Graph-based Ontology Classification in OWL 2 QL

Domenico Lembo and Valerio Santarelli and Domenico Fabio Savo

Department of Computer, Control and Management Engineering Antonio Ruberti
Sapienza Università di Roma, Italia

SAPIENZA
UNIVERSITÀ DI ROMA

10th Extended Semantic Web Conference (ESWC 2013)
Montpellier, France, May 2013

Ontology classification: the problem of computing all subsumption relationships inferred in an ontology between predicate names in the ontology signature, i.e., name concepts (classes), roles (object-properties), and attributes (data-properties).

Classification is a core service for ontology reasoning, and can be exploited for tasks such as:

- ontology navigation
- ontology visualization
- query answering
- explanation

Designing efficient methods for ontology classification is a challenging issue, since in general it is a costly operation.

Classification by ontology reasoners

Popular reasoners for OWL 2 ontologies, such as **FaCT++**, **Hermit**, **Pellet**, **Racer**, offer optimized classification services for expressive DLs, through algorithms based on model construction through tableau (or hyper-tableau).

Other reasoners such as **ELK**, **Snorocket**, and **JCel** are specifically tailored to intensional reasoning over logics of the $\mathcal{EL}$ family (the logical underpinning of OWL 2 EL), and show excellent performances of ontologies in these languages.

The **CB** reasoner is a *consequence-driven* reasoner for the Horn-$\mathcal{SHIQ}$ DL.

So far, no techniques specifically tailored for classification in OWL 2 QL.

The goal: efficient computation of classification in OWL 2 QL

We provide a new method for **ontology classification in OWL 2 QL.**

A simple idea

Encode the ontology TBox into a graph, and compute the transitive closure of the graph to obtain the ontology classification: take advantage of the analogy between simple inference rules in DLs and graph reachability.

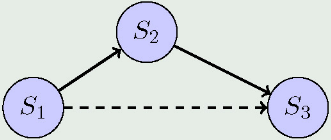
Example

TBox:

- $S_1 \sqsubseteq S_2$
- $S_2 \sqsubseteq S_3$

Inferred inclusion:

- $S_1 \sqsubseteq S_3$



How does graph-based classification work

Classification of an OWL 2 QL ontology:

- for an OWL 2 QL ontology, we show that it is possible to construct a graph whose transitive closure represents the major sub-task for classification of the ontology
- we show that the computed classification only misses "trivial" inclusion assertions inferred by unsatisfiable predicates in the ontology (predicates that always have an empty interpretation in every model of the ontology)
- we provide an algorithm that exploits the transitive closure of the graph, and, through the application of a set of rules, computes all unsatisfiable predicates, allowing to obtain the complete classification of the ontology

Outline

1. Introduction to OWL 2 QL
2. Computation of graph-based ontology classification in OWL 2 QL
3. Implementation and evaluation of the graph-based ontology classification algorithm
4. Conclusions and future works

OWL 2 QL is the "data oriented" profile of OWL 2.

Expressions in OWL 2 QL

$$B \longrightarrow A \mid \exists Q \qquad\qquad Q \longrightarrow P \mid P^-$$
$$C \longrightarrow B \mid \neg B \mid \exists Q.A \qquad R \longrightarrow Q \mid \neg Q$$

Assertions in OWL 2 QL

$B \sqsubseteq C$ *(concept inclusion)*
$Q \sqsubseteq R$ *(role inclusion)*

We call *positive inclusions* axioms of the form $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \exists Q.A$, and $Q_1 \sqsubseteq Q_2$, and *negative inclusions* axioms of the form $B_1 \sqsubseteq \neg B_2$, and $Q_1 \sqsubseteq \neg Q_2$.

Theorem

Let $\mathcal{T}$ be an OWL 2 QL TBox containing only positive inclusions, and let S_1 and S_2 be two atomic concepts or two atomic roles. $S_1 \sqsubseteq S_2$ is entailed by $\mathcal{T}$ if and only if at least one of the following conditions holds:

1. a set $\mathcal{P}$ of positive inclusions exists in $\mathcal{T}$, such that $\mathcal{P} \models S_1 \sqsubseteq S_2$;
2. $\mathcal{T} \models S_1 \sqsubseteq \neg S_1$.

It follows that $\mathcal{T}$-classification $\equiv \{\Phi_{\mathcal{T}} \cup \Omega_{\mathcal{T}}\}$, where:

- $\Phi_{\mathcal{T}}$ contains only positive inclusions for which statement 1 holds
- $\Omega_{\mathcal{T}}$ contains only positive inclusions for which statement 2 holds

1. Encode positive inclusions in $\mathcal{T}$ into a digraph $\mathcal{G}_{\mathcal{T}}$: each node in $\mathcal{G}_{\mathcal{T}}$ represents a concept or role, and each arc a positive inclusion.

Definition

Let $\mathcal{T}$ be an OWL 2 QL TBox over a signature Σ_P. We call the digraph representation of $\mathcal{T}$ the digraph $\mathcal{G}_{\mathcal{T}} = (\mathcal{N}, \mathcal{E})$ built as follows:

1. for each atomic concept A in Σ_P, $\mathcal{N}$ contains the node A;
2. for each atomic role P in Σ_P, $\mathcal{N}$ contains the nodes P, P^-, $\exists P$, $\exists P^-$;
3. for each concept inclusion $B_1 \sqsubseteq B_2 \in \mathcal{T}$, $\mathcal{E}$ contains the arc (B_1, B_2);
4. for each role inclusion $Q_1 \sqsubseteq Q_2 \in \mathcal{T}$, $\mathcal{E}$ contains the arcs (Q_1, Q_2), (Q_1^-, Q_2^-), $(\exists Q_1, \exists Q_2)$, and $(\exists Q_1^-, \exists Q_2^-)$;
5. for each concept inclusion $B_1 \sqsubseteq \exists Q.A \in \mathcal{T}$, $\mathcal{N}$ contains the node $\exists Q.A$, and $\mathcal{E}$ contains the arcs $(B_1, \exists Q.A)$ and $(\exists Q.A, \exists Q)$;

2. Compute the transitive closure of $\mathcal{G}_{\mathcal{T}}$: $\mathcal{G}^* = (\mathcal{N}, \mathcal{E}^*)$

We denote with $\alpha(\mathcal{E}^*)$ the set of arcs $(S_1, S_2) \in \mathcal{E}^*$ such that both terms S_1 and S_2 denote in $\mathcal{T}$ either two atomic concepts or two atomic roles.

Theorem

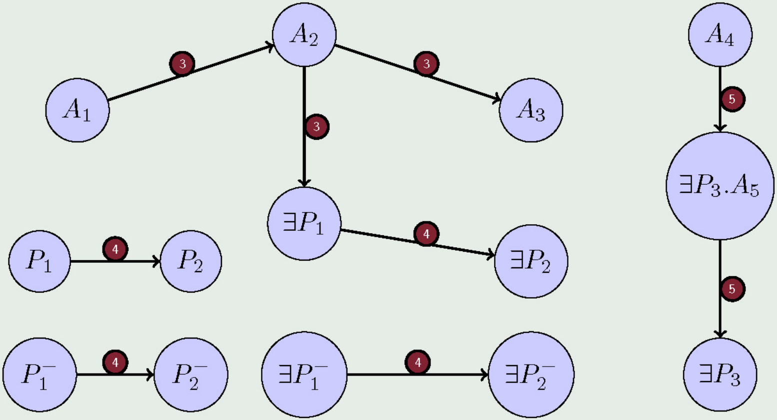
Let $\mathcal{T}$ be an OWL 2 QL TBox and let $\mathcal{G}_{\mathcal{T}} = (\mathcal{N}, \mathcal{E})$ be its digraph representation. Let S_1 and S_2 be two atomic concepts or two atomic roles. An inclusion assertion $S_1 \sqsubseteq S_2$ belongs to $\Phi_{\mathcal{T}}$ if and only if there exists in $\alpha(\mathcal{E}^*)$ an arc (S_1, S_2).

As a consequence of the above theorem, we define algorithm ComputeΦ, that takes as input an OWL 2 QL TBox $\mathcal{T}$, builds $\mathcal{G}_{\mathcal{T}}$, computes $\mathcal{G}^*$, and returns the set $\Phi_{\mathcal{T}}$.

Computation of $\Phi_{\mathcal{T}}$: Example

Example

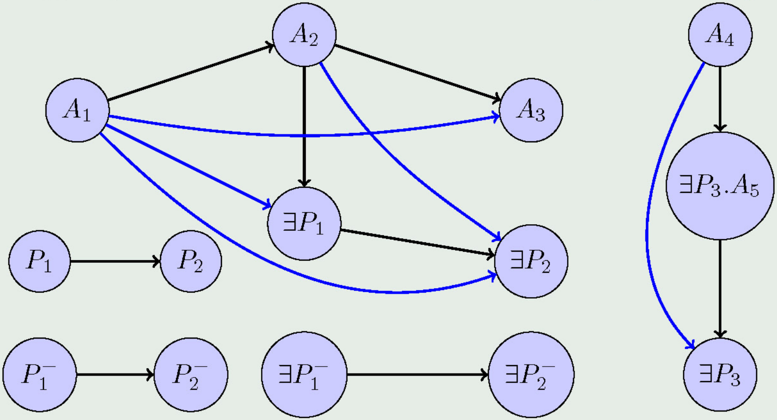
TBox: $A_1 \sqsubseteq A_2$ $\quad A_2 \sqsubseteq A_3$ $\quad A_2 \sqsubseteq \exists P_1$ $\quad A_4 \sqsubseteq \exists P_3.A_5$ *(concept inclusions)*
$P_1 \sqsubseteq P_2$ *(role inclusion)*



Example

TBox: $A_1 \sqsubseteq A_2$ $A_2 \sqsubseteq A_3$ $A_2 \sqsubseteq \exists P_1$ $A_4 \sqsubseteq \exists P_3.A_5$ *(concept inclusions)*

$P_1 \sqsubseteq P_2$ *(role inclusion)*



Computation of $\Omega_{\mathcal{T}}$: algorithm computeUnsat

Algorithm: computeUnsat
Input: an OWL 2 QL TBox $\mathcal{T}$
Output: a set of concept and role expressions

```
Emp ← ∅;
foreach negative inclusion S1 ⊑ ¬S2 ∈ T do
    Emp ← Emp ∪ {predecessors(S1, G*_T) ∩ predecessors(S2, G*_T)}  /* step 1 */
    foreach n1 ∈ predecessors(S1, G*_T) do                          /* step 2 */
        foreach n2 ∈ predecessors(S2, G*_T) do
            if (n1 = ∃Q⁻ and n2 = A) or (n2 = ∃Q⁻ and n1 = A)
            then Emp ← Emp ∪ {∃Q.A};
Emp′ ← ∅;
while Emp ≠ Emp′ do
    Emp′ ← Emp;
    foreach S ∈ Emp′ do
        foreach n ∈ predecessors(S, G*_T) do
            Emp ← Emp ∪ {n};                                          /* step 3 */
            if n = P or n = P⁻ or n = ∃P or n = ∃P⁻                  /* step 4 */
            then Emp ← Emp ∪ {P, P⁻, ∃P, ∃P⁻};
            if there exists B ⊑ ∃Q.n ∈ T
            then Emp ← Emp ∪ {∃Q.n};
return Emp.
```

- The set **predecessors**$(n, \mathcal{G}^*)$ contains n and all n' s.t. $\mathcal{G}^*$ contains (n', n).

Computation of $\Omega_{\mathcal{T}}$: algorithm computeUnsat

For each $S_1 \sqsubseteq \neg S_2$, computes **predecessors**$(S_1, \mathcal{G}^*_{\mathcal{T}})$ and **predecessors**$(S_2, \mathcal{G}^*_{\mathcal{T}})$:

(Step 1) all predicates whose corresponding nodes occur in both **predecessors**$(S_1, \mathcal{G}^*_{\mathcal{T}})$ and **predecessors**$(S_2, \mathcal{G}^*_{\mathcal{T}})$ are unsatisfiable;

(Step 2) all qualified existential roles $\exists Q.A$ whose node $\exists Q^-$ occurs in **predecessors**$(S_1, \mathcal{G}^*_{\mathcal{T}})$ (resp. **predecessors**$(S_2, \mathcal{G}^*_{\mathcal{T}})$**)** and node A in **predecessors**$(S_2, \mathcal{G}^*_{\mathcal{T}})$ (resp. **predecessors**$(S_1, \mathcal{G}^*_{\mathcal{T}})$) are unsatisfiable.

Further unsatisfiable predicates are identified through a cycle, in which:

(Step 3) if $S \in$ Emp, then all expressions corresponding to the nodes in **predecessors**$(S, \mathcal{G}^*_{\mathcal{T}})$ are in Emp;

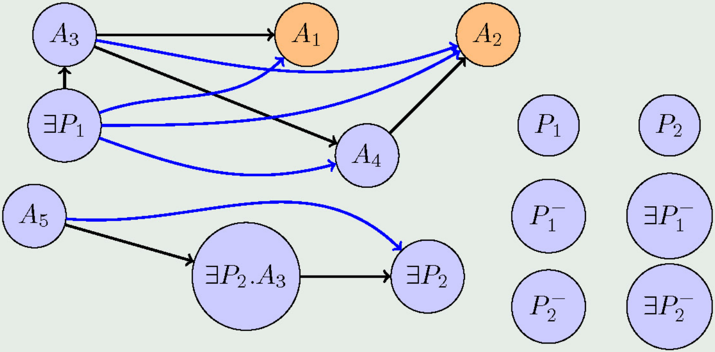
(Step 4)

1. if at least one of the expressions $P, P^-, \exists P, \exists P^-$ is in Emp, then all four expressions are in Emp;
2. for each expression $\exists Q.A$ in $\mathcal{N}$, if $A \in$ Emp, then $\exists Q.A \in$ Emp.

Computation of $\Omega_{\mathcal{T}}$: Example

Example

TBox: $A_3 \sqsubseteq A_4$ $\quad A_4 \sqsubseteq A_2$ $\quad A_3 \sqsubseteq A_1$ $\quad \exists P_1 \sqsubseteq A_3$ $\quad A_5 \sqsubseteq \exists P_2.A_3$ $\quad A_1 \sqsubseteq \neg A_2$

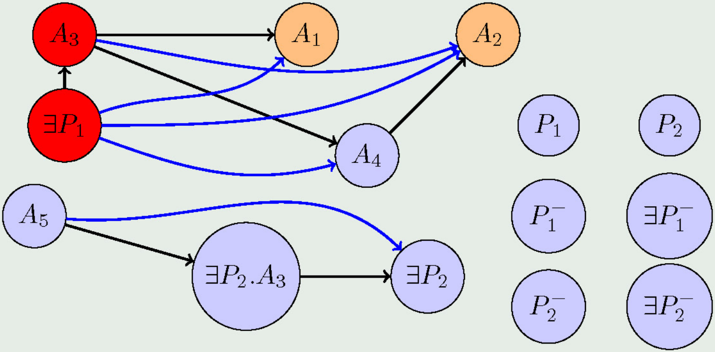


predecessors$(A_1, \mathcal{G}^*_{\mathcal{T}}) = \{A_1, A_3, \exists P_1\}$
predecessors$(A_2, \mathcal{G}^*_{\mathcal{T}}) = \{A_2, A_4, A_3, \exists P_1\}$

Computation of $\Omega_{\mathcal{T}}$: Example

Example

TBox: $A_3 \sqsubseteq A_4$ $A_4 \sqsubseteq A_2$ $A_3 \sqsubseteq A_1$ $\exists P_1 \sqsubseteq A_3$ $A_5 \sqsubseteq \exists P_2.A_3$ $A_1 \sqsubseteq \neg A_2$

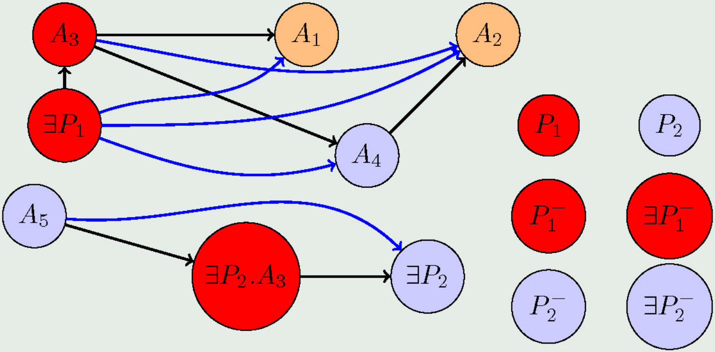


Emp $= \{A_3, \exists P_1\}$

Computation of $\Omega_{\mathcal{T}}$: Example

Example

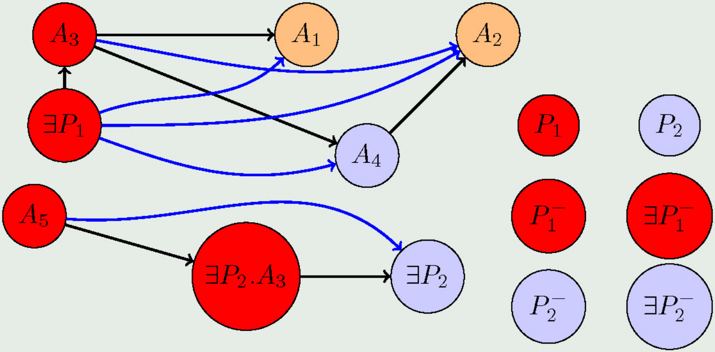
TBox: $A_3 \sqsubseteq A_4$ $A_4 \sqsubseteq A_2$ $A_3 \sqsubseteq A_1$ $\exists P_1 \sqsubseteq A_3$ $A_5 \sqsubseteq \exists P_2.A_3$ $A_1 \sqsubseteq \neg A_2$



Emp $= \{A_3, \exists P_1, P_1, P_1^-, \exists P_1^-, \exists P_2.A_3\}$

Example

TBox: $A_3 \sqsubseteq A_4$ $A_4 \sqsubseteq A_2$ $A_3 \sqsubseteq A_1$ $\exists P_1 \sqsubseteq A_3$ $A_5 \sqsubseteq \exists P_2.A_3$ $A_1 \sqsubseteq \neg A_2$



Emp $= \{A_3, \exists P_1, P_1, P_1^-, \exists P_1^-, \exists P_2.A_3, A_5\}$

Computation of $\Omega_{\mathcal{T}}$

The following theorem shows that algorithm computeUnsat can be used for computing the set containing all the unsatisfiable concepts and roles in $\mathcal{T}$.

Theorem

Let $\mathcal{T}$ be an OWL 2 QL TBox and let S be either an atomic concept or an atomic role in Σ_P. $\mathcal{T} \models S \sqsubseteq \neg S$ if and only if $S \in$ computeUnsat$(\mathcal{T})$.

The following theorem states that the graph-based technique is sound and complete with respect to the problem of classifying an OWL 2 QL TBox.

Theorem

Let $\mathcal{T}$ be an OWL 2 QL TBox and let S_1 and S_2 be either two atomic concepts or two atomic roles. $\mathcal{T} \models S_1 \sqsubseteq S_2$ if and only if $S_1 \sqsubseteq S_2 \in \mathsf{Compute}\Phi(\mathcal{T}) \cup \mathsf{Compute}\Omega(\mathcal{T})$.

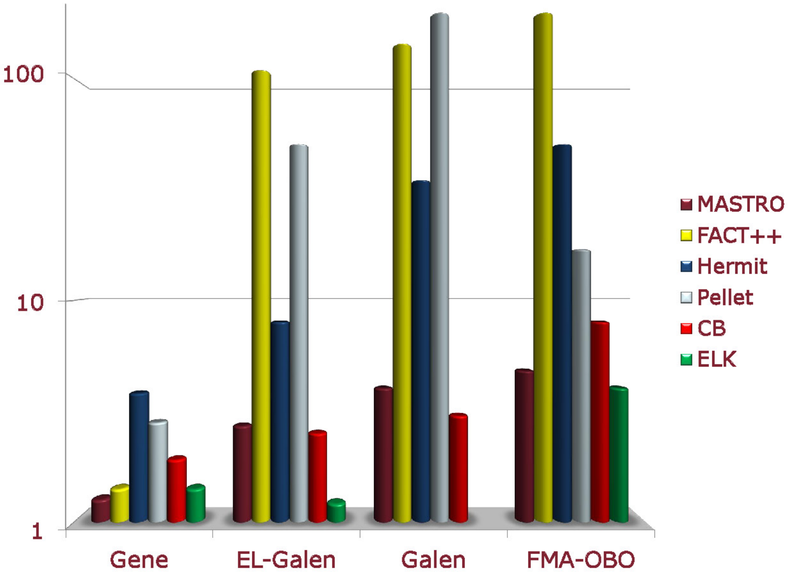
By exploiting these theoretical results, we have developed a Java-based OWL 2 QL classification module for the MASTRO reasoner for Ontology-Based Data Access (OBDA). In this implementation, the transitive closure of the digraph $\mathcal{G}_\mathcal{T}$ is based on a breadth first search through $\mathcal{G}_\mathcal{T}$.

We have performed comparative experiments on a suite of 20 ontologies, testing MASTRO against several popular ontology reasoners:

- the **FaCT++**, **Hermit**, **Pellet** OWL 2 reasoners
- the **CB** Horn-$\mathcal{SHIQ}$ reasoner
- the **ELK** OWL 2 EL reasoner

Each benchmark ontology was preprocessed through an approximation procedure prior to classification in order to fit OWL 2 QL expressivity.

Classification test results (seconds)



Conclusions and future work

We have presented a technique for efficiently computing classification of OWL 2 QL ontologies, based on the idea of encoding the ontology TBox into a directed graph, and reducing core reasoning to computation of the transitive closure of the graph.

Even though the current implementation relies on a naive algorithm for computation of transitive closure, test results on benchmark ontologies offer promising results.

Future Work:

- development of more efficient technique for transitive closure
- optimization of procedure for identification of unsatisfiable predicates
- extension of technique to computation of all inclusions inferred by the TBox
- extention of graph-based classification to more expressive languages

Thank you!

References I

[Sirin & al 07] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz.
Pellet: A practical OWL-DL reasoner.
J. of Web Semantics, 5(2):51-53, 2007.

[Glimm & al 12] B. Glimm, I. Horrocks, B. Motik, R. Shearer, and G. Stoilos.
A novel approach to ontology classification.
J. of Web Semantics, 14:84-101, 2012.

[Tsarkov & al 06] D. Tsarkov and I. Horrocks.
Fact++ description logic reasoner: System description.
In *Proc. of IJCAR 2006*, pages 292–297, 2006.

[Haarslev & Möller 01] V. Haarslev and R. Möller.
RACER system description.
In *Proc. of IJCAR 2001*, pages 701-706, 2001.

[Kazakov & al 11] Y. Kazakov and M. Krötzsch and F. Simančík.
Concurrent Classification of $\mathcal{EL}$ Ontologies.
In *Proc. of ISWC 2011*, pages 305-320, 2011.

[Lawley & Bousquet 10] M. Lawley and C. Bousquet.
Fast classification in Protègè: Snorocket as an OWL 2 EL reasoner.
In *Proc. of AOW 2010*, pages 45-50, 2011.

[Mendez & al 11] J. Mendez, A. Ecke, and A. Turhan.
Implementing completion-based inferences for the $\mathcal{EL}$-family.
In *Proc. of DL 2011*, 2011.

[Kazakov 09] Y. Kazakov.
Consequence-driven reasoning for horn SHIQ ontologies.
In *Proc. of IJCAI 2009*, pages 2040-2045, 2009.

[Civili & al 13] C. Civili, M. Console, D. Lembo, L. Lepore, R. Mancini, A. Poggi, M. Ruzzi, V. Santarelli, and D. F. Savo.
Mastro Studio: a system for Ontology-Based Data Management.
In *Proc. of OWLED 2013*, (to appear).

[Calvanese & al 11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D.F. Savo.
The Mastro system for ontology-based data access.
Semantic Web, 2(1):43–53, 2011.

[Motik & al 11] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz.
OWL 2 Web Ontology Language - Profiles (2nd edition).
W3C Recommendation, World Wide Web Consortium, Dec. 2012.