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

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.
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 JCell 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.
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
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
Preliminaries: OWL 2 QL

OWL 2 QL is the “data oriented” profile of OWL 2.

Expressions in OWL 2 QL

\[ B \rightarrow A \mid \exists Q \]
\[ C \rightarrow B \mid \neg B \mid \exists Q \cdot A \]
\[ Q \rightarrow P \mid P^- \]
\[ R \rightarrow Q \mid \neg Q \]

Assertions in OWL 2 QL

\[ B \sqsubseteq C \quad \text{(concept inclusion)} \]
\[ Q \sqsubseteq R \quad \text{(role inclusion)} \]

We call positive inclusions axioms of the form \( B_1 \sqsubseteq B_2 \), \( B_1 \sqsubseteq \exists Q \cdot A \), 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
Computation of $\Phi_T$

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

Definition

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

1. for each atomic concept $A$ in $\Sigma_P$, $\mathcal{N}$ contains the node $A$;
2. for each atomic role $P$ in $\Sigma_P$, $\mathcal{N}$ 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 $(\exists Q.A, \exists Q)$;
Compute the transitive closure of $\mathcal{G}_T$: $\mathcal{G}^* = (\mathcal{N}, \mathcal{E}^*)$

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 $\mathcal{G}_T = (\mathcal{N}, \mathcal{E})$ 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 $\Phi_T$ 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 Compute$\Phi$, that takes as input an OWL 2 QL TBox $\mathcal{T}$, builds $\mathcal{G}_T$, computes $\mathcal{G}^*$, and returns the set $\Phi_T$. 

Computation of $\Phi_T$: 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)

(Role inclusion)

Graph-based Ontology Classification in OWL 2 QL

28/05/2013
Computation of $\Phi_T$: 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)  
role inclusion

Example

Graph-based Ontology Classification in OWL 2 QL
Computation of $\Omega_T$: algorithm computeUnsat

Algorithm: computeUnsat
Input: an OWL 2 QL TBox $T$
Output: a set of concept and role expressions
Emp $\leftarrow \emptyset$;

foreach negative inclusion $S_1 \subseteq \neg S_2 \in T$ do
    Emp $\leftarrow$ Emp $\cup \{\text{predecessors}(S_1, G_T^*) \cap \text{predecessors}(S_2, G_T^*)\}$  /* step 1 */
    foreach $n_1 \in \text{predecessors}(S_1, G_T^*)$ do /* step 2 */
        foreach $n_2 \in \text{predecessors}(S_2, G_T^*)$ do
            if $(n_1 = \exists Q^- \text{ and } n_2 = A)$ or $(n_2 = \exists Q^- \text{ and } n_1 = A)$
                then Emp $\leftarrow$ Emp $\cup \{\exists Q.A\}$;

Emp$'$ $\leftarrow \emptyset$;
while Emp $\neq$ Emp$'$ do
    Emp$'$ $\leftarrow$ Emp;
    foreach $S \in$ Emp$'$ do
        foreach $n \in \text{predecessors}(S, G_T^*)$ do /* step 3 */
            Emp $\leftarrow$ Emp $\cup \{n\}$;
            if $n = P$ or $n = P^-$ or $n = \exists P$ or $n = \exists P^-$ /* step 4 */
                then Emp $\leftarrow$ Emp $\cup \{P, P^-, \exists P, \exists P^-\}$;
            if there exists $B \subseteq \exists Q.n \in T$
                then Emp $\leftarrow$ Emp $\cup \{\exists Q.n\}$;

return Emp.

- The set $\text{predecessors}(n, G^*)$ contains $n$ and all $n'$ s.t. $G^*$ contains $(n', n)$.
For each \( S_1 \sqsubseteq \neg S_2 \), computes \( \text{predecessors}(S_1, \mathcal{G}_T^*) \) and \( \text{predecessors}(S_2, \mathcal{G}_T^*) \):

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

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

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

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

(Step 4)

1. if at least one of the expressions \( P, P^-, \exists P, \exists P^- \) is in \( \text{Emp} \), then all four expressions are in \( \text{Emp} \);

2. for each expression \( \exists Q.A \) in \( \mathcal{N} \), if \( A \in \text{Emp} \), then \( \exists Q.A \in \text{Emp} \).
Computation of $\Omega_T$: 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:**

$\text{predecessors}(A_1, G^*_T) = \{A_1, A_3, \exists P_1\}$

$\text{predecessors}(A_2, G^*_T) = \{A_2, A_4, A_3, \exists P_1\}$
Example

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

\( \text{Emp} = \{ A_3, \exists P_1 \} \)
Example

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

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

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

Emp = \{A_3, \exists P_1, P_1, P_1^-, \exists P_1^-, \exists P_2 \cdot A_3, A_5\}
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 $\Sigma_P$. $\mathcal{T} \models S \sqsubseteq \neg S$ if and only if $S \in \text{computeUnsat}(\mathcal{T})$. 
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 $S_1 \sqsubseteq S_2 \in \text{Compute}\Phi(\mathcal{T}) \cup \text{Compute}\Omega(\mathcal{T})$. 
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 $G_T$ is based on a breadth first search through $G_T$.

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-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.
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
- extension of graph-based classification to more expressive languages
Thank you!
   Pellet: A practical OWL-DL reasoner.

   A novel approach to ontology classification.

   Fact++ description logic reasoner: System description.

[Haarslev & Möller 01] V. Haarslev and R. Möller.
   RACER system description.

   Concurrent Classification of $\mathcal{EL}$ Ontologies.

   Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner.
Implementing completion-based inferences for the $\mathcal{EL}$-family.

Consequence-driven reasoning for horn SHIQ ontologies.

Mastro Studio: a system for Ontology-Based Data Management.

The Mastro system for ontology-based data access.

OWL 2 Web Ontology Language - Profiles (2nd edition).