors $\sqcap, \forall, (\geq\ n\, R), (\leq\ n\, R)$ and $\neg$ (on basic concepts only).

The paper is organized as follows. Section 2 provides the basic definitions of the problem that we consider. In section 3, we propose an encoding for core-CLASSIC Aboxes. In section 4, we provide an algorithm which, given a database encoding an Abox $\mathcal{A}$, and a fact f to be added, computes the set of facts in the database that are relevant to the fact f.

[1] while being careful with the mismatch open/close world assumption

2 Preliminaries

Let $\mathcal{V}$ a basic vocabulary composed of basic concepts and roles, and let $\mathcal{L}$ a description logic language defined by a set of constructors. We first define the notion of Abox *relative to* $\mathcal{V}$ and $\mathcal{L}$. We then define the notion of underlying Tbox. Finally we define the relevance scope of an update.

Definition 2.1: *Let $\mathcal{V}$ be a vocabulary composed of basics concepts and roles, $\mathcal{L}$ be a description logic language defined by a set of constructors, and $\mathcal{O}$ a set of individuals. An Abox relative to $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{O}$ is a set of facts of the form $C(a)$, or $R(a,b)$, where C and R are concept and role expressions that can be built from $\mathcal{V}$ using the constructors of $\mathcal{L}$, and a and b are individuals of $\mathcal{O}$.*

Example 2.1:
Let $\mathcal{A}_1$:

$\{(\leq 1\, R_0)(a_0), (\geq 2\, R_1)(a_1), (\forall R_2 A_2)(a_1),$
$(\forall R_0(\forall R_2.A_2))(a_3), R_0(a_0, a_1), R_1(a_1, a_2)\}$

$\mathcal{A}_1$ is an Abox relative to the vocabulary $\mathcal{V}_1 = \{A_1, A_2, A_3, R_0, R_1, R_2\}$, the language core-CLASSIC, and the set $\mathcal{O}$ of individuals $\{a_0, a_1, a_2, a_3\}$.

Definition 2.2: *Let $\mathcal{A}$ be an Abox relative to $\mathcal{V}$, $\mathcal{L}$ and $\mathcal{O}$. The underlying terminology is composed of a set of concept definitions giving a concept name to each concept expression that is a conjunct in concept expressions appearing in $\mathcal{A}$.*

Example 2.2: The terminology corresponding to the Abox $\mathcal{A}_1$ is

$C_1 := (\leq 1\, R_0)$
$C_2 := (\geq 2\, R_1)$
$C_3 := (\forall R_2 A_2)$
$C_4 := (\forall R_0(\forall R_2.A_2))$

Given an Abox $\mathcal{A}$ and a fact f, let $Th(\mathcal{A} \cup f)$ be the set of facts that can be logically entailed from $\mathcal{A} \cup f$, and let $\Delta(\mathcal{A} \cup f)$ the set of facts that can be logically entailed from $\mathcal{A} \cup f$ while not belonging to $\mathcal{A} \cup f$.

$$Th(\mathcal{A} \cup f) = \mathcal{A} \cup f \cup \Delta(\mathcal{A} \cup f)$$

$\Delta(\mathcal{A} \cup f)$ includes the facts that have to be explicitly added to the database as a result of the adding of f if we want the database to be saturated after the update by f.

Existing algorithms (e.g constraint-based algorithms, CLASSIC propagation algorithm) can be used to compute $\Delta(\mathcal{A} \cup f)$. The problem is that for the use of those algorithms, it is implicitly assumed that the whole Abox $\mathcal{A}$ is provided as an input.

We consider a setting in which such an assumption does not hold. The problem is therefore to load a *subset of relevant facts* that are sufficient for saturating the Abox when adding the fact f. Existing DL algorithms will be applied to that subset.

Definition 2.3: *Given an Abox $\mathcal{A}$ and a fact f to be added, the set of relevant facts $\mathcal{R}el(f, \mathcal{A})$ for f in $\mathcal{A}$ is the minimal subset S_A of $\mathcal{A}$ such that $\Delta(\mathcal{A} \cup f) = \Delta(S_A \cup f)$.*

Example 2.3:
Let $f : (\forall R_1 A_1)(a_1)$.

$$\mathcal{R}el(f, \mathcal{A}_1) = \{(\leq 1\, R_0)(a_0), R_0(a_0, a_1), R_1(a_1, a_2)\}$$

Those facts, together with f, are sufficient to compute $\Delta(\mathcal{A}_1 \cup f)$.

$$\Delta(\mathcal{A}_1 \cup f) = \{A_1(a_2), (\forall R_0(\forall R_1 A_1))(a_0)\}$$

3 Encoding core-CLASSIC Aboxes

Recall that all the conjuncts appearing in concept expressions in the Abox are named and defined in the Tbox. There is no definition of conjunction of concepts in the Tbox.

- The facts of the Abox which are of the form $(C1 \sqcap \ldots \sqcap Cn)(a)$ are represented in the database by the encoding of the n facts $a : C1 \ldots a : Cn$. In the same way, $(\forall R(C1 \sqcap \ldots \sqcap Cn))(a)$ is represented by the encoding of the n facts $(\forall RC1)(a) \ldots (\forall RCn)(a)$,
- all the facts of the Abox that are stored in the database are named by constants $f_1, \ldots, f_n$,
- the two kinds of facts (instances of a concept or of a role predicate) are encoded by two EDB predicates $ConceptFact$ and $RoleFact$
  - $ConceptFact(f, c, a)$ denotes that the fact named f and corresponding to the belonging of the constant a to the concept named c is present in the Abox.
  - $ConceptRole(f, r, a, b)$ denotes that the fact named f and corresponding to the relationship between a and b through the role r is stated in the Abox
- each form of concept (depending on its main constructor) is encoded by as many EDB predicates as constructors in the language.
  - $ForAll(cn, r, c)$ denotes that the subconcept named cn and corresponding to the concept $\forall rc$ is refered in the Abox, and present in the Tbox,
  - $Atleast(cr, r, n)$ denotes that the subconcept named cn and corresponding to the concept $(\geq n\, r)$ is refered in the Abox, and present in the Tbox,
  - $Atmost(cr, r, n)$ denotes that the subconcept named cn and corresponding to the concept $(\leq n\, r)$ is refered in the Abox, and present in the Tbox,
  - $Basic(A)$ denotes that the basic concept A is refered in the Abox, and present in the Tbox,
  - $NegPrimitive(A)$ denotes that the negation of basic concept $\neg A$ is refered in the Abox, and present in the Tbox.

4 Updating a core-CLASSIC Abox

We consider updates that are only adding facts to the Abox. We can add either a fact of the form $C(a)$ or a fact of the form $R(a, b)$, where C is a core-CLASSIC concept expressions C, and R is a basic role. We assume that the underlying Tbox can be loaded in main memory so that all the subsumption relations and disjunction relations between concepts are known (or can be computed on demand).

We assume that the databases which encode our Aboxes are saturated and we want to guarantee that they are saturated after each update.

Definition 4.1: *A database $\mathcal{DB}$ encoding an Abox $\mathcal{A}$ is saturated if for every fact f*

- *if f can be put in the form $(C_1 \sqcap \ldots \sqcap C_n)(a)$, if $\mathcal{A} \models f$ then $\forall i, C_i(a) \in \mathcal{DB}$*
- *if f is of the form $(\leq n\, R)(a)$, if $\mathcal{A} \models f$ then $(\leq n_m\, R)(a) \in \mathcal{DB}$, where n_m is the minimal integer n such that $\mathcal{A} \models (\leq n\, R)(a)$*
- *if f is of the form $(\geq n\, R)(a)$, if $\mathcal{A} \models f$ then $(\geq n_M\, R)(a) \in \mathcal{DB}$, where n_M is the maximal integer n such that $\mathcal{A} \models (\geq n\, R)(a)$*
- *otherwise, if $\mathcal{A} \models f$ then $f \in \mathcal{DB}$*

In the case of the adding of $C(a)$,

- either, the concept C already belong to the Tbox, or is a conjunction of concepts that belong to the Tbox, and therefore the Tbox does not have to be updated,
- or, it is not the case, and then the Tbox has to be updated (by adding and naming the conjuncts of C)

The updating of the database encoding the Abox, and possibly of the underlying Tbox, proceeds in two steps.

The first step takes care of all that concerns the Tbox (Tbox update and subsumption reasoning). It consists of possibly updating the Tbox and of determining the set *ToAdd* of the facts that will have to be added explicitly to the database encoding the Abox in the second step.

If the fact to be added is of the form $R(a, b)$, the first step simply set *ToAdd* to be $\{R(a, b)\}$.

If the fact to be added is of the form $C(a)$, let $C_1 \ldots C_n$ be the conjuncts composing the concept C ($n = 1$ if C is not a conjunction of concepts)[2].

1. Let *ToAdd* be the set of facts $\{C_1(a), \ldots, C_n(a)\}$
2. For every conjunct CC of C (possibly C itself if it is not a conjunction of concepts) which does not belong to Tbox, add a new definition in the Tbox for CC.
3. For every conjunct CC of C, for every concept D in the Tbox such that $CC \prec D$ [3], add $D(a)$ to the set *ToAdd*: $D(a)$ has to be explicitly added to the database encoding the Abox, if we want it to be saturated.
4. If C is equivalent to $(\forall R.\bot)$, add $(\leq 0\, R)(a)$ to the set *ToAdd*.

The second step consists of adding to the database encoding the Abox each fact of the set *ToAdd* that has been determined at the first step. Those facts, together with facts that are already stored in the database, can in turn entail new facts (which, therefore will also have to be added to the database). Consequently, it is necessary to determine the set of facts in the database that are relevant to each element of the set *ToAdd*. We describe in section 4.2 an algorithm that determines the set of facts in the database which are relevant to a fact F. This algorithm relies on a set of inference rules which characterizes the logical entailment of facts for a core-CLASSIC Aboxes.

4.1 Characterization of entailment for core-CLASSIC facts

Let $\mathcal{A}$ a core-CLASSIC Abox.

We define an inference relation $\vdash$ on facts by the following set of rules.

R_0: **If** $f \in \mathcal{A}$ **Then** $\mathcal{A} \vdash f$
R_1: **If** $\mathcal{A} \vdash C(a)$ and $C \prec D$ **Then** $\mathcal{A} \vdash D(a)$
R_2: **If** $\mathcal{A} \vdash (\forall R.C)(a), \mathcal{A} \vdash R(a, b)$ **Then** $\mathcal{A} \vdash C(b)$
R_3: **If** $\mathcal{A} \vdash R(a, b_1), \ldots, \mathcal{A} \vdash R(a, b_n), (\forall i \neq j, b_i \neq b_j)$ **Then** $\mathcal{A} \vdash (\geq n\, R)(a)$
R_4: **If** $\mathcal{A} \vdash R(a, b_1), \ldots, \mathcal{A} \vdash R(a, b_n)$, $\mathcal{A} \vdash C(b_1), \ldots, \mathcal{A} \vdash C(b_n), (\forall i \neq j, b_i \neq b_j)$, $\mathcal{A} \vdash (\leq n\, R)(a)$ **Then** $\mathcal{A} \vdash (\forall R.C)(a)$
R_5: **If** $\mathcal{A} \vdash (\forall R.\bot)(a)$ **Then** $\mathcal{A} \vdash (\leq 0\, R)(a)$
R_6: **If** $\mathcal{A} \vdash C(a), \mathcal{A} \vdash D(a)$, and $C \cap D \prec \bot$ **Then** $\mathcal{A} \vdash \bot$

Theorem 4.1: *The set of inference rules is sound and complete: for every Abox $\mathcal{A}$ in core-CLASSIC, for every fact f,*

$$\mathcal{A} \models f \iff \mathcal{A} \vdash f$$

4.2 Algorithm for computing the set of relevant facts to an update

The updates that we consider are the elements of the set *ToAdd*. There are facts of the form $R(a, b)$, or $C(a)$ where C is a concept which cannot be put in the form of a conjunction of concepts (we are in the second step of the updating process). In addition, the facts that can be entailed from $C(a)$ through subsumption have already been treated. Therefore, the rule R_1 and R_5 do not need being considered.

Given a fact f to be added to a saturated database encoding an Abox, the other inference rules are used to determine the *neighborhood of f*, which consists of those facts that are present in the database and which may cause some inference rules to get fired in the presence of

[2] concepts $\forall R.(C_1 \sqcap \ldots \sqcap C_n)$ are considered as conjunctions of concepts

[3] This subsumption can be checked using any subsumption algorithm since the Tbox is supposed to be in the main memory

f. Then, we determine the set $VirtualFacts$ of facts that can be infered from f, its neighborhood and the inference rules. It is done by an iterative forward-chaining process, each step determining the facts that can be infered as an immediate consequence of some inference rules, until no new fact can be obtained.

We define formally the notion of neighborhood of a fact, and we state the property that makes it the core of our algorithm. We then describe our algorithm that determines the set of relevant facts for a given fact f.

Definition 4.2:
Let $\mathcal{DB}_{\mathcal{A}}$ be a saturated database encoding a core-CLASSIC Abox $\mathcal{A}$, and let f be a fact to be added.

- *If f is of the form $C(a)$:*
  $Neighborhood(C(a), \mathcal{DB}_{\mathcal{A}}) =$
  $\{R(a,o) \mid R(a,o) \in \mathcal{DB}_{\mathcal{A}}\}$
  $\cup\{R(o,a) \mid R(o,a) \in \mathcal{DB}_{\mathcal{A}}\}$
  $\cup\{D(a) \mid D(a) \in \mathcal{DB}_{\mathcal{A}}\}$
  $\cup \bigcup_{\{o \mid R(a,o) \in \mathcal{DB}_{\mathcal{A}}\}} \{D(o) \mid D(o) \in \mathcal{DB}_{\mathcal{A}}\}$
  $\cup \bigcup_{\{o \mid R(o,a) \in \mathcal{DB}_{\mathcal{A}}\}} \{D(o) \mid D(o) \in \mathcal{DB}_{\mathcal{A}}\}$
- *If f is of the form $R(a,b)$:*
  $Neighborhood(R(a,b), \mathcal{DB}_{\mathcal{A}}) =$
  $\bigcup_{\{C \mid C(a) \in \mathcal{DB}_{\mathcal{A}}\}} Neighborhood(C(a), \mathcal{DB}_{\mathcal{A}})$
  $\cup \bigcup_{\{D \mid D(b) \in \mathcal{DB}_{\mathcal{A}}\}} Neighborhood(D(b), \mathcal{DB}_{\mathcal{A}})$

It is important to note that the neighborhood of a fact in a database encoding a Abox can be easily obtained by a set of simple SQL queries asked to the database.

Definition 4.3:
Given a Abox $\mathcal{A}$ and a fact f, g is an immediate consequence of $\mathcal{A}$ and f (denoted $f, \mathcal{A} \vdash_1 g$) iff $g \notin \mathcal{A}$ and g is obtained as a consequence of the application of one of the inference rule to $\mathcal{A}$ and f.

The following proposition states that the immediate consequences of an Abox $\mathcal{A}$ and a fact f are contained in the neighborhood of f in the database encoding $\mathcal{A}$.

Proposition 4.1:
Let $\mathcal{A}$ be a core-CLASSIC Abox, $\mathcal{DB}_A$ the saturated database encoding $\mathcal{A}$, and f a fact.
If $f, \mathcal{A} \vdash_1 g$ then $f, Neighborhood(f, \mathcal{DB}_A) \vdash_1 g$.

The algorithm for determining the set of relevant facts to a given fact f in a database $\mathcal{DB}_A$ is the following:

RelevantFacts(f, $\mathcal{DB}_A$):
$VirtualFacts := \{f\}$; $NH := \emptyset$
`repeat`
$NH := NH \bigcup_{f \in VirtualFacts}$ `Neighborhood(f,`$\mathcal{DB}_A$`)`
$VirtualFacts := \{g \mid NH \cup VirtualFacts \vdash_1 g\}$
`until` ($VirtualFacts = \emptyset$) `or` ($\perp \in VirtualFacts$)
return NH

Theorem 4.2: *Let f a fact of the form $C(a)$ or $R(a,b)$ to be added to a core-*CLASSIC *Abox $\mathcal{A}$. Let $\mathcal{DB}_{\mathcal{A}}$ the saturated database resulting from the encoding of $\mathcal{A}$. The set of facts returned by $RelevantFacts(f, \mathcal{DB}_A)$ contains the facts in $\mathcal{DB}$ which are relevant to f:*

$$\mathcal{R}el(f, \mathcal{A}) \subset RelevantFacts(f, \mathcal{DB}_A)$$

5 Conclusion and Perspectives

In this paper, we have presented an approach for combining the storage and querying benefits of database management systems with the reasoning services of DL systems. It is based on encoding an Abox as a relational database that is saturated at each update so that, at query time, answering queries is efficient because not requiring any inference. The saturation step relies on an algorithm that determines the set of relevant facts to a given fact in a saturated database encoding an Abox.

This work has been implemented in the Delphi [1] environment. Delphi provides a relational database embedded in a object-oriented Pascal-like programming environment. We have started a testing protocol for measuring the number of relevant facts w.r.t the size of the Aboxes. This protocol is based on random uniform samples of core-CLASSIC Aboxes relative to a fixed Tbox, which have been generated by a MCMC simulation algorithm [2]. The results that we have obtained so far are preliminary. They concern random Aboxes relative to a a given Tbox having 100 concepts (including 80 % of basic concepts or negation of basic concepts), and to a set of 100 individuals. Those Aboxes, when saturated, contain around 8400 facts. In the current experiment, the percentage of relevant facts to a given fact is very weak (less than 10%). This preliminary experiment needs to be completed.

The updates that we have considered so far are addition operations. The next step will be the handling of retraction operations. When retracting a fact f from a database encoding a given Abox, we have to also retract those facts whose logical entailment depends on f and which cannot be logically entailed without the presence of f. For making it possible, we have to determine and keep trace of the dependency links between facts when they are added. For that purpose, the characterization of the logical entailment by inference rules is a useful tool.

This work is done in the setting of a rather restricted DL language. We claim that this language is a reasonable compromise between the database world and the

DL world. However, one interesting perspective is to investigate how to extend this work to a more expressive DL language.

References

[1] Borland Delphi 3.0

[2] Quentin Elhaik and M-C Rousset and Bernard Ycart. Generating Random Benchmarks for Description Logics. In *Proceedings of DL'98*, 1998.

[3] D.Nardi F.M Donini, M. Lenzerini and A.Schaerf. Reasoning in description logics. In *Principles of Artificial Intelligence*. G.Brewska (ed), Springer Verlag, 1995.

[4] J.R Wright, E.S. Weixelbaum, K. Brown, G.T Vesonder, S.R Palmer, J.I Berman, and H.H Moore. A knowledge-based configurator that supports sales, engineering and manufacturing at AT&T network systems. In *Proceedings of IAAI-93*, 1993.