CEDAR: Efficient Reasoning for the Semantic Web

Samir Amir and Hassan Aït-Kaci
LIRIS, Département Informatique, Université Claude Bernard Lyon 1, Villeurbanne, France.
Email: samir.amir@univ-lyon1.fr, hassan.ait-kaci@univ-lyon1.fr

***Abstract*—We present a new version of CEDAR, a taxonomic reasoner for large-scale ontologies. This extended version provides fuller support for TBox reasoning, checking consistency, and retrieving instances. CEDAR is built on top of the $\mathcal{OSF}$ formalism and based on an entirely new architecture which includes several optimization techniques. Using $\mathcal{OSF}$ graph structures, we define a bidirectional mapping between $\mathcal{OSF}$ structure and the Resource Description Framework (RDF) allowing a translation from $\mathcal{OSF}$ queries into SPARQL for retrieving instances. Experiments were carried out using very large ontologies. The results achieved by CEDAR were compared to those obtained by well-known Semantic Web reasoners such as FaCT++, Pellet, HermiT, TrOWL, and RacerPro. CEDAR performs on a par with the best systems for concept classification and several orders of magnitude more efficiently in terms of response time for Boolean query-answering.**

***Keywords*-Ontology Classification; Query Answering; Binary Encoding;**

I. Introduction

Knowledge reasoning is deriving information that is implied by the information already present in the knowledge base. The latter consists of a TBox, which contains intensional knowledge (schema) in the form of a terminology describing properties of concepts, and an ABox which contains extensional knowledge (data) and is used to describe individuals.

Nowadays, several reasoning systems exist in the area of the Semantic Web (SW). Most of these reasoners are based on top of Description Logic (DL) and use a variation of Tableau method as reasoning algorithms. The limitation of these well known methods for large scale reasoning has been documented heavily in the literature [17], [8].

The robustness and scalability of large-scale reasoning have been addressed as part of the CEDAR project[1] where we proposed a new reasoner. It is built on top of the $\mathcal{OSF}$ (Ordered-Sorted Features) formalism and based on an entirely new architecture which incorporates several optimization techniques [2], [6]. In its initial version, the capability of the CEDAR reasoner was limited to taxonomy reasoning. It beat all OWL reasoners by several orders of magnitude in term of Boolean query answering. In addition, its performance regarding classification was comparable with reputedly efficient Semantic Web reasoners for large scale ontologies.

In order to tackle fuller challenges of the Semantic Web, more expressiveness power is required. This new work presents a more expressive reasoner adding notable improvements over the previous one. While the previous version could only support bare taxonomies, this new version allows role-oriented Description Logic concepts; namely, universal and existential role concepts. We demonstrate that the performance improvement of the previous reasoner over the state of the art is maintained with this new expressive power.

In the new version, we add: (1) functional attributes (called *features*); and, (2) set-valued features (to represent DL roles). In addition to the reasoning services provided by the previous version such as classification and cycle detection, in its current version, CEDAR checks the consistency of queries by normalizing a TBox. Finally, an approach for retrieving instances is proposed for translating $\mathcal{OSF}$ queries into SPARQL. A TBox is then used to optimize SPARQL queries using $\mathcal{OSF}$ reasoning.

Our experimental evaluation with the large-scale ontologies showed that the new version of CEDAR remains the best of all existing reasoners in term of classification and several orders-of-magnitude more efficient in terms of response time for TBox reasoning. The obtained results will be discussed in detail in this paper.

The remainder of this paper is organized as follows: we start with a preliminary section to give an overview about ψ-terms and $\mathcal{OSF}$ formalism. In Section III, we describe the building blocks of CEDAR reasoner, namely; TBox reasoning and query rewriting. In Section IV, we present and discuss the experimental results. Finally, we present a conclusions in Section V.

II. Preliminary

A. Order-sorted feature constraints

In this section, we recall briefly ψ-terms that represent the basis of a logic of record structures called $\mathcal{OSF}$ logic. In [4], ψ-terms were proposed as flexible record structures for logic programming. Indeed, ψ-terms can be seen as a generalization of first-order. However, ψ-terms are of wider interest (see [5] for more details). The easiest way to describe a ψ-term is with the example shown in Figure 1. Here is a ψ-term that may be used to denote a generic *professor* object structure:

```
X:professor(name ⇒ string, age ⇒ 30,
teachesAt ⇒ Y:university (location ⇒
string, established ⇒ 1988))
```

Figure 1. Example of sort and features definition using $\mathcal{OSF}$ syntax

[1]http://cedar.liris.cnrs.fr/

Namely, a 30 year-old professor who has a name which is of type string, and teaches at university. The university has a location which is of type string, and established in 1988. This expression looks like a record structure. Like a typical record, it has field names; i.e., the symbols on the left of $\Rightarrow$. We call these *feature* symbols. In contrast with conventional records, however, ψ-terms can carry more information. Namely, the fields are attached to *sort*[2] symbols (e.g., professor, name, string, 30, etc). These sorts may indifferently denote individual values (e.g., 30) or sets of values (e.g., university, string). In fact, values are assimilated to singleton-denoting sorts. Sorts are partially ordered so as to reflect set inclusion; $professor < employee$ means that all professors are employees. More details about $\mathcal{OSF}$ algebra and constraint-solving methods can be found in [6].

B. $\mathcal{OSF}$ for the Semantic Web

RDF is a standard formal notation, proposed by World Wide Web Consortium (W3C) to represent data on the Web. An RDF dataset is a set of RDF triples and each triple has a subject, a predicate and an object. An RDF dataset forms a directed, labeled graph, where subjects and objects are vertices and predicates are labels on the directed edges.

Figure 1 shown in the previous section depicts that the $\mathcal{OSF}$ formalism describes data that takes the form of rooted labelled graphs. This enables transforming the $\mathcal{OSF}$ into RDF, which rely on a set of transformation rules that we have defined in this paper. Figure 2 shows an example of an RDF graph corresponding to the ψ-term shown by Figure 1.

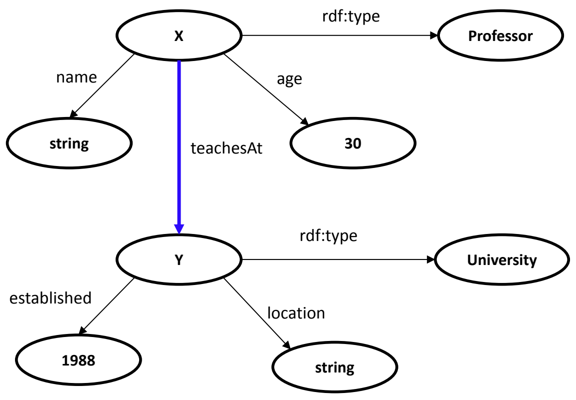

Figure 2. RDF Graph structure

III. CEDAR Reasoner

CEDAR reasoner is an reasoner built on top of $\mathcal{OSF}$ formalism. In its previous version [7], it was one of the fastest reasoners in term of classification. For Boolean query answering, CEDAR outperformed all OWL reasoners by several orders of magnitude. A such efficiency comes from the combination of logic programming techniques [6] and binary encoding [3]. In the previous version, we were supporting only taxonomies. Indeed, in order to tackle the challenges of the Semantic Web, a more expressive power was required.

In its new version we shall present in this paper, CEDAR has been extended by adding more constructs, mainly *features* and *aggregates*. The specification of each constructs is discussed in Section III-A where a comparison with Description Logic ($\mathcal{DL}$) based constructs must be provided. CEDAR whose architecture is shown in Figure 3, is a complete reasoner which performs on TBox and ABox levels. First of all, it starts with a classification phase performed at the TBox level which includes cycle detection, transitive closure performing and features propagation. Then, the encoded TBox is saved on the disk independently from the ABox and used for query normalisation and consistency checking. Finally, for retrieving instances, the query in normal form is translated to SPARQL. In the Section III-B, we discuss the classification and the query normalisation steps and give an example for SPARQL query generation process from ψ-term.

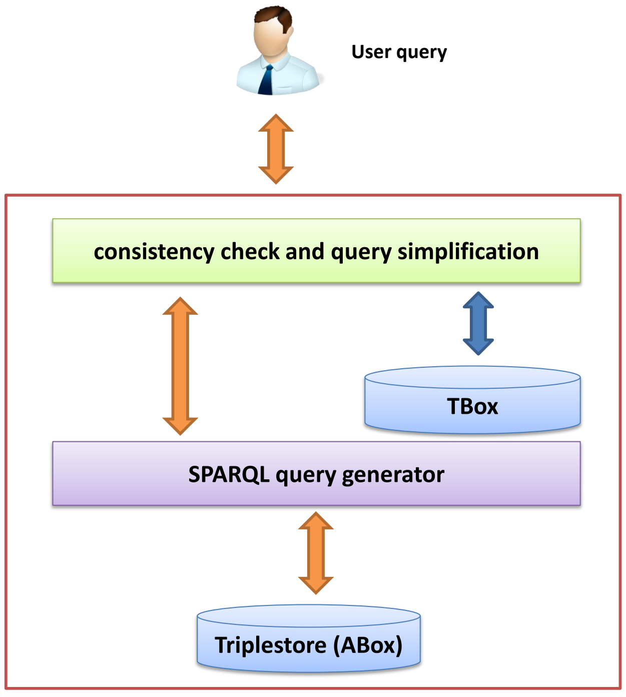

Figure 3. CEDAR Reasoner building blocks

A. $\mathcal{OSF}$ vs $\mathcal{DL}$

1) roles and features: $\mathcal{OSF}$ formalism deals with functional features while $\mathcal{DL}$ is based on relational features. A relational feature is a binary relation or, equivalently, a set-valued function. In other words, a multi-valued functional attribute may be aggregated into sets. For instance, given a concept *professor* denoting a set of professors, and a concept *course* denoting a set of courses. Using $\mathcal{DL}$ semantics, we can define the role *teaches* with

[2]We use "sort" as a synonym of atomic "class" or "concept." In other words, sorts are partially ordered symbols.

domain *professor* and range *course* to denote the set of all the pairs of professors and the courses they teach (*i.e.,* $[\![teaches]\!] \subseteq [\![professor]\!] \times [\![course]\!]$).[3]

In $\mathcal{OSF}$ logic, a role is seen as a set-denoting function [1]. This is crucial because $\mathcal{OSF}$ inference rules, based on graph unification, rely on it in order to be correct [6]. It is for this reason that we propose to represent a relational role as a set-valued function. Indeed, for any sets S and $S^{'}$, a binary relation $r \subseteq S \times S^{'}$ can be seen as a function $f_r : S \to \mathcal{P}(S^{'})$, where $\mathcal{P}(S^{'})$ is the powerset of $S^{'}$; *i.e.,* the set of all subsets of $S^{'}$. Then,

$$\forall x \in S, f_r(x) \overset{\text{DEF}}{=\!=} \{ y \in S^{'} \mid \langle x,y \rangle \in y \} \tag{1}$$

However, for this to be possible, basic $\mathcal{OSF}$ logic must be extended for a new syntactic construct denoting the powerset of a concept. Given a concept c, we write a **set-of(c)** the concept denoting the sets of subsets of $[\![c]\!]$, i.e.,

$$[\![\textbf{set-of(c)}]\!] \overset{\text{DEF}}{=\!=} \mathcal{P}([\![c]\!]). \tag{2}$$

Considering, for example, the *teaches* role with domain *professor* and range *course*, this can now be expressed as the functional feature:

$$f_{teaches} : professor \to \textbf{set-of(course)}$$

to denote the function:

$$[\![f_{teaches}]\!] : [\![professor]\!] \to \mathcal{P}([\![course]\!])$$

that associates to a professor the set of course he teaches. More details about the semantic features and sub-features can be found in [1].

2) Property Restrictions: The semantics of universal and existential roles in a Description Logic language (such as OWL) is given by:

$$\begin{array}{lcl} [\![\forall R.C]\!] & \overset{\text{DEF}}{=\!=} & \{ x \in \mathcal{D} \mid \textbf{range}(R) \subseteq [\![C]\!] \} \\ [\![\exists R.C]\!] & \overset{\text{DEF}}{=\!=} & \{ x \in \mathcal{D} \mid \textbf{range}(R) \cap [\![C]\!] \neq \emptyset \} \end{array}$$

where $\textbf{range}(R) \overset{\text{DEF}}{=\!=} \{ y \mid \exists x \, \langle x,y \rangle \in [\![R]\!] \}$.

This, in $\mathcal{OSF}$ logic, corresponds respectively to:

$$\begin{array}{lcl} [\![\forall R.C]\!] & \overset{\text{DEF}}{=\!=} & \bigcup_{i=1}^{n} A_i \textbf{ s.t. } B_i \subseteq [\![C]\!] \\ [\![\exists R.C]\!] & \overset{\text{DEF}}{=\!=} & \bigcup_{i=1}^{n} A_i \textbf{ s.t. } B_i \cap [\![C]\!] \neq \emptyset \end{array}$$

where:

$$[\![R]\!] \overset{\text{DEF}}{=\!=} \bigcup_{i=1}^{n} A_i \times B_i. \tag{3}$$

This is still not conspicuously expressed in terms of $\mathcal{OSF}$ Logic since one needs yet to know how this may be defined using only functional features and the **set-of** construct. In order to do so, we simply use the fact that a role R defined as a binary relation such as Equation (3) is equivalently definable using a set-valued functional feature $f_R : A_i \to \mathcal{P}(B_i)$, whereby:

$$\forall x \in A_i; \forall y \in B_i; \;\; xRy \iff y \in f_R(x)$$

for $i = 1, \ldots, n$. So the two following definitions express equivalently Description Logic's universal and existential role concepts, respectively, using only $\mathcal{OSF}$ primitives:

$$\begin{array}{lcl} [\![\forall R.C]\!] & \overset{\text{DEF}}{=\!=} & \bigcup_{i=1}^{n} A_i \textbf{ s.t. } f_R(A_i) \subseteq [\![\textbf{set-of}(C)]\!] \\ [\![\exists R.C]\!] & \overset{\text{DEF}}{=\!=} & \bigcup_{i=1}^{n} A_i \textbf{ s.t. } f_R(A_i) \cap [\![\textbf{set-of}(C)]\!] \neq \emptyset \end{array}$$

This is because:

$$f_R(A_i) \subseteq \mathcal{P}(B_i) \subseteq \mathcal{P}([\![C]\!]) = [\![\textbf{set-of}(C)]\!].$$

B. Consistency check and query normalization

In our previous work [2], we proposed a binary encoding approach for representing taxonomies. An efficient classification algorithm was also proposed for calculating transitive closure.

Consistency check and query normalization are two primordial steps which precede query execution. Prior to this, a classification is performed in order to compute all implicit subsumptions relationships and properties. An efficient classification algorithm was proposed in our previous work [2]. In the current version, we have extended the existing algorithm to perform features propagation.

Once classification is performed, the encoded ontology is saved on the disk once and for all. It is used to check queries constituency check and normalization. Indeed, before executing a query, it is obvious that there is no need to scan the ABox to retrieve instances if the query is not consistent. Moreover, making a query in its normal form allows to reduce the search space for instance retrieval part. Table I shows the binary encoding of the partial order set (poset) shown in the Figure 4. In [2], we developed a binary encoding technique in that allows to find the Maximum Lower Bound (MLB) for each Boolean query. Thus, it can be used for ψ-terms normalization since the features refer to the binary codes corresponding to their domain. Figure 4 shows an example of an ontology describing a set of academic employees and organizations. The binary codes corresponding to the taxonomy is shown by Table I. The features *doResearch* and *teachesAt* have researcher and teacher as domain respectively. In order to fully understand the process, let us consider the query Q_1 corresponding to the given ψ-term:

```
  person(doResearch ⇒ set-of(researchCenter),
teachesAt ⇒ set-of(university))
```

Q_1 aims to retrieve the set of persons which are teaching at *university* and doing research at *researchCenter* at the same time. Indeed, we can clearly identify the sort *professor* is the most specific one. In fact, the sort corresponding to *professor* is the MLB of the root sort (*person*) and the sorts which are the domains of the

[3]We use the notation [[foo]] to denote the meaning of *foo*.

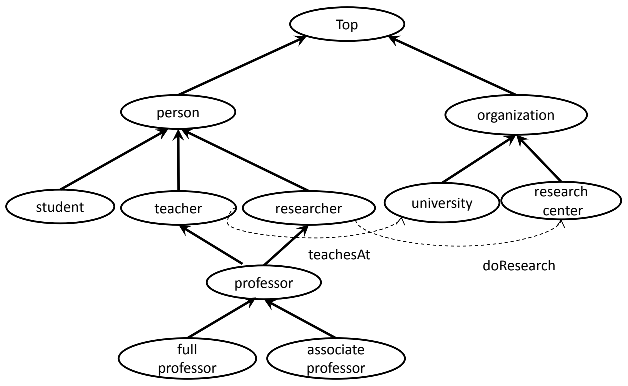

Figure 4. Example of taxonomy with features

Table I
BINARY ENCODING CORRESPONDING TO THE POSET SHOWN IN FIGURE 4).

Sort	Code
Top	11111111111
organization	01110000000
university	00100000000
research center	00010000000
person	00001111111
researcher	00000100111
teacher	00000010111
student	00000001000
professor	00000000111
associate professor	00000000010
full professor	00000000001

two features presents in Q_1. Namely, *researcher* and *teacher*. The MLB is the intersection (conjunction query) of the binary code corresponding to *person*, *teacher* and *researcher* which are represented by ”00001111111”, ”00000010111” and ”00000100111”. The result of that intersection is ”00000000111” which corresponds to the *professor* sort. Thus the normalized format is :

```
  professor(doResearch ⇒ set-of(researchCenter),
teachesAt ⇒ set-of(university))
```

Following the same reasoning, the query Q_2 given by the ψ-term below returns a bottom value represented by the code ”00000000000”. In this case, the query is considered inconsistent with the TBox because the *doResearch* is not a domain of *student* . Thus, there is no need to search an instance for that query.

```
  student(doResearch ⇒ set-of(researchCenter))
```

C. SPARQL query generation

Once a user query is simplified, a ψ-term is rewritten in SPARQL format for efficient instance retrieval. Figure 5 shows the SPARQL query corresponding to the query Q_1 defined in the Section III-B while Figure 6 shows the SPARQL query for Q_1 in the normalized format.

One can clearly see that the SPARQL query in the normalized format has less constraints than the first one. Therefore, a smaller search space in the ABox part.

```
SELECT  ?x
WHERE
{
?x  rdf:type person.
?x  doResearch ?y
?y  rdf:type researchCenter.
?x  teachesAt ?z
?z  rdf:type university.
}
```

Figure 5. Generated SPARQL from Q_1 without normalization

```
SELECT  ?x
WHERE
{
?x  rdf:type professor.
}
```

Figure 6. Generated SPARQL from Q_1 with normalization

Especially, if a datatype indexing is already performed by the triplestore. However, the query shown in the previous example does not contain value. For this reason, we consider that is not important to include features for SPARQL query generation. This is obviously not the case where features have values as a range. Essentially, in this case, these features must be included in the SPARQL generated query as shown in the example given by Figure 7 which supposes that the name of the student is *Dupont*. Finally, it is important to mention that for each type present in the query, the binary code of must be sent with the query to the triplestore. Indeed, the binary code identifies the subtypes (subsorts) concerned by that query. For instance, for the *professor* type, the binary code ”00000000111” allows to identify the sorts *associate professor* and *full professor* which are the subsorts of *professor*.

```
SELECT  ?x
WHERE
{
?x  rdf:type student.
?x  name     "Dupont"
}
```

Figure 7. Generated SPARQL from Q_1 with normalization and values

IV. EXPERIMENTAL EVALUATION

Classification and query normalization algorithms have been evaluated and compared with other SW reasoners, under the exact same conditions, and over medium and very large ontologies. The characteristics of each ontology in term of number of sorts, and properties are shown in Table II.

Except the taxonomy and properties, we removed all non-necessary axioms. A comparative evaluation was conducted to assess the performance of CEDAR over the existing reasoners which have been implemented by using OWL-API.[4]

[4]http://owlapi.sourceforge.net

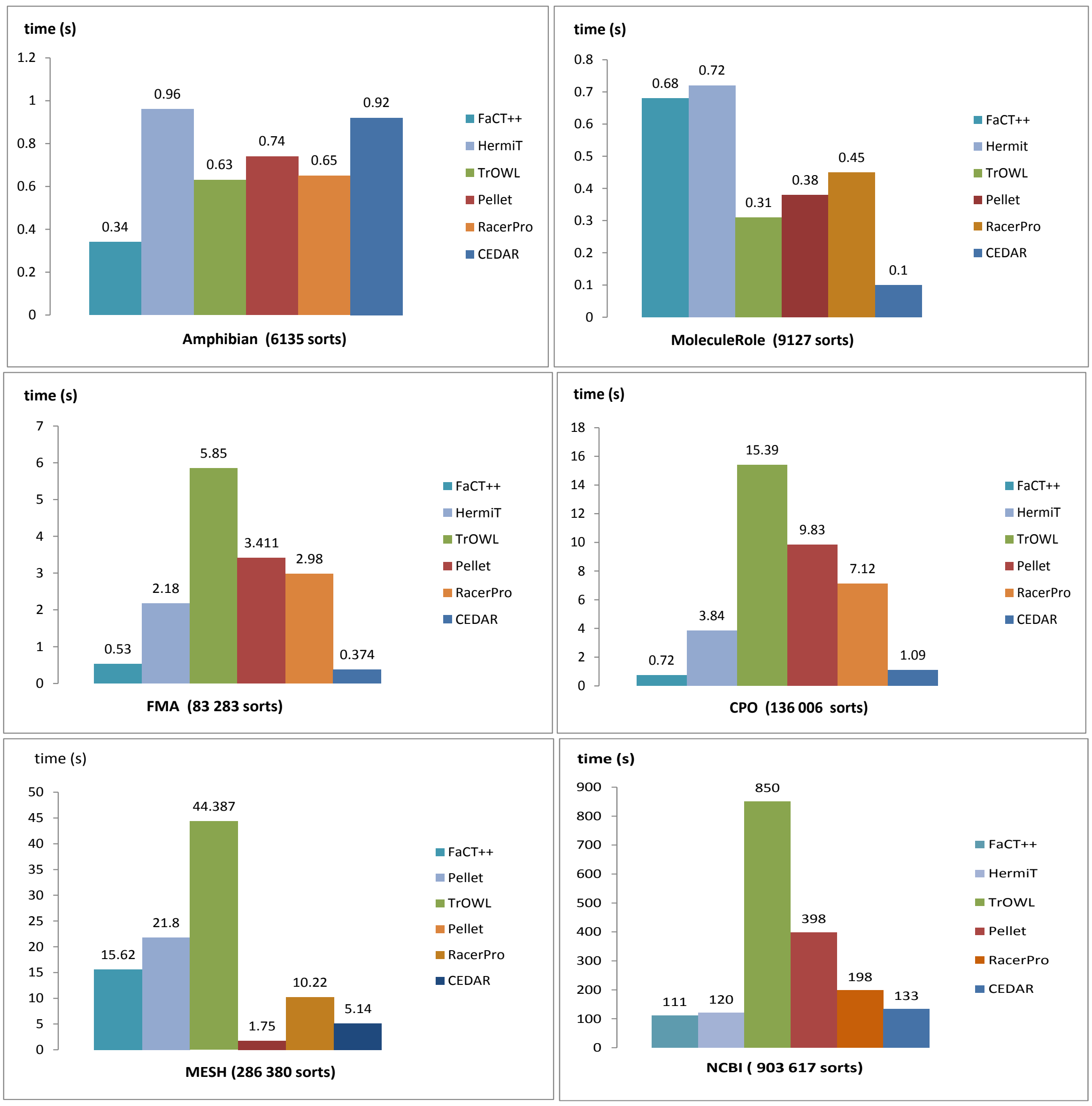

Figure 8. Classification time per reasoner

Table II
BENCHMARKS USED FOR EXPERIMENTS.

Ontology	sorts	properties
Amphibian [12]	6135	30 (generated)
Molecule Role [18]	9127	7
FMA [15]	83 283	77
CPO [10]	136 006	55
MESH [13]	286 380	32
NCBI [9]	903 617	30 (generated)

A. *Classification*

Figure 8 shows the comparative classification time performances for each reasoner on each of the ontologies described above. Note that the SnoRocket [11] reasoner was excluded from this comparison because it does not support the *owl*:*allValuesFrom* constraint.

From Figure 8, it can be seen that CEDAR is not always the best, but it performs among the best reasoners. Especially, for the large scale ontologies where it is roughly ten time faster then TrOWL. This latter is the worst in term of classification performances. This is due to the fact that TrOWL applies syntactic and semantic approximation from OWL2 to OWL-QL [14]. Contrary to what has been stated in [16], FaCT++ performs better than HermiT for all ontologies. This is perhaps due to the limited number of axioms available for these ontologies (taxonomy and roles). RacerPro and Pellet showed an average performance compared to the rest of reasoners.

B. *TBox Reasoning*

In order to perform the evaluation of TBox reasoning step which is necessary for query normalization, we have randomly generated a set of queries, of the form:

$$X : S(f_1 \Rightarrow S_1, f_2 \Rightarrow S_2, f_3 \Rightarrow Y : S_3(f_4 \Rightarrow S_4, f_5 \Rightarrow S_5), \ldots, f_n \Rightarrow S_n) \;\; for \;\; 10 \leq n \leq 100.$$

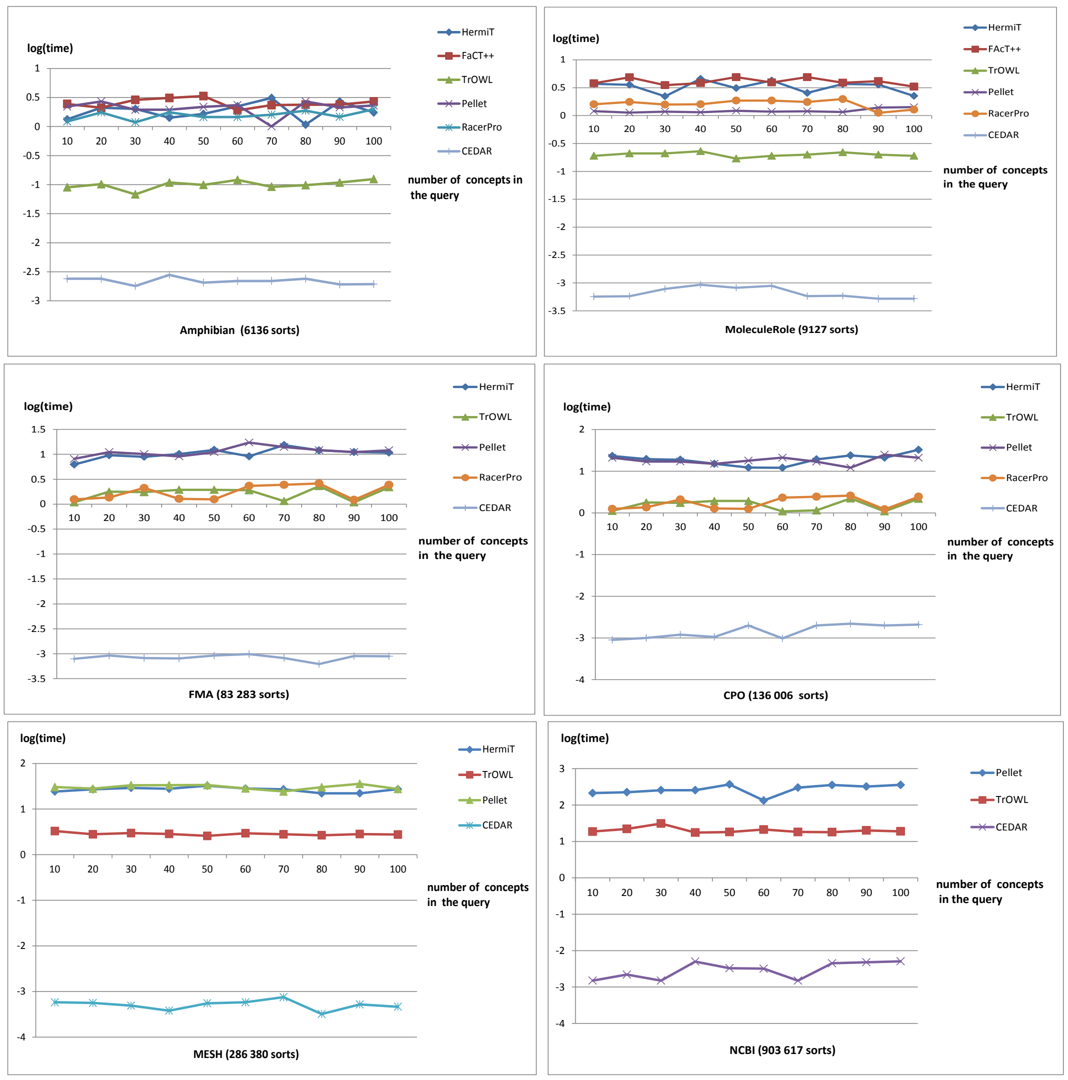

Figure 9. TBox query response time per reasoner

Figure 9 shows the query response time performances in logarithmic scale where CEDAR is orders-of-magnitude faster than other reasoners. It can be seen also that TrOWL has a good performance comparing to other Description Logic reasoners. According to our understanding, the poor performance of TrOWL is it applies applies syntactic and semantic approximation. Hermit, FaCT++, RacerPro and Pellet show close results for medium scale ontologies. However, FaCT++ could not provide an answer before 30 minutes for the large scale ontologies (FMA, CPO, MESH and NCBI). For NCBI, which is the largest ontology, Except for CEDAR, Pellet and TrOWL, all reasoner took more then 30 minutes to provide an answer.

C. *Discussion*

The results reported in this section show that CEDAR performs well for classification, TBox reasoning and query answering. Indeed, in its first version, CEDAR taxonomic reasoner showed a very good results in term of classification. In the current version, we extended the taxonomy classification algorithm by adding features propagation. This extension was not largely expensive in terms of time since it is based on already encoded taxonomy. For each feature, the set of domain sorts is identified by the binary code of the original sort for which this feature was declared.

The performances we observed for the TBox reasoning step (Section IV-B) is also related to the binary encoding technique where CEDAR was orders-of-magnitude faster than other reasoners. In the current version, we have added *features* and *aggregates* for the CEDAR reasoner. In addition to its name, features are identified by its domain and range. Thereby, TBox reasoning phase uses the binary code for query normalisation and consistency verification process. In fact, query normalization is simply a Boolean operation (conjunction) applied to the binary code corresponding to domains sorts.

V. Conclusion

In this paper, we presented and implemented a new version of CEDAR reasoner which was supporting only taxonomies in its previous version. We added features and aggregates and proved that CEDAR can express DL-based relational roles.

A comparative evaluation was carried using several highly reputed Semantic Web reasoners. Experiments showed that CEDAR performs among the bests SW reasoners in terms of classification and several orders-of-magnitude better in term of TBox reasoning.

As for further work, we are extending this work to enable CEDAR to support more complex queries such as disjunction or filtering. Since $\mathcal{OSF}$ is a powerful description and reasoning language, the main part of the work to be done is to adapt $\mathcal{OSF}$ to the Semantic Web. We are also planing to develop our own triplestore that would be highly efficient in retrieving instances.

References

[1] H. Aït-Kaci, "Conceptual roles as order-sorted features—relations as functional aggregates," CEDAR Project, LIRIS, Département d'Informatique, Université Claude Bernard Lyon 1, Villeurbanne, France, CEDAR Technical Report (forthcoming), 2014.

[2] H. Aït-Kaci and S. Amir, "Classifying and querying very large taxonomies," CEDAR Project, LIRIS, Département d'Informatique, Université Claude Bernard Lyon 1, Villeurbanne, France, CEDAR Technical Report (forthcoming), 2014.

[3] H. Aït-Kaci, R. Boyer, P. Lincoln, and R. Nasr, "Efficient implementation of lattice operations," *ACM Transactions on Programming Languages and Systems*, vol. 11, no. 1, pp. 115–146, January 1989.

[4] H. Aït-Kaci and R. Nasr, "LOGIN: A logic programming language with built-in inheritance," *Journal of Logic Programming*, vol. 3, pp. 185–215, 1986.

[5] H. Aït-Kaci and A. Podelski, "Towards a meaning of life," in *Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming (Passau, Germany)*, ser. Lecture Notes in Computer Science, J. Maluszyński and M. Wirsing, Eds., no. 528. Springer-Verlag, August 1991, pp. 255–274.

[6] H. Aït-Kaci, A. Podelski, and S. C. Goldstein, "Order-sorted feature theory unification," *Journal of Logic Programming*, vol. 30, no. 2, pp. 99–124, 1997.

[7] S. Amir and H. Aït-Kaci, "Cedar: a fast taxonomic reasoner based on lattice operations," in *International Semantic Web Conference (Posters & Demos)*, 2013, pp. 9–12.

[8] K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer, "Comparison of reasoners for large ontologies in the owl 2 el profile," *Semant. web*, vol. 2, no. 2, pp. 71–87, Apr. 2011.

[9] S. Federhen, "The ncbi taxonomy database." *Nucleic Acids Research*, vol. 40, no. Database-Issue, pp. 136–143, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/nar/nar40.html

[10] R. Hoehndorf, M. A. Harris, H. Herre, G. Rustici, and G. V. Gkoutos, "Semantic integration of physiology phenotypes with an application to the cellular phenotype ontology." *Bioinformatics*, vol. 28, no. 13, pp. 1783–1789, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/bioinformatics/bioinformatics28.html

[11] M. J. Lawley and C. Bousquet, "Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner," in *Proceedings of the 2nd Australasian Ontology Workshop: Advances in Ontologies*, T. Meyer, M. A. Orgun, and K. Taylor, Eds., AOW'10. Adelaide, Australia: ACS, December 2010, pp. 45–50.

[12] A. M. Maglia, J. L. Leopold, L. A. Pugener, and S. Gauch, "An anatomical ontology for amphibians." in *Pacific Symposium on Biocomputing*, R. B. Altman, A. K. Dunker, L. Hunter, T. Murray, and T. E. Klein, Eds. World Scientific, 2007, pp. 367–378. [Online]. Available: http://dblp.uni-trier.de/db/conf/psb/psb2007.html

[13] N. L. of Medicine, "Medical subject headings MeSH," 2003.

[14] J. Z. Pan, Y. Ren, N. Jekjantuk, and J. Garcia, "Reasoning the fma ontologies with trowl." in *ORE*, ser. CEUR Workshop Proceedings, S. Bail, B. Glimm, R. S. Gonalves, E. Jimnez-Ruiz, Y. Kazakov, N. Matentzoglu, and B. Parsia, Eds., vol. 1015. CEUR-WS.org, 2013, pp. 107–113.

[15] C. Rosse and J. L. V. Mejino, Jr., "A reference ontology for biomedical informatics: The foundational model of anatomy," *J. of Biomedical Informatics*, vol. 36, no. 6, pp. 478–500, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2003.11.007

[16] R. Shearer, B. Motik, and I. Horrocks, "HermiT: A Highly-Efficient OWL Reasoner," in *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, A. Ruttenberg, U. Sattler, and C. Dolbear, Eds., Karlsruhe, Germany, October 26–27 2008.

[17] K. Srinivas, "Owl reasoning in the real world: Searching for godot," in *Description Logics*, 2009.

[18] S. Yamamoto, T. Asanuma, T. Takagi, and K. I. Fukuda, "The molecule role ontology: An ontology for annotation of signal transduction pathway molecules in the scientific literature: Conference papers," *Comp. Funct. Genomics*, vol. 5, no. 6-7, pp. 528–536, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1002/cfg.v5:6/7