J Intell Inf Syst (2016) 47:447–467
DOI 10.1007/s10844-015-0374-3

Measuring similarity of individuals in description logics over the refinement space of conjunctive queries

Antonio A. Sánchez-Ruiz[1] · Santiago Ontañón[2] · Pedro A. González-Calero[1] · Enric Plaza[3]

Received: 17 June 2015 / Accepted: 15 July 2015 /
Published online: 2 August 2015

Abstract Similarity assessment is a key operation in several areas of artificial intelligence. This paper focuses on measuring similarity in the context of Description Logics (DL), and specifically on similarity between individuals. The main contribution of this paper is a novel approach based on measuring similarity in the space of *Conjunctive Queries*, rather than in the space of concepts. The advantage of this approach is two fold. On the one hand, it is independent of the underlying DL and therefore there is no need to design similarity measures for different DL, and, on the other hand, the approach is computationally more efficient than searching in the space of concepts.

Keywords Similarity assessment · Description logics · Conjunctive queries

✉ Antonio A. Sánchez-Ruiz
antsanch@ucm.es

Santiago Ontañón
santi@cs.drexel.edu

Pedro A. González-Calero
pedro@sip.ucm.es

Enric Plaza
enric@iiia.csic.es

[1] Departamento Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, Madrid, Spain

[2] Computer Science Department, Drexel University, Philadelphia, PA, 19104 USA

[3] IIIA-CSIC, Artificial Intelligence Research Institute, Campus Campus de la Universitat Autònoma de Barcelona, 08193 Bellaterra, Catalonia, Barcelona, Spain

1 Introduction

Description Logics (DL) (Baader et al. 2003) are one of the most widespread standards for knowledge representation in many application areas. Gaining momentum through the Semantic Web initiative, DL popularity is also related to a number of tools for knowledge acquisition and representation, as well as inference engines, that have been made publicly available.

In this paper, we focus on the problem of similarity assessment when the domain knowledge is represented using an ontology and formalized using DL. Specifically, we focus on the problem of measuring similarity between domain entities represented as individuals. An important aspect of the similarity measure presented in this paper is that, in addition to a numerical similarity value, it also provides an explicit description of the information that both individuals share as well as the information that only appears in one of them but not in the other. Therefore, our approach provides some explanation capabilities that may result useful in a number of different applications such as, for example, recommender systems.

The technique presented in this paper was initially developed as part of our research on knowledge-intensive case-based reasoning (CBR) systems (Aamodt and Plaza 1994) in which DL has also become the technology of choice for representing knowledge (Sánchez-Ruiz-Granados et al. 2009; Cojan and Lieber 2010; González-Calero et al. 1999). CBR is a problem solving paradigm that do not rely solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. Therefore, similarity assessment plays a key role in these type of systems. The technique presented in this paper, however, is not only interesting to CBR systems, it is of general application to anyone interested in measure the similarity of individuals in an expressive DL ontology.

Our similarity measure S_Q, works as follows: 1) given two individuals, we convert them into *DL Conjunctive Queries*, 2) the similarity between the two queries is measured using a refinement-operator-based similarity measure (Ontanón and Plaza 2012; Sánchez-Ruiz-Granados et al. 2011). There are four main advantages in the S_Q similarity approach with respect to previous approaches to the problem of similarity assessment in DL defined in the space of DL concepts (d'Amato et al. 2008; Sánchez-Ruiz-Granados et al. 2011): 1) the conversion process from individuals to queries does not lose information (the conversion to concepts usually causes some loss of information), 2) the language used to represent conjunctive queries is independent of the particular DL being used (and thus our approach can be applied to any DL), 3) assessing similarity in the space of queries is computationally more efficient than assessing similarity in the space of concepts, and 4) the aforementioned capability of providing explanations about the similarity both in terms of shared and specific information of the individuals being compared.

The rest of this paper is organized as follows. Sections 2 and 3 introduce the necessary concepts of Description Logics, Conjunctive Queries and Refinement Operators. Then, in Section 4, we introduce a new refinement operator for Conjunctive Queries. Section 5 presents the S_Q similarity measure between individuals, which is illustrated with an example in Section 5.3. In Section 6 we describe the implementation techniques that support the efficient implementation of the proposed approach. Section 7 presents an experimental evaluation of our approach. The paper closes with related work, conclusions and directions for future research.

Table 1 $\mathcal{EL}$ concepts and semantics

Concept	Syntax	Semantics
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Atomic concept	A	$A^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

2 Background

Description Logics (Baader et al. 2003) (DL) are a family of knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way.

DL represent knowledge using three types of basic entities: *concepts*, *roles* and *individuals*. Concepts provide the domain vocabulary required to describe sets of individuals with common features, roles describe relationships between individuals, and individuals represent concrete domain entities. DL expressions are built inductively starting from finite and disjoint sets of atomic concepts (N_C), roles (N_R) and individual names (N_I).

The expressivity and the reasoning complexity of a particular DL depends on the available constructors in the language. Although the similarity measure proposed in this work is independent of the description logic being used (the only effect being computation time), in this paper we will focus on the $\mathcal{EL}$ logic, a light-weight DL with good computational properties that serves as a basis for the OWL 2 EL profile.[1] $\mathcal{EL}$ is expressive enough to describe large biomedical ontologies, like SNOMED CT (Bodenreider et al. 2007) or the Gene Ontology (Ashburner 2000), while maintaining interesting computational properties such as concept subsumption being polynomial. The $\mathcal{EL}$ concept syntax and semantics is shown in Table 1. $\mathcal{EL}$ includes a top concept (the most general concept), intersections and existential restrictions.

A DL knowledge base (KB), $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, consists of two different types of information: $\mathcal{T}$, the *TBox* or terminological component, which contains concept and role axioms and describes the domain vocabulary; and $\mathcal{A}$, the *ABox* or assertional component, which uses the domain vocabulary to assert facts about individuals. For the purposes of this paper, a TBox is a finite set of concept and role axioms of the type given in Table 2, and an ABox is a finite set of axioms about individuals of the type shown in Table 3.

Regarding semantics, an interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the *interpretation domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function*. The interpretation function relates each atomic concept $A \in N_C$ with a subset of $\Delta^{\mathcal{I}}$, each atomic role $R \in N_R$ with a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each individual $a \in N_I$ with a single element of $\Delta^{\mathcal{I}}$. The interpretation function is extended to general concepts as shown in Table 1.

An interpretation $\mathcal{I}$ is a *model* of a knowledge base $\mathcal{K}$ iff the conditions described in Tables 2 and 3 are fulfilled for every axiom in $\mathcal{K}$. A concept C is *satisfiable* w.r.t. a knowledge base $\mathcal{K}$ iff there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$.

The basic reasoning operation in DL is *subsumption*, that induces a subconcept-superconcept hierarchy. We say that the concept C is subsumed by the concept D (C is more specific than D) if all the instances of C are also instances of D. Formally, C is subsumed

[1] http://www.w3.org/TR/owl2-profiles/

Table 2 TBox axioms

Axiom	Syntax	Semantics
Concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Disjointness	$C \sqcap D \equiv \bot$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
Role domain	$domain(R) = A$	$(x, y) \in R^{\mathcal{I}} \rightarrow x \in A^{\mathcal{I}}$
Role range	$range(R) = A$	$(x, y) \in R^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}$

by D w.r.t. the knowledge base $\mathcal{K}$ ($C \sqsubseteq_{\mathcal{K}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. When the knowledge base $\mathcal{K}$ is known we can simplify the notation and write $C \sqsubseteq D$. Finally, an *equivalence axiom* $C \equiv D$ is just an abbreviation for when both $C \sqsubseteq D$ and $D \sqsubseteq C$ hold, and a *strict subsumption axiom* $C \sqsubset D$ simply means that $C \sqsubseteq D$ and $C \not\equiv D$.

Figure 1 shows an example knowledge base that we use in this paper for exemplification purposes. The TBox contains axioms to define some vocabulary about pizzas and ingredients: $Mozzarella$ is a type of $Cheese$; $Margherita$ is a type of $Pizza$ with $Tomato$ and $Mozzarella$; a $CheesyPizza$ is any $Pizza$ with $Cheese$, etc. On the right side of the figure, there is a graphical representation of the concept hierarchy induced by the subsumption relation among those concepts. The ABox, in turn, contains axioms to describe two individuals: a margherita pizza and a pizza with chicken and vegetable toppings.

2.1 DL conjunctive queries

DL knowledge bases can be queried in order to retrieve individuals that meet certain conditions – in a way similar to the way queries are used to retrieve data from databases. In order to define queries, along with the set of atomic concepts (N_C), atomic roles (N_R) and individual names (N_I) from knowledge bases, we also need a disjoint set of variable names (N_V).

Definition 1 (Conjunctive Query)
A DL *conjunctive query* $Q(\vec{x}, \vec{y})$ is a logic formula $\exists \vec{y}.\psi(\vec{x}, \vec{y})$ where ψ is conjunction of terms of the form $A(x)$, $R(x, y)$, $x = y$ and $x \neq y$, in which $A \in N_C$ is an atomic concept, $R \in N_R$ is a role, and x and y are either individual names from N_I or variable names taken from the disjoint sets $\vec{x}, \vec{y} \subset N_V$.

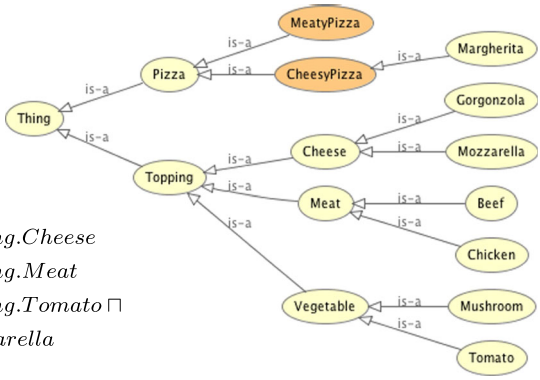
The sets $\vec{x}$ and $\vec{y}$ contain, respectively, all the *answer variables* and *quantified variables* of the query. A boolean conjunctive query $Q(\emptyset, \vec{y})$, or just $Q(\vec{y})$, is a query in which all the variables are quantified.

Table 3 ABox axioms

Axiom	Syntax	Semantics
Concept instance	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role assertion	$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
Same individual	$a = b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
Different individual	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

TBox

$$
\begin{aligned}
Topping &\sqsubseteq \top \\
Cheese &\sqsubseteq Topping \\
Mozzarella &\sqsubseteq Cheese \\
&\dots \\
Pizza &\sqsubseteq \top \\
CheesyPizza &\equiv Pizza \sqcap \exists hasTopping.Cheese \\
MeatyPizza &\equiv Pizza \sqcap \exists hasTopping.Meat \\
Margherita &\sqsubseteq Pizza \sqcap \exists hasTopping.Tomato \sqcap \\
&\quad \exists hasTopping.Mozzarella \\
range(hasTopping) &\sqsubseteq Topping
\end{aligned}
$$



ABox

$Margherita(p1), Pizza(p2), hasTopping(p2, t1), Chicken(t1), hasTopping(p2, t2), Vegetable(t2)$

Fig. 1 Example of knowledge base describing different types of pizzas and toppings

For example, next we show two possible conjunctive queries related to the pizza knowledge base:

$$
\begin{aligned}
Q_1(\{x_1\}, \{\}) &= Pizza(x_1) \\
Q_2(\{x_1\}, \{y_1\}) &= Pizza(x_1) \wedge hasTopping(x_1, y_1) \wedge Tomato(y_1)
\end{aligned}
$$

To define the semantics of general DL queries, let us begin considering only boolean queries. Let $VI(Q)$ be the set of variables and individuals in the query Q. An interpretation $\mathcal{I}$ is a model of a boolean query $Q(\vec{y})$, noted as $\mathcal{I} \models \exists \vec{y} : Q(\vec{y})$ or shortly as $\mathcal{I} \models Q$, if there is a variable substitution $\theta : VI(Q) \rightarrow \Delta^{\mathcal{I}}$ such that $\theta(a) = a^{\mathcal{I}}$ for each individual $a \in VI(Q)$, and $\mathcal{I} \models \alpha\theta$ for each term α in the query. The notation $\alpha\theta$ denotes the query atom α where the variables of α are substituted according to θ. A knowledge base $\mathcal{K}$ entails a boolean query Q, noted as $\mathcal{K} \models Q$, if every model of $\mathcal{K}$ satisfies Q.

Now let us consider queries with answer variables.

Definition 2 (**Query Answer**) An *answer* to a query $Q(\vec{x}, \vec{y})$ w.r.t. a knowledge base $\mathcal{K}$ is a variable substitution θ that maps the answer variables in $\vec{x}$ to individuals in $\mathcal{K}$ such that the boolean query $Q(\vec{x}\theta, \vec{y})$ is entailed by $\mathcal{K}$ as defined above.

The notation $Q(\mathbf{x}\theta, \mathbf{y})$ represents the query where all the distinguished variables have been replaced according to θ. Note that, for interpreting boolean queries, we use a substitution that maps quantified variables to arbitrary elements of the domain $\Delta^{\mathcal{I}}$, whereas for a query answer we require the answer variables to be mapped to named individuals in the ABox. In other words, while answer variables have to be mapped to individuals from the knowledge base signature, quantified variables can also be mapped to anonymous individuals that do not appear in the knowledge base signature but whose existence can be inferred.

Definition 3 (**Query Answer Set**) The *answer set* of a query $Q(\vec{x}, \vec{y})$ w.r.t. $\mathcal{K}$, noted as $Q(\mathcal{K})$, is the set containing all the answers to the query $Q(\vec{x}, \vec{y})$ w.r.t. $\mathcal{K}$.

For example, given the knowledge base in Fig. 1, the previous query Q_1 will retrieve all the existing pizzas ($Q_1(\mathcal{K}) = \{\{p1/x_1\}, \{p2/x_1\}\}$), while the query Q_2 will retrieve only those pizzas with tomato ($Q_2(\mathcal{K}) = \{\{p1/x1\}\}$). Notice that the reasoner infers that $p1$ has tomato because it is a margherita pizza and, therefore, the quantified variable y_1 in the query is mapped to an anonymous individual that does not exist in the ABox signature.

2.2 Query subsumption

We can define a subsumption relation between queries similar to the subsumption relation between concepts. In this way, queries can be organized into a hierarchy where the most general queries are above the most specific ones.

Definition 4 (Query Subsumption)
A query $Q(\mathbf{x}, \mathbf{y})$ is subsumed by another query $Q'(\mathbf{x}, \mathbf{y}')$ w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (noted as $\mathcal{K} \models Q \sqsubseteq Q'$) if, for every possible ABox $\mathcal{A}'$ and the knowledge base $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$, it holds that $Q(\mathcal{K}') \subseteq Q'(\mathcal{K}')$ (i.e. that the answer set of Q is contained in the answer set of Q').

Query containment is very closely related to query answering. The standard technique of *query freezing* (Ullman 2000) can be used to reduce query containment to query answering in DL (Motik 2006). To decide query subsumption, we build a canonical ABox $\mathcal{A}_Q$ from the query $Q(\mathbf{x}, \mathbf{y})$ by replacing each of the variables in $\mathbf{x}$ and $\mathbf{y}$ with fresh individual names not appearing in the KB. Let θ be the substitution denoting the mapping of variables $\mathbf{x}$ to the fresh individuals. Then, for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, $\mathcal{K} \models Q \sqsubseteq Q'$ iff θ is in the answer set of Q' w.r.t. to $\mathcal{K}_Q = (\mathcal{T}, \mathcal{A}_Q)$.

For example, considering the knowledge base about pizzas, query Q_3 below subsumes Q_4 because any margherita pizza is also a pizza with tomato and thus any answer to Q_4 is also an answer to Q_3.

$$Q_3(\{x_1\}, \{y_1\}) = Pizza(x_1) \wedge hasTopping(x_1, y_1) \wedge Tomato(y_1)$$
$$Q_4(\{x_1\}) = Margherita(x_1)$$

Note that Definition 4 assumes that both Q and Q' share the same set of answering variables. In the general case, we say that the query $Q(\mathbf{x}, \mathbf{y})$ is subsumed by query $Q'(\mathbf{x}', \mathbf{y}')$ if there is a containment mapping $\theta : \mathbf{x}' \rightarrow \mathbf{x} \cup N_I$ such that $Q(\mathbf{x}, \mathbf{y})$ is subsumed by $Q'(\mathbf{x}'\theta, \mathbf{y}')$.

3 Refinement operators

This section briefly summarizes the notion of *refinement operator* and the concepts relevant for this paper (see van der Laag and Nienhuys-Cheng 1998 for a more in-depth analysis of refinement operators). Refinement operators are defined over quasi-ordered sets.

Definition 5 A *quasi-ordered set* is a pair $(S, \leq)$, where S is a set, and $\leq$ is a binary relation among elements of S that is *reflexive* ($a \leq a$) and *transitive* (if $a \leq b$ and $b \leq c$ then $a \leq c$).

If $a \leq b$ and $b \leq a$, we say that $a \approx b$, or that they are *equivalent*. Refinement operators are defined as follows:

Definition 6 A *downward refinement operator* ρ over a quasi-ordered set $(S, \preceq)$ is a function such that $\forall a \in S : \rho(a) \subseteq \{b \in S | b \preceq a\}$.

Definition 7 An *upward refinement operator* γ over a quasi-ordered set $(S, \preceq)$ is a function such that $\forall a \in S : \gamma(a) \subseteq \{b \in S | a \preceq b\}$.

In other words, upward refinement operators generate elements of S which are "bigger" (which in this paper will mean "more general"), whereas downward refinement operators generate elements of S which are "smaller" (which in this paper will mean "more specific"). Typically, the symbol γ is used to symbolize upward refinement operators, and ρ to symbolize either a downward refinement operator, or a refinement operator in general. A common use of refinement operators is for navigating sets in an orderly way, given a starting element. Typically, the following properties of operators are considered desirable:

- A refinement operator ρ is *locally finite* if $\forall a \in S : \rho(a)$ is finite.
- A downward refinement operator ρ is *complete* if $\forall a, b \in S | a \preceq b : a \in \rho^*(b)$.
- An upward refinement operator γ is *complete* if $\forall a, b \in S | a \preceq b : b \in \gamma^*(a)$.
- A refinement operator ρ is *proper* if $\forall a, b \in S\ b \in \rho(a) \Rightarrow a \not\approx b$.

where ρ^* means the *transitive closure* of a refinement operator. Intuitively, *local finiteness* means that the refinement operator is computable, *completeness* means we can generate, by refinement of a, any element of S related to a given element a by the order relation $\preceq$ (except maybe those which are equivalent to a), and *properness* means that a refinement operator does not generate elements which are equivalent to a given element a.

Regarding DL queries, the set of DL conjunctive queries and the subsumption relation between queries (Definition 4) form a quasi-ordered set based upon their subsumption relation. Next, we describe a downward refinement operator for DL conjunctive queries, which will allow us to define the similarity measure presented in this paper.

4 A downward refinement operator for DL conjunctive queries

The following rewriting rules define an downward refinement operator for DL Conjunctive Queries. A rewriting rule is composed of three parts: the applicability conditions of the rewriting rule (shown between square brackets), the original DL query (above the line), and the refined DL query (below the line).

(R1) Concept Specialization

$$\left[x_1 \in VI(Q) \wedge A_2(x_1) \notin Q \wedge A_2 \sqsubset A_1 \wedge \nexists A' \in N_C : A_2 \sqsubset A' \sqsubset A_1\right]$$

$$\frac{Q(\vec{x}, \vec{y}) = A_1(x_1) \wedge \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}, \vec{y}) = A_1(x_1) \wedge A_2(x_1) \wedge \alpha_1 \wedge \ldots \wedge \alpha_n}$$

(R2) Concept Introduction

$$\left[x_1 \in VI(Q) \wedge A_1 \in max\{A \in N_C \mid \forall A'(x_1) \in Q : A \not\sqsubseteq A' \wedge A' \not\sqsubseteq A\}\right]$$

$$\frac{Q(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}, \vec{y}) = A_1(x_1) \wedge \alpha_1 \wedge \ldots \wedge \alpha_n}$$

(R3) Role Introduction

$$\left[x_1, x_2 \in VI(Q) \wedge R \in N_R \wedge R'(x_1, x_2) \notin Q\right]$$

$$\frac{Q(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}, \vec{y}) = R_1(x_1, x_2) \wedge \alpha_1 \wedge \ldots \wedge \alpha_n}$$

(R4) Variable Instantiation

$$[\theta : V(Q) \rightarrow N_I]$$

$$\frac{Q(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}\theta, \vec{y}\theta) = \alpha_1\theta \wedge \ldots \wedge \alpha_n\theta}$$

(R5) Quantified Variable Introduction

$$[x_2 \in N_V \setminus V(Q)]$$

$$\frac{Q(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}, \vec{y} \cup \{x_2\}) = \alpha_1 \wedge \ldots \wedge \alpha_n \wedge \top(x_2)}$$

(R6) Equality introduction

$$[x_1, x_2 \in VI(Q) \wedge x_1 = x_2 \notin Q]$$

$$\frac{Q(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n \wedge x_1 = x_2}$$

(R7) Inequality introduction

$$[x_1, x_2 \in VI(Q) \wedge x_1 \neq x_2 \notin Q]$$

$$\frac{Q(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n}{Q'(\vec{x}, \vec{y}) = \alpha_1 \wedge \ldots \wedge \alpha_n \wedge x_1 \neq x_2}$$

Rules R1 and R2 refine a query either specializing an existing atomic concept or introducing a new atomic concept that is neither more general nor more specific than the existing ones (for example a sibling in the concept hierarchy). Rule R3 introduces new role assertions between pairs of variables or individuals in the query. Note that we do not provide a rule to specialize role assertions since the $\mathcal{EL}$ logic does not allow role hierarchies. Rule R4 replaces variables in the query by named individuals in the knowledge base. Rule R5 introduces a new quantified variable to the query. And finally, rules R6 and R7 add equality and inequality axioms between variables or individuals in the query.

For the sake of space we do not provide proofs, but it is easy to verify that given a finite TBox and a finite ABox, the previous refinement operator is locally finite, not proper and complete.

For the purposes of similarity assessment however, we will restrict ourselves to a simpler refinement operator, only using rules R1, R2, R3, R4 and R5, and we will assume that all the variables and individuals in the query are different from each other. Under these assumptions, the restricted refinement operator is still locally finite, not proper and complete.

These simplifications greatly reduce the computational complexity of the similarity measure presented in this paper. For example, these simplifications prevent infinite refinement chains where all the queries are equivalent, such as:

$$A(x_1) \rightarrow A(x_1) \wedge \top(y_1) \rightarrow A(x_1) \wedge \top(y_1) \wedge \top(y_2) \rightarrow \ldots$$

Dealing with these infinite chains using other approaches and exploring different refinement operators that offer a different trade-off of completeness and efficiency is part of our future work.

5 Similarity based on query refinements

This section presents a new similarity measure, S_Q, between individuals in the ABox. S_Q consists of two main steps (described in the following two subsections): First, individual names a and b are transformed into conjunctive DL queries, Q_a and Q_b. Second, the similarity value between Q_a and Q_b is determined using the refinement operator presented above.

5.1 From individuals to queries

Given an individual name a and an ABox $\mathcal{A}$, we define the *individual graph* of a as follows.

Definition 8 (**Individual graph**) An *individual graph* $G_a \subseteq \mathcal{A}$ is a set of ABox axioms with one distinguished individual $a \in N_I$ such that:

- $\forall C \in N_C$, if $C(a) \in \mathcal{A}$ then $C(a) \in G_a$
- $\forall C \in N_C$, if $C(b) \in G_a$ then $\forall R \in N_R$, if $R(b, c) \in \mathcal{A}$ then $R(b, c) \in G_a$
- $\forall R \in N_R$, if $R(b, c) \in G_a$ then $\forall D \in N_C$ if $D(c) \in \mathcal{A}$ then $D(c) \in G_a$

In other words, if we represent the ABox as a graph, where each individual is a node, and each role axiom is a directed edge, the individual graph of an individual name a would be the connected graph resulting from all the nodes and edges reachable from a.

We can transform an individual graph to an *equivalent* conjunctive query applying a substitution that replaces the distinguished individual by a new answer variable, and the remaining individuals by new quantified variables. Note that the conversion is straightforward since ABox axioms and DL query terms are alike and no information is lost in the translation.

The following example shows the individual graph related to individual $p2$ from the knowledge base in Fig. 1 and its equivalent conjunctive query. Notice that, for clarity, we take the liberty of representing the set of axioms of the individual graph as the conjunction of its elements.

$$\begin{aligned} G_{p2} &= Pizza(p2) \wedge hasTopping(p2, t1) \wedge Chicken(t1) \wedge \\ &\quad hasTopping(p2, t2) \wedge Vegetable(t2) \\ Q_{p2}(\{x_1\}, \{y_1, y_2\}) &= Pizza(x_1) \wedge hasTopping(x_1, y_1) \wedge Chicken(y_1) \wedge \\ &\quad hasTopping(x_1, y_2) \wedge Vegetable(y_2) \end{aligned}$$

5.2 Similarity between queries

Our proposed similarity measure for conjunctive DL queries is based on the following intuitions (see Fig. 2):

First, given two queries Q_1 and Q_2 such that $Q_2 \sqsubseteq Q_1$, it is possible to reach Q_2 from Q_1 by applying a complete downward refinement operator ρ to Q_1 a finite number of times, i.e. $Q_2 \in \rho^*(Q_1)$.

Second, the number of times a refinement operator needs to be applied to reach Q_2 from Q_1 is an indication of how much more specific Q_2 is than Q_1. The length of the chain of refinements to reach Q_2 from Q_1, which will be noted as $\lambda(Q_1 \xrightarrow{\rho} Q_2)$, is an indicator of how much information Q_2 contains that was not contained in Q_1. It is also an indication of their similarity: the smaller the length, the higher their similarity.

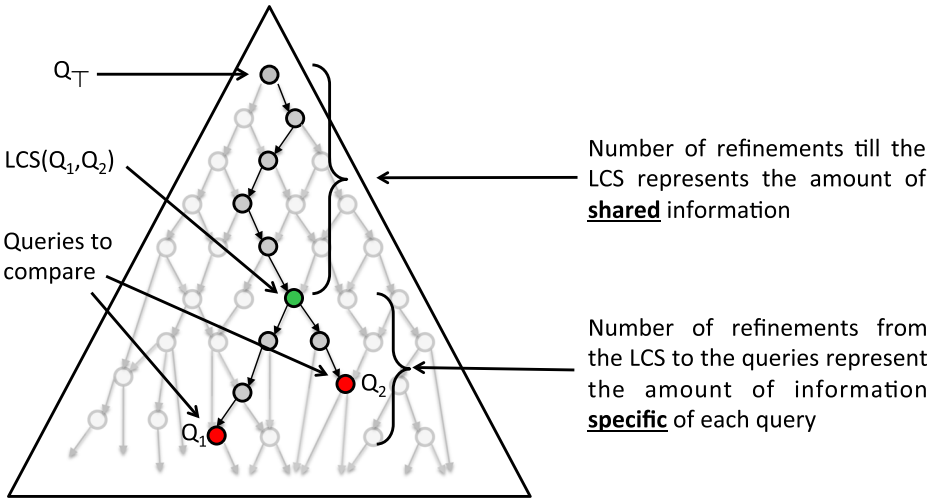


Fig. 2 Query distance based on refinements

Given the unfeasibility of computing the exact minimum length (due to combinatorial explosion), in our experiments we used a greedy search approach using the rewriting rules R1, R2, R3, R4 and R5 that were introduced in Section 4. This greedy approach does not ensure obtaining the shortest chain between to queries, but is computationally efficient.

Third, given any two queries, their *least common subsumer* (LCS) is the most specific query which subsumes both. Given two queries Q_1 and Q_2, the LCS can be computed by starting with the most general query $Q_\top$ and refining it with the refinement operator until no more refinements can be applied that result in a query that still subsumes Q_1 and Q_2.

The LCS of two queries contains all that is shared between two queries, and the more they share the more similar they are. $\lambda(Q_\top \xrightarrow{\rho} LCS)$ measures the distance from the most general query, $Q_\top$, to the LCS, and it is a measure of the amount of information shared by Q_1 and Q_2.

Finally, the similarity between two queries Q_1 and Q_2 can be measured as the ratio between the amount of information contained in their LCS and the total amount of information contained in Q_1 and Q_2. These ideas are collected in the following formula:

$$S_\rho(Q_1, Q_2) = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

where:

$$\begin{aligned}
\lambda_1 &= \lambda(Q_\top \xrightarrow{\rho} LCS(Q_1, Q_2)) \\
\lambda_2 &= \lambda(LCS(Q_1, Q_2) \xrightarrow{\rho} Q_1) \\
\lambda_3 &= \lambda(LCS(Q_1, Q_2) \xrightarrow{\rho} Q_2)
\end{aligned}$$

Thus, the similarity between two individuals a and b, is defined as:

$$S_Q(a, b) = S_\rho(Q_a, Q_b)$$

where Q_a and Q_b are the queries corresponding to the individual graphs of a and b, respectively.

Notice that the proposed similarity measure is conceptually related to the well known Jaccard index (Jaccard 1901) for sets. The Jaccard index is a well-known similarity measure between sets consisting on dividing the size of their intersection by the size of their union (when the sets are identical, this is 1, and when they are disjoint, this is 0). Notice that λ_1, in a way, measures the size of what is shared between two queries (the intersection), and $\lambda_1 + \lambda_2 + \lambda_3$ measures the total amount of information in the two queries (their union). Thus, one could see our proposed similarity measure as an adaptation of the Jaccard index to refinement graphs.

Additionally, notice that the length of the refinement paths highly depend on how fine-grained is the set of concepts in the ontology. For example, if the ontology contains a large set of intermediate concepts between a concept A_1 and another concept A_2, then the distance between an individual of type A_1 and A_2 will be high. However, this will be reversed if the ontology does not have that many intermediate concepts. This problem has been studied in the past in the context of conceptual hierarchies (Resnik 1995), and can be addressed by computing a weight for each refinement that is based on how much additional information a given refinement introduces. The amount of information can be estimated if a large enough collection of individuals is known by measuring how many individuals are not covered any more after each refinement. This approach can also be used to customize similarity assessment for specific domains, and we will explore it as part of our future work.

5.3 Example

Figure 3 shows the chain of refinements used to compute the similarity between a margherita pizza and a pizza with some vegetable topping and chicken. We begin with the most general query with one answer variable and we refine it until we reach the LCS, that describes the common part of those pizzas: both have at least two toppings and one of them is a vegetable. Next we refine the LCS until we reach each one of the original queries.

Their similarity is computed using the length of the refinement paths:

$$S_Q(p1, p2) = S_\rho(Q_{p1}, Q_{p2}) = \frac{8}{8+3+5} = 0.5$$

It is important to note that, besides the similarity value, we also obtain an explicit description of the information that both pizzas share (the LCS) and the information that is specific of each one of them. This information can be very valuable, for example, in systems that explain to the users the computed similarity value.

6 Implementation details

6.1 Using answer variables instead of quantified variables

One of the assumptions we make in order to simplify the search space is that all the variables in the queries represent different individuals. However, current DL reasoners do not support terms of the form $?y_1 \neq ?y_2$ in conjunctive queries when $?y_1$ and $?y_2$ are quantified variables. Note that this is a problem, since we can not simply replace all the quantified variables with answer variables because the first ones can be mapped to anonymous individuals while the second ones can only be mapped to named individuals in the ontology signature.

The workaround we used to this problem consists of two steps: (1) we pre-process the knowledge base and add new individuals to explicitly represent the anonymous individuals

Queries for G_{p1} and G_{p2}
$Q_{p1}(\{x_1\},\{\}) = Margherita(x_1)$
$Q_{p2}(\{x_2\},\{y_1,y_2\}) = Pizza(x_2) \wedge hasTopping(x_2,y_1) \wedge Chicken(y_1) \wedge hasTopping(x_2,y_2) \wedge Vegetable(y_2)$

Path from Q_T to $LCS(Q_{p1},Q_{p2})$
$1 : \top(x_3)$
$2 : Pizza(x_3)$
$3 : Pizza(x_3) \wedge \top(y_3)$
$4 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge \top(y_3)$
$5 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Topping(y_3)$
$6 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Vegetable(y_3)$
$7 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Vegetable(y_3) \wedge \top(y_4)$
$8 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Vegetable(y_3) \wedge hasTopping(x_3,y_4) \wedge \top(y_4)$
$9 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Vegetable(y_3) \wedge hasTopping(x_3,y_4) \wedge Topping(y_4)$

Path from $LCS(Q_{p1},Q_{p2})$ to Q_{p1}
$10 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Tomato(y_3) \wedge hasTopping(x_3,y_4) \wedge Topping(y_4)$
$11 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Tomato(y_3) \wedge hasTopping(x_3,y_4) \wedge Mozzarella(y_4)$
$12 : Pizza(p1) \wedge hasTopping(p1,y_3) \wedge Tomato(y_3) \wedge hasTopping(p1,y_4) \wedge Mozzarella(y_4)$

Path from $LCS(Q_{p1},Q_{p2})$ to Q_{p2}
$13 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Vegetable(y_3) \wedge hasTopping(x_3,y_4) \wedge Meat(y_4)$
$14 : Pizza(x_3) \wedge hasTopping(x_3,y_3) \wedge Vegetable(y_3) \wedge hasTopping(x_3,y_4) \wedge Chicken(y_4)$
$15 : Pizza(p2) \wedge hasTopping(p2,y_3) \wedge Vegetable(y_3) \wedge hasTopping(p2,y_4) \wedge Chicken(y_4)$
$16 : Pizza(p2) \wedge hasTopping(p2,t1) \wedge Vegetable(t1) \wedge hasTopping(p2,y_4) \wedge Chicken(y_4)$
$17 : Pizza(p2) \wedge hasTopping(p2,t1) \wedge Vegetable(t1) \wedge hasTopping(p2,t2) \wedge Chicken(t2)$

Fig. 3 Refinement paths to compute $S_p(Q_{p1}, Q_{p2})$

that do not exist in the knowledge base signature but whose existence can be inferred from the knowledge base axioms, and (2) we only use answer variables in the queries, which are now equivalent to quantified variables, since the knowledge base has been populated with all the possible anonymous individuals to which quantified variables can be assigned to. This process is analogous to the process known as *saturation* in Inductive Logic Programming (Rouveirol 1994).

In the $\mathcal{EL}$ logic, the only axioms that can be used to infer the existence of a new individual have the form $(\exists R.C)(a)$ meaning that individual a is related by role R to another individual with type C. If there is not another individual b in the KB signature such that $R(a,b)$ and $C(b)$ then we create a new individual sk and add the axioms $R(a,sk)$ and $C(sk)$ to the knowledge base. Figure 4 presents our algorithm.

```
Data: Knowledge Base KB
foreach individual a not processed yet do
    foreach class ∃R.C such that KB ⊨ (∃R.C)(a) do
        if there is not individual b such that KB ⊨ R(a,b) ∧ C(b) then
            sk ← new individual;
            KB ← KB ∪ {R(a,sk), C(sk)};
        end
    end
end
```

Fig. 4 Algorithm to explicitly represent anonymous individuals in the knowledge base

For example, given the following KB:

$$Margherita \sqsubseteq Pizza \sqcap \exists hasTopping.Tomato \sqcap \exists hasTopping.Mozzarella$$
$$Margherita(p1)$$

we can infer the existence of two anonymous individuals that our algorithm will replace with new named individuals $sk1$ and $sk2$:

$$Margherita(p1), hasTopping(p1, sk1), Tomato(sk1),$$
$$hasTopping(p2, sk2), Mozzarella(sk2)$$

6.2 Performance and redundant refinements

The similarity measure described so far relies on answering several queries using a knowledge base that remains unchanged during the entire process. Due to the static nature of the knowledge base, we can speed up the time required to answer the queries if we pre-compute all the relations between the individuals as well as the concepts they belong to.[2] In fact, if we only use answer variables in the queries, the search of the refinement paths can be performed without the need of a reasoner since all the inferences can be cached beforehand. In the presence of dynamic knowledge bases, however, we will have to rely on the reasoner to dynamically answer all the DL queries generated during the search in the space of refinements and the process will be slower. Anyway, the progress made in recent years regarding incremental reasoning in DL (Grau et al. 2010; Parsia et al. 2006; Kazakov and Klinov 2013) lead us to think that these times will improve in the near future.

When we look carefully at the refinement paths obtained with our refinement operator we can see that some of refinements do not add new information to the previous one. For example, the following 2 refinements are part of the similarity assessment shown in Fig. 3.

$$4 : Pizza(x_3) \wedge hasTopping(x_3, y_3) \wedge \top(y_3)$$
$$5 : Pizza(x_3) \wedge hasTopping(x_3, y_3) \wedge Topping(y_3)$$

Refinement 5 does not add new information that was not already in refinement 4 since the range of the role $hasTopping$ is the concept $Topping$. In other words, both queries are semantically equivalent. Since the refinement operator is not proper, we can not guarantee each refinement to be strictly more specific than the previous one, so the length of the refinement path is in fact an approximation to the semantic distance.

From a conceptual point of view, these *redundant* refinements should not be taken into account to compute the distance. In order to remove them, we have to traverse the sequence of refinements checking if each query subsumes, and thus is equivalent to, the previous one ($Q_{i+1} \sqsubseteq Q_i$).

$$Q_1 \xrightarrow{\rho} Q_2 \xrightarrow{\rho} Q_3 \xrightarrow{\rho} Q_4 \xrightarrow{\rho} \ldots$$

Unfortunately, the standard technique to decide query subsumption adds new axioms to the knowledge base (see paragraph about query freezing in Section 2.2) and, as we have already explained, the time required to answer queries increases significantly when we deal with dynamic knowledge bases.

[2]Notice that this approach require large amounts of memory and might no be practical for large knowledge bases.

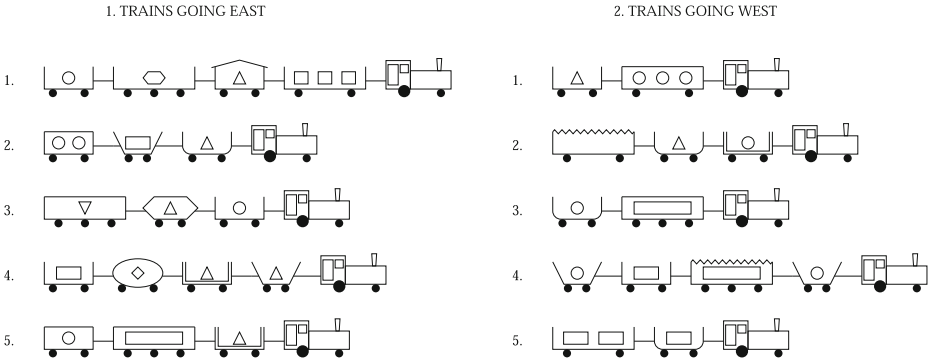


Fig. 5 Trains data set as introduced by Larson and Michalski (1977)

In our experiments, we did not observe significant differences in the results obtained removing the redundant refinements. Therefore, the extra computation time required to filter these refinements did not pay off. However, in some other domains, this might not be the case, and it might be worth spending additional computation time to remove redundant refinements.

7 Experiments

In order to evaluate the S_Q similarity measure, we used six relational machine learning data sets:

- *Trains*: shown in Fig. 5 as presented by Larson and Michalski (1977). Like in our previous work on similarity assessment (Sánchez-Ruiz-Granados et al. 2011), we selected this dataset since it is available in many representation formalisms (Horn clauses, feature terms and description logic), and therefore, we can compare our similarity measure with existing similarity measures in the literature. The dataset consists of 10 trains, 5 of them labelled as "West", and 5 of them labelled as "East."
- *Pairs*: this is a subset of the *poker hands* dataset from the UCI machine learning repository. It contains 50 poker hands, 20 of which have a "pair", and 30 of which do not. Each instance is just a set of five cards. The OWL version of this dataset was obtained from the DL-Learner project by Lehmann and Hitzler (2007).
- *Straight*: this is another subset of the *poker hands* dataset from the UCI machine learning repository. It contains 55 poker hands, four of which have a "straight" and 51 of which do not. Both the Pairs and Straight datasets contains rather simple example representation, but a complex target concept (any machine learning algorithm attempting to solve this task needs to learn the concept of a "pair" or a "straight" in Poker, which is not trivial). Each card in the hand is described by its rank and suit. Also, relations such as which rank is the successor of which other rank is stored (for example the "Queen" of "Spades" is related to the "King" of "Spades" with a "successor" relation, since the "King" is the next card after the "Queen"). The OWL version of this dataset was obtained from the DL-Learner project by Lehmann and Hitzler (2007).
- *Lymphography*: from the UCI machine learning repository. This dataset is propositional. The OWL version of this dataset was obtained from the DL-Learner project by Lehmann and Hitzler (2007).

– *Pizzas*: this dataset uses a simplified $\mathcal{EL}$ version of the ontology presented in Horridge (2009) and has been populated with 22 pizzas belonging to two different classes. We created this dataset specifically for evaluating one of the main advantages of our similarity measure with respect to other measures: some of the key aspects that need to be taken into account to correctly classify pizzas are not present explicitly in the instances, but need to be inferred. For example, a pizza might be just labeled as Margherita, from which we can infer that it has tomato and cheese. Examples used earlier in this paper are taken from this data set.

We compared our similarity measure against several others: $S_{DL\rho}$ (Sánchez-Ruiz-Granados et al. 2011), a similarity measure for the $\mathcal{EL}$ description logic based on refinements in the space of concepts; González-Calero et al. (1999), a similarity measure for acyclic concepts in description logic; RIBL (Emde and Wettschereck 1996), which is a Horn clause similarity measure; SHAUD (Armengol and Plaza 2003), which is a similarity measure for feature terms; and S_λ and S_π (Ontañón and Plaza 2009), which are similarity measures for feature terms but also based on the idea of refinement operators. For RIBL, we used we used the original version of the trains dataset, and automatically translated versions of the other datasets. For SHAUD, S_λ, and S_π, we used the feature term version of the trains dataset used in Ontanón and Plaza (2012), which is a direct conversion from the original Horn clause dataset without loss; the pairs, straight and pizza datasets were automatically translated to feature terms; the lymphography data set was translated to feature terms using the original propositional dataset as the starting point. For the DL similarity measures, we used the versions of these datasets created by Lehmann and Hitzler (2007).

We compared the similarity measures in different ways:

– Classification accuracy of a nearest-neighbor algorithm.
– *Average best rank* of the first correct example: if we take one of the instances, and sort the rest of the trains according to their similarity with the selected instance, which is the position in this list (rank) of the first instance with the same solution as the selected instance (e.g. West or East in the case of the trains dataset).
– Average time taken to compute similarity between two individuals.

Table 4 shows the results obtained when we compared the similarity measures in a nearest-neighbor algorithm. Not all similarity measures could be applied to all data sets. For example, $S_{DL\rho}$ has scalability issues, and would only complete execution in the trains dataset. González et al. cannot handle circular definitions, and thus could not be applied to Pairs and Straight.

Concerning classification accuracy, we can see that our new similarity measure, S_Q, achieves comparable results to other similarity measures in the literature. Specifically, it obtains the second highest accuracy for the trains data set, very close to the highest for the Lymphography data set, and the highest in the Pizzas data set. We would like to highlight the performance of S_Q in the pizzas data set. In fact, S_Q is the only similarity measure that takes into account the information that can be inferred from the ontology, which explains the results. RIBL, SHAUD, S_λ and S_π are at a great disadvantage in this dataset since some of the information that S_Q can infer form the ontology cannot be inferred in the same way in feature terms or Horn clauses (and those similarity measures do not have any mechanisms for inference anyway). Concerning average best rank, we see similar results as for accuracy. S_Q obtains results similar to other similarity measures, except in a few datasets. It obtains the best results in the pizzas dataset, and does not perform well in the Straight data set (where classification accuracy, however, was quite high).

Table 4 Comparison of several similarity metrics (times do not take into account data preprocessing, required for some of the similarity measures). The best results for each dataset are highlighted in bold

	S_Q	$S_{DL\rho}$	G. et al.	RIBL	SHAUD	S_λ	S_π
Classification Accuracy using 1-NN							
Trains	60.00	**70.00**	50.00	60.00	50.00	40.00	50.00
Pairs	79.59	–	–	63.27	71.43	83.67	**91.84**
Straight	87.27	–	–	**92.73**	80.00	89.09	83.64
Lymph.	89.19	–	**89.86**	78.38	50.00	79.05	79.05
Pizzas	**86.36**	–	27.27	70.24	70.24	70.24	70.24
Average Best Rank							
Trains	2.30	2.60	**1.50**	1.90	1.90	2.30	2.10
Pairs	1.53	–	–	1.63	1.98	1.18	**1.10**
Straight	1.96	–	–	1.29	1.36	1.73	**1.27**
Lymph.	2.47	–	2.36	1.45	2.16	1.49	**1.44**
Pizzas	**1.23**	–	1.77	2.04	2.04	2.04	2.04
Average Time (in milliseconds)							
Trains	15.69	4422.36	0.47	6.63	75.47	45.44	**0.12**
Pairs	189.42	–	–	1.09	33.67	22.97	**0.24**
Straight	207.14	–	–	0.86	23.70	19.81	**0.16**
Lymph.	1.20	–	**0.01**	0.18	0.99	1.77	0.04
Pizzas	271.98	–	**0.03**	0.24	0.83	1.28	0.11

It is important to note that S_Q does not only compute a similarity value between the two individuals being compared. As a result of the search process, it also produces an explicit description of the structure shared by the two individuals (the LCS) and the part that is specific of each one of them. We think that this information might be very useful in interactive systems to explain the similarity assessment to the user, and we plan to develop this idea as part of our future work.

Concerning only the three similarity measures we considered for DL, $S_{DL\rho}$ is much slower than S_Q, and González et al. is faster, but at the cost of not handling circular structures, nor fully exploiting information that can be inferred form the ontology (as can be see for the results in the pizzas dataset), and not providing an explanation of the similarity value. $S_{DL\rho}$ does provide an explanation of the similarity since it performs a search in the space of concepts, but it is much slower than S_Q. The space of queries is narrower than the space of concepts since queries can only contain atomic concepts and roles while general DL concepts can combine any of the constructors in the language. However, this limitation does not necessarily affect the quality of the similarity, since atomic concepts represent the vocabulary chosen by domain experts to describe domain entities, and therefore atomic concepts represent the most important conceptualizations in the domain. Also, many practical optimizations can be performed, such as sorting the query term in such a way that the most restrictive axioms ones are evaluated first.

In summary, S_Q is a new practical approach to assess similarity for expressive DL, with a similar classification accuracy than existing similarity measures for other representation formalisms, but more general than González et al.'s, and more efficient than our

previous measure $S_{DL\rho}$. In addition, S_Q computes an explicit representation of the information shared between the two entities that can be used for explanation, and it is able to deal with anonymous individuals whose existence can be inferred from the ontology axioms.

8 Related work

The work presented in this paper extends our previous work on similarity on Description Logics (Sánchez-Ruiz-Granados et al. 2011), where we studied how to assess similarity between individuals by transforming them to concepts, and then assessing the similarity of these concepts. The approach presented in this paper is more general (since the language DL queries is common to all DL), more efficient, and more accurate (since in our previous work, we might lose information when converting individuals to concepts).

Other approaches to similarity assessment for Description Logics exist. For example, d'Amato et al. (2008) propose to measure concept similarity as a function of the intersection of their interpretations, which is, in fact, an approximation to the semantic similarity of concepts. The approximation is better or worse depending on how good is the sample of individuals used for assessing similarity. Thus, a good sample of individuals is required. Other approaches that do not require the use of a good sample of individuals also have been proposed. González-Calero et al. (1999) present a similarity measure for description logic designed for case-based reasoning systems. This similarity measure is based on the idea of hierarchical aggregation, in which the similarity between two instances is computed as an aggregation of the similarity of the values in their roles.

There has been a significant amount of work on defining similarity measures for Horn Clauses, in the Inductive Logic Programming (ILP) community. Hutchinson (1997) presented a distance based on the least general generalization (lgg) of two clauses. The lgg of two clauses is a clause that subsumes both (i.e. that is more general than both), and such that there is no other, more specific, clause that also subsumes both. The Hutchinson distance is computed as the addition of the sizes of the variable substitutions required to transform the lgg into each of the two clauses. As has been pointed out in the literature (Ramon 2002; Ontanón and Plaza 2012), this is a very rough measure, which fails to take into account a significant amount of information. Also, notice that this measure is analogous to the Jaccard index (with the lgg playing the role of the intersection, and using the size of the variable substitutions as a measure of the difference in size between the intersection and the union), and is also related to the refinement-based approaches in Sánchez-Ruiz-Granados et al. (2011) and Ontañón and Plaza (2009), but is more coarse grained.

One of the most influential similarity measures for Horn Clauses is that in RIBL (Relational Instance-Based Learning) (Emde and Wettschereck 1996). RIBL's similarity measure follows a "hierarchical aggregation" approach: the similarity of two objects is a function of the similarity of their attributes (calling this function recursively if the values are themselves structured objects). However, RIBL presents a number of problems: first, it implicitly assumes that values that are "further away" from the root of an object will play a lesser role in similarity (assumption that doesn't hold in many problems); second, its hierarchical procedure makes it better suited for objects whose representation does not contain cycles. Finally, similarity measures have to be defined for different types of data inside of RIBL (e.g. for numerical values, categorical values, etc.). This last point is clearly illustrated in the work of Horváth et al. (2001), where the present an extension of RIBL able to deal with lists by incorporating an edit-distance whenever a list data type is reached. An earlier

similarity measure related to RIBL was that of Bisson (1992). Other similarity measures specifically defined for Horn Clauses include the work of Nienhuys-Cheng (1997), and of Ramon (2002).

Another related area is that of similarity in Feature Logics (Carpenter 1992), a.k.a. Feature Terms, a representation formalism that like Horn-Clauses and Description Logics is a subset of First-Order Logic. SHAUD, presented by Armengol and Plaza (2003), is a similarity measure also following the "hierarchical aggregation" approach but designed for the feature logic representation formalism. SHAUD also assumes that the terms do not have cycles, and in the same way as RIBL it can handle numerical values by using specialized similarity measures for different data types. Our previous work (Ontanón and Plaza 2012) addresses these problems, and presented a collection of general similarity measures for feature terms based on refinement operators.

Bergmann and Stahl (1998) present a similarity measure specific for object-oriented representations based on the concepts of intra-class similarity (measuring similarity among all the common features of two objects) and inter-class similarity (providing a maximum similarity given to object classes). This similarity is defined in a recursive way, thus following the same "hierarchical aggregation" idea as RIBL and SHAUD, limiting the approach to tree representations. These similarity measures are basically an attempt to generalize standard Euclidean or Manhattan distances to structured data.

Another related area is that of kernels for structured representations, which allow the application of machine learning techniques such as Support Vector Machines to complex data represented as graphs (such as chemical molecules). Typically, kernels for graphs are based on the idea of finding common substructures between two graphs (the idea is that if substructures are similar, then graphs are similar), which is akin to the similarity measures between sets, such as Jaccard's, which try to find common elements (the intersection) between two sets. For example, Kashima et al. (2003) present a kernel for graphs based on random-walks. Fanizzi et al. (2008) also studied how to encapsulate their similarity measure for Description Logics into a Kernel.

Similarity measures for complex molecular structures in domains of biology or chemistry have also been widely studied (Willett et al. 1998), and they are typically grouped into three classes (Raymond et al. 2003): sequence similarities, fingerprint-based and graph-based. Sequence similarities can be directly applied to molecules such DNA fragments, that can be directly represented as sequences (although they lose any of their structural properties, such as their 3D configuration). Fingerprint similarities transform each molecule in a sequence of binary features where each feature determines whether a particular molecule exhibits some particular property or contains a certain substructure. Graph-based similarity measures for molecules are typically based on computing the maximum common subgraph of two graphs, which is related to the idea of computing the least common subsumer in DLs. This is a computationally expensive process, and thus there are a number of strategies to simplify the computations. For instance, Raymond et al. (2002) propose a two-stage process using upper bounds to filter out the molecules for which we require exact computations.

Additionally, there are several ideas that are relevant for similarity assessment. For example, the idea of *propositionalization* (transforming a structured instance into a propositional instance) (Kramer et al. 2001) is very relevant for similarity assessment. Another related area is that of Formal Concept Analysis (FCA), where work on similarity assessment between concepts is starting to be studied (Formica 2006; Alqadah and Bhatnagar 2011).

9 Conclusions and future work

This paper has presented a new approach to assess similarity between individuals in Description Logics. Although developed as part of our research on knowledge-intensive CBR, the technique presented here is of general application to any problem that requires assessment of similarity between individuals expressed in DL, such as automated clustering, or information retrieval. In addition, this technique also provides an explicit description of the information that both individuals being compared share and the information that is specific of each one of them, thus providing an explanation capability that can be useful in a number of different applications.

The main advantage of our proposed similarity approach with respect to previous approaches is that the search is performed in the space of conjunctive queries instead of the space of concepts and, therefore, it is independent of the particular DL being used and computationally more efficient, as our experimental results show (see Table 4).

Concerning future work, we are interested in the idea of using the explanation capabilities of this similarity measure in systems that interact with humans and have to explain the computed solutions. There is a major challenge in representing intuitively the knowledge contained in a conjunctive query so that domain experts can understand the similarity values without having to be experts in logic.

Another interesting line of research is to formally study the computational complexity of the similarity measure for different refinement operators. For example, we want to study the trade-offs between relaxing the requirement of having a complete refinement operator (reducing the search space, and thus the computational complexity), the resulting computational complexity and the performance of the resulting similarity measure. Additionally, in the experiments we have only considered entities represented using individuals, but our work could be extended to consider another type of attributes such as numeric attributes, or dates.

Regarding CBR systems specifically, the refinement paths contain valuable information about case adaptation. Note that we can transform one individual into another simply traversing the refinement paths from one individual to the least-common subsumer and then down again until the other individual. If we use individuals to represent cases, we do not only know when two cases are similar, we also know what their differences are.

Finally, the notion of *similarity* is highly context-dependent: two pizzas can be considered similar or not depending on the problem at hand or the purpose of comparison. There are different ways to customize the similarity measure presented in this paper for a given domain. For example, we could populate the ontology with more concepts to increase the granularity of the refinements. Another approach would be to annotate the ontology with distances between concepts so that not all the refinements contribute the same. Alternatively, we could adapt the *disintegration* approach presented in Ontanón and Plaza (2012), where refinements were used to generate a collection of relational-patterns, and weights for each pattern could be calculated in a domain-specific way for similarity purposes. Finally, if we consider the query as a graph (similarly to the individual graph introduced in Section 5.1) we could weight the refinements according to the distances between the variables involved in the refinement and a distinguished variable in the query representing the main entity. Although all these approaches are feasible, determining specific weights and distances required to customize the proposed similarity measure using any of the approaches described above is not trivial, and will be subject of our future work.

Acknowledgments This research has been partially supported by the Spanish Government projects Cognitio (TIN2012-38450-C03-03) and PerSO (TIN2014-55006-R).

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, *7*(1), 39–59.

Alqadah, F., & Bhatnagar, R. (2011). Similarity measures in formal concept analysis. *Annals of Mathematics and Artificial Intelligence*, *61*(3), 245–256.

Armengol, E., & Plaza, E. (2003). Relational case-based reasoning for carcinogenic activity prediction. *Artificial Intelligence Review*, *20*(1-2), 121–141.

Ashburner, M. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, *25*, 25–29.

Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., & Patel-Schneider, P.F. (Eds.) (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. New York: Cambridge University Press.

Bergmann, R., & Stahl, A. (1998). Similarity measures for object-oriented case representations. In: *Proc. European workshop on case-based reasoning, ewcbr-98, lecture notes in artificial intelligence* (pp. 8–13). Springer Verlag.

Bisson, G. (1992). Learing in FOL with a similarity measure. In: *Proceedings of AAAI 1992* (pp. 82–87).

Bodenreider, O., Smith, B., Kumar, A., & Burgun, A. (2007). Investigating subsumption in SNOMED CT: an exploration into large description logic-based biomedical terminologies. *Artificial Intelligence in Medicine*, *39*, 183–195. doi:10.1016/j.artmed.2006.12.003. http://portal.acm.org/citation.cfm?id=1240342.1240604.

Carpenter, B. (1992). *The logic of typed feature structures*. New York.

Cojan, J., & Lieber, J. (2010). An algorithm for adapting cases represented in an expressive description logic. In: *Case-based reasoning. Research and development, 18th international conference on case-based reasoning, ICCBR 2010, Alessandria, Italy, July 19-22, 2010. Proceedings* (pp. 51–65).

d'Amato, C., Staab, S., & Fanizzi, N. (2008). On the influence of description logics ontologies on conceptual similarity. In: *Proceedings of the 16th international conference on knowledge engineering, lecture notes in computer science* (Vol. 5268, pp. 48–63). Springer-Verlag.

Emde, W., & Wettschereck, D. (1996). Relational instance based learning. In: L. Saitta (ed.) *Machine learning - proceedings 13th international conference on machine learning* (pp. 122–130). Morgan Kaufmann Publishers.

Fanizzi, N., dAmato, C., & Esposito, F. (2008). Learning with kernels in description logics. In: *Inductive logic programming* (pp. 210–225). Springer.

Formica, A. (2006). Ontology-based concept similarity in formal concept analysis. *Information Sciences*, *176*(18), 2624–2641.

González-Calero, P.A., Díaz-Agudo, B., & Gómez-Albarrán, M. (1999). Applying DLs for retrieval in case-based reasoning. In: *Proceedings of the 1999 description logics workshop (DL'99)*.

Grau, B.C., Halaschek-Wiener, C., Kazakov, Y., & Suntisrivaraporn, B. (2010). Incremental classification of description logics ontologies. *JAR*, *44*(4), 337–369.

Horridge, M. (2009). A practical guide to building OWL ontologies using protege 4 and CO-ODE tools. Edition 1.2. Tech. rep., The University Of Manchester. http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_2.pdf.

Horváth, T., Wrobel, S., & Bohnebeck, U. (2001). Relational instance-based learning with lists and terms. *Machine Learning*, *43*(1-2), 53–80.

Hutchinson, A. (1997). Metrics on terms and clauses. In: *ECML '97: Proceedings of the 9th European conference on machine learning, lecture notes in computer science* (Vol. 1224, pp. 138–145). Springer.

Jaccard, P. (1901). Etude comparative de la distribution florale dans une portion des Alpes et du Jura. Impr. Corbaz.

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In: *Proceedings of the twentieth international conference (ICML 2003)* (pp. 321–328). AAAI Press.

Kazakov, Y., & Klinov, P. (2013). Incremental reasoning in el+ without bookkeeping. In: T. Eiter, B. Glimm, Y. Kazakov, M. Krötzsch (eds.), *Description logics, CEUR workshop proceedings* (vol. 1014, pp. 294–315.) CEUR-WS.org.

Kramer, S., Lavrač, N., & Flach, P. (2001). *Propositionalization approaches to relational data mining*. Springer.

van der Laag, P.R.J., & Nienhuys-Cheng, S.H. (1998). Completeness and properness of refinement operators in inductive logic programming. *Journal of Logic Programming*, *34*(3), 201–225.

Larson, J., & Michalski, R.S. (1977). Inductive inference of VL decision rules. *SIGART Bulletin*, *63*(63), 38–44. doi:10.1145/1045343.1045369.

Lehmann, J., & Hitzler, P. (2007). A refinement operator based learning algorithm for the LC description logic. In: H. Blockeel, J. Ramon, J.W. Shavlik, P. Tadepalli (eds.), *ILP, Lecture notes in computer science* (Vol. 4894, pp. 147–160). Springer.

Motik, B. (2006). Reasoning in description logics using resolution and deductive databases. Ph.D. thesis, Univesitat Karlsruhe (TH), Karlsruhe, Germany.

Nienhuys-Cheng, S.H. (1997). *Distance between Herbrand interpretations. A measure for approximations to a target concept*. Springer.

Ontañón, S., & Plaza, E. (2009). On similarity measures based on a refinement lattice. in: D. Wilson, L. McGinty (eds.) *Proceedings of ICCBR-2009, no. 5650 in lecture notes in artificial intelligence* (pp. 240–255). Springer-Verlag.

Ontanón, S., & Plaza, E. (2012). Similarity measures over refinement graphs. *Machine Learning*, *87*, 57–92.

Parsia, B., Halaschek-Wiener, C., & Sirin, E. (2006). Towards incremental reasoning through updates in OWL-DL. In: *Reasoning on the web workshop-WWW2006*. Edinburgh.

Ramon, J. (2002). Clustering and instance based learning in first order logic. Ph.D. thesis, Ph. D. Thesis, KU Leuven.

Raymond, J.W., Blankley, C.J., & Willett, P. (2003). Comparison of chemical clustering methods using graph- and fingerprint-based similarity measures. *Journal of Molecular Graphics and Modelling*, *21*(5), 421–433.

Raymond, J.W., Gardiner, E.J., & Willett, P. (2002). Rascal: calculation of graph similarity using maximum common edge subgraphs. *Computer Journal*, *45*(6), 631–644.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. arXiv: cmp-lg/9511007.

Rouveirol, C. (1994). Flattening and saturation: two representation changes for generalization. *Machine Learning*, *14*(2), 219–232.

Sánchez-Ruiz-Granados, A.A., González-Calero, P.A., & Díaz-Agudo, B. (2009). Abstraction in knowledge-rich models for case-based planning. In: *Case-based reasoning research and development, proc. 8th ICCBR, lecture notes in computer science* (Vol. 5650, pp. 313–327). Springer.

Sánchez-Ruiz-Granados, A.A., Ontañón, S., González-Calero, P.A., & Plaza, E. (2011). Measuring similarity in description logics using refinement operators. In: *ICCBR* (pp. 289–303).

Ullman, J.D. (2000). Information integration using logical views. *Theoretical Computer Science*, *239*(2), 189–210.

Willett, P., Barnard, J.M., & Downs, G.M. (1998). Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, *38*(6), 983–996.