Refinement-based Similarity Measure over DL Conjunctive Queries *

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

[1] Dep. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid (Spain)
antsanch@fdi.ucm.es, pedro@sip.ucm.es
[2] Computer Science Department
Drexel University
Philadelphia, PA, USA 19104
santi@cs.drexel.edu
[3] IIIA-CSIC, Artificial Intelligence Research Institute
Campus Univ. Aut. Barcelona, 08193 Bellaterra, Catalonia (Spain),
enric@iiia.csic.es

Abstract. Similarity assessment is a key operation in case-based reasoning and other 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 thus, 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.

1 Introduction

Description Logics (DL) are one of the most widespread standards for knowledge representation in many application areas [3]. 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. For these reasons, DL has also become the technology of choice for representing knowledge in knowledge-intensive case-based reasoning systems [19,7,10].

In this paper, we focus on the problem of similarity assessment in DL, in order to enable general purpose case-based reasoning systems that use this formalism to represent domain knowledge. Specifically, we focus in the problem of measuring similarity between individuals. The similarity measure presented in

* Supported by Spanish Ministry of Economy and Competitiveness under grant TIN2009-13692-C03-03

this paper, 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 [17,20]. The approach presented in this paper differs from previous work [8,20] in that the S_Q similarity is defined over the space of DL Conjunctive Queries, rather than in the space of DL concepts.

There are three main advantages in the S_Q similarity approach: 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), and 3) assessing similarity in the space of queries is computationally more efficient than assessing similarity in the space of concepts, as we will show in the experimental evaluation section.

The rest of this paper is organized as follows. Section 2 introduces the necessary concepts of Description Logics, Conjunctive Queries and Refinement Operators respectively. Then, in Section 3, we introduce a new refinement operator for Conjunctive Queries. Section 4 presents the S_Q similarity measure between individuals, which is illustrated with an example in §4.3. Section 5 presents an experimental evaluation of our approach. The paper closes with related work, conclusions and directions for future research.

2 Background

Description Logics [3] 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.

Description Logics (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 allow to 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), atomic roles (N_R) and individual names (N_I).

The expressivity and the reasoning complexity of a particular DL depends on the available concept constructors in the language. Although the proposed similarity measure is independent of the description logic being used (the only effect being computation time), in this paper we will use the $\mathcal{EL}$ logic, a light-weight DL with good computational properties that serves as a basis for the OWL 2 EL profile[4]. $\mathcal{EL}$ is expressive enough to describe large biomedical ontologies, like SNOMED CT [6] or the Gene Ontology [2],while maintaining important properties such as concept subsumption being polynomial. The $\mathcal{EL}$ concept constructs are the top concept, intersection and existential restrictions (see Table 1).

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

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

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	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

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

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	$range(R) = A$	$(x, y) \in R^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}$

Table 2. TBox axioms.

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 can be extended to complex 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 $C^{\mathcal{I}} \neq \emptyset$.

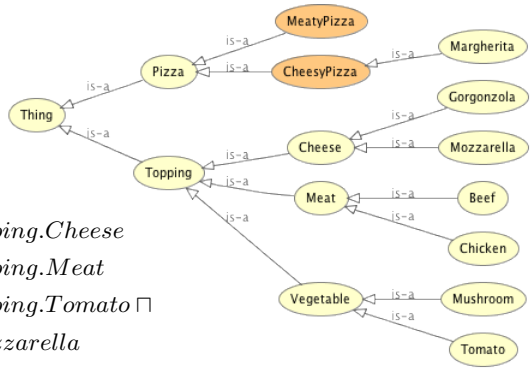
The basic reasoning operation in DL is *subsumption*, that induces a subconcept-superconcept hierarchy. We say that the concept C is subsumed by the concept D (C is more specific than D) if all the instances of C are also instances of D. Formally, C is subsumed by D w.r.t. the knowledge base $\mathcal{K}$ ($C \sqsubseteq_{\mathcal{K}} D$) 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 $C \not\equiv D$.

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	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

Table 3. ABox axioms.

TBox

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



ABox

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

Fig. 1. Example of knowledge base

Figure 1 shows an example knowledge base that we will use in the rest of the paper. 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. 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 that queries are used to retrieve data in 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 need as well a disjoint set of variable names (N_V).

Definition 1. (Conjunctive Query)
A DL conjunctive query $Q(\boldsymbol{x}, \boldsymbol{y})$ *is a logic formula* $\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ *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 an atomic role, and* x *and* y *are either individual names from* N_I *or variable names taken from the sets* $\boldsymbol{x}, \boldsymbol{y} \subset N_V$.

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

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 $Q(\boldsymbol{y})$, noted as $\mathcal{I} \models \exists \boldsymbol{y} : Q(\boldsymbol{y})$ 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 $Q(\boldsymbol{x}, \boldsymbol{y})$ w.r.t. a knowledge base $\mathcal{K}$ is a variable substitution θ that maps the answer variables in $\boldsymbol{x}$ to individuals in $\mathcal{K}$ such that the boolean query $Q(\boldsymbol{x}\theta, \boldsymbol{y})$ 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 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.

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

For example, given the knowledge base in Figure 1, let us consider the queries:

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

Now, the query Q_1 below 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 marguerita pizza although there is no individual of type tomato explicitly asserted in the ABox.

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* [21] can be used to reduce query containment to query answering in DL [16]. 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 $\mathcal{K}_Q = (\mathcal{T}, \mathcal{A}_Q)$.

Note that Definition 4 assumes that both Q and Q' share the same set of answering variables, which is enough for the purposes of this paper. For example, considering the pizza knowledge base, 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.

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

2.3 Refinement Operators

This section briefly summarizes the notion of *refinement operator* and the concepts relevant for this paper (see [13] for a more in-depth analysis of refinement operators). Refinement operators are defined over *quasi-ordered sets*. 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* and *transitive*. 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 5. (Refinement Operator)
A refinement operator ρ *over a quasi-ordered set* $(S, \leq)$ *is a function such that* $\forall a \in S : \rho(a) \subseteq \{b \in S | b \leq a\}$.

In other words, refinement operators (sometimes called *downward refinement operators*) generate elements of S which are "smaller" (which in this paper means "more specific"). The complementary notion of an *upward refinement operator*, that generates elements that are "bigger", also exists, but is irrelevant for this paper. 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 $\forall a, b \in S | a \leq b : a \in \rho^*(b)$.
- 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, *locally 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 $\leq$ (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. In this way, we only need to define a refinement operator for DL conjunctive queries to specialize or generalize them.

3 A 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

$$[A_2(x_1) \notin Q \wedge A_2 \sqsubset A_1 \wedge \nexists A' : A_2 \sqsubset A' \sqsubset A_1]$$

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

(R2) Concept Introduction

$$[x_1 \in V(Q) \wedge A_1 \in max\{A \in N_A \mid \forall A'(x_1) \in Q : A \not\sqsubseteq A' \wedge A' \not\sqsubseteq A\}]$$

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

(R3) Role Introduction

$$[x_1, x_2 \in V(Q) \wedge R_1 \in max\{R \in N_R \mid \forall R'(x_1, x_2) \in Q : R \not\sqsubseteq R' \wedge R' \not\sqsubseteq R\}]$$

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

(R4) Variable Introduction

$$[x_1 \in VI(Q), x_2 \in N_V \setminus V(Q)]$$

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

(R5) Variable Instantiation

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

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

Rules R1 and R2 refine a query either specializing an existing type or introducing a new type that is neither more general nor more specific than the existing ones (for example a sibling in the concept hierarchy). Rule R3 works analogously to R2 but introducing roles instead of concepts. Note that we do not provide a rule to specialize role assertions since the $\mathcal{EL}$ logic does not allow role hierarchies. Rule R4 introduces a new quantified variable in the query, and R5 binds an existing variable to a concrete individual in the knowledge base.

For the sake of space we do not provide proofs, but it is easy to verify that the previous refinement operator is locally finite and not proper. The refinement operator is also complete if we only consider the space of DL conjunctive queries

with a *fixed set of answer variables* (none of the above rules adds new answer variables) in which *all the variables represent distinct individuals*.

Although the assumption that all the variables are different restricts the set of queries that can be represented, it also simplifies to a large degree the similarity assessment process that we introduce in the following section, since it prevents refinement chains of infinite length in which all the queries are equivalent:

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

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.

4 Similarity Based on Query Refinements

The similarity S_Q proposed in this paper consists of two main steps (described in the following two subsections). First, given individuals a and b, we transform them into conjunctive DL queries, Q_a and Q_b. Second, using the refinement operator presented above, we measure the similarity between Q_a and Q_b.

4.1 From Individuals to Queries

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

Definition 6. (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* $R(b,c) \in G_a$
- $\forall R \in N_R, C \in N_C$, *if* $C(b) \in G_a$ *and* $R(b,c) \in \mathcal{A}$ *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 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, what is more important, no information is lost in the translation.

For example, next we show an individual graph and its equivalent DL query:

$$\begin{aligned} G_{p1} = {} & Pizza(p1) \wedge hasTopping(p1, t1) \wedge Chicken(t1) \wedge \\ & hasTopping(p1, t2) \wedge Vegetable(t2) \\ Q_{p1}(\{x_1\}, \{y_1, y_2\}) = {} & Pizza(x_1) \wedge hasTopping(x_1, y_1) \wedge Chicken(y_1) \wedge \\ & hasTopping(x_1, y_2) \wedge Vegetable(y_2) \end{aligned}$$

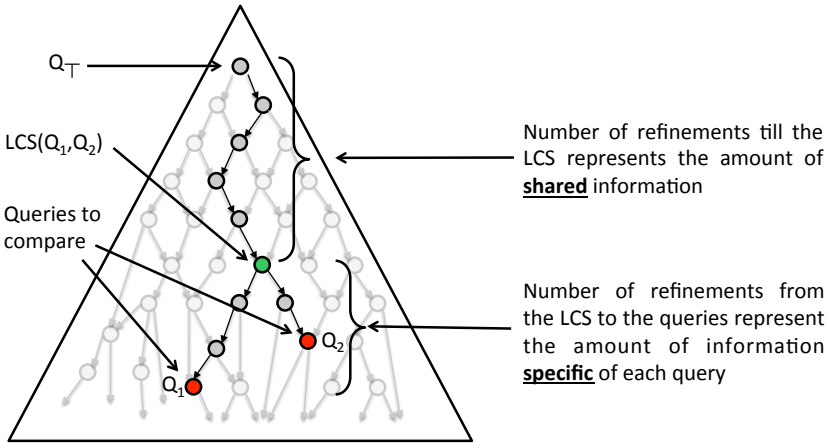


Fig. 2. Query distance based on refinements.

4.2 Similarity over in the CQ space

Our proposed similarity measure for conjunctive DL queries is based on the following intuitions (see Figure 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. Note, however, that since our refinement operator is not proper, some of the refinements in the refinement chain do not add new information to the previous query, and they should not be taken into account. The length of the chain of useful refinements (those that produce proper specializations) 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. In our experiments, we used a greedy search algorithm to compute this length, which does not ensure obtaining the shortest chain, but that is computationally efficient.

Third, given any two queries, their *least common subsumer* (LCS) is the most specific query which subsumes both. 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, which 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 contain 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:

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

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.

4.3 Example

In this section we show an example of the S_Q similarity works. Suppose we want to compute the similarity between a pizza margherita $p1$ and a pizza $p2$ with some vegetable and chicken. Figure 3 shows the queries representing both pizzas and the chain of refinements used to compute their similarity. Note that the refinements marked with an asterisk do not add new information and therefore they are not taken into account while computing the length of the refinement paths (steps 5 and 9 do not add new information because the role *hasTopping* has range *Topping* and thus we can infer that type for y_3 and y_4). Their LCS describes the common part of the pizzas: both have at least two ingredients and one of them is a vegetable. Their similarity is determined as follows:

$$S_Q(p1, p2) = S_\rho(Q_{p1}, Q_{p2}) = \frac{6}{6+3+5} = 0.43$$

5 Experiments

In order to evaluate the S_Q similarity measure, we used the trains data set shown in Figure 4 as presented by Michalski [14]. Like in our previous work on similarity assessment [20], 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."

We compared our similarity measure against 7 others: $S_{DL\rho}$ [20], a similarity measure for the $\mathcal{EL}$ description logic; González et al. [10], a similarity measure for acyclic concepts in description logic; RIBL [9], which is a Horn clause similarity measure; SHAUD [1], which is a similarity measure for feature terms; and S_λ, S_π, and $S_{w\pi}$ [18], which are similarity measures for feature terms but also based on the idea of refinement operators. For RIBL, we used the original version of the trains dataset, for SHAUD, S_λ, S_π, and $S_{w\pi}$, we used the feature term version

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 $S_\rho(Q_{p1},Q_{p2})$

of the dataset used in [17], which is a direct conversion from the original Horn clause dataset without loss, and for the DL similarity measures, we used the version created by Lehmann and Hitzler [15].

We compared the similarity measures in five different ways:

- Classification accuracy of a nearest-neighbor algorithm.
- *Average best rank* of the first correct example: if we take one of the trains, and sort the rest of the trains according to their similarity with the selected train, which is the position in this list (rank) of the first train with the same solution as the selected train (West or East).
- Jaro-Winkler distance: the Jaro-Winkler measure [22] can be used to compare two orderings. We measure the similarity of the rankings generated by our similarity measure with the rankings generated with the others.
- Mean-Square Difference (MSD): the mean square difference with respect to our similarity measure, S_Q.
- Average time take to compute similarity between two individuals.

Table 4 shows the results we obtained by using a leave-one-out evaluation. Concerning classification accuracy, we can see that our similarity measure S_Q

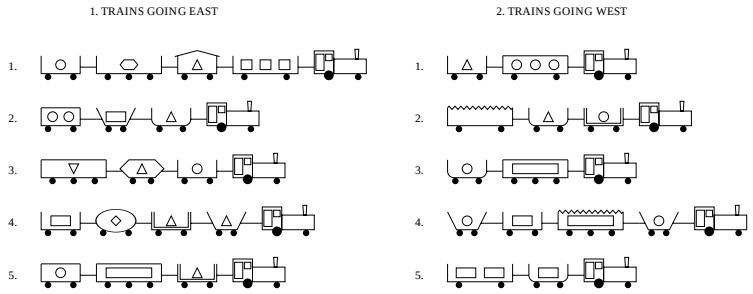


Fig. 4. Trains data set as introduced by Michalski [14].

	S_Q	$S_{DL\rho}$	González et al.	RIBL	SHAUD	S_λ	S_π	$S_{w\pi}$
Accuracy 1-NN	70%	70%	50%	60%	50%	40%	50%	80%
Best Rank	1.4	1.4	1.5	2.0	2.0	2.3	2.1	1.7
Jaro-Winkler	-	0.75	0.68	0.79	0.73	0.76	0.76	0.77
MSD	-	0.03	0.21	0.07	0.06	0.07	0.11	0.16
Avg. Time	0.55s	175.74s	0.01s	0.01s	0.07s	0.04s	0.00s*	0.00s*

Table 4. Comparison of several similarity metrics in the trains dataset (* These times do not take into account data preprocessing, required for these two techniques). Note that the Jaro-Winkler and MSD values are computed with respect to S_Q.

achieves a high classification accuracy, higher than most other similarity measures, except $S_{w\pi}$. We would like to emphasize that the trains data-set is only apparently simple, since the classification criteria is a complex pattern which involves several elements from different cars in a train. The only similarity measure that came close is $S_{w\pi}$, which achieved an 80% accuracy (it misclassified trains west 1 and west 3). Concerning the average best rank, our measure obtains the best score (tied with our previous measure $S_{DL\rho}$). Concerning the Jaro-Winkler and MSD results, we can see that in the trains data set S_Q produces similarities similar to $S_{DL\rho}$, RIBL, and S_λ.

Where our new similarity measure stands out is in terms of time. We can see that, compared to our previous $S_{DL\rho}$ similarity measure, the S_Q similarity is very fast. This is because 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.

The other similarity measure for DL (González et al.'s) is much faster, but it is specialized to individual graphs that can be represented as trees, and would not work for individual graphs that contain cycles.

In summary, S_Q is a new practical approach to assess similarity for expressive DL, with a similar classification accuracy or better than existing similarity measures, but more general than González et al.'s, and more efficient than our previous measure $S_{DL\rho}$.

6 Related Work

The work presented in this paper extends our previous work on similarity on Description Logics [20], 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 we might lose information when converting individuals to concepts).

D'Amato et al. [8] 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 have been proposed in order to assess similarity between individuals or concepts without requiring the use of a good sample of individuals. González et al. [10] 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.

In addition to similarity in Description Logics, there has been a significant amount of work in the more general topic of similarity assessment for other forms of complex data representation formalisms. Hutchinson [12] presented a distance based on the anti-unification of two terms. The Hutchinson distance is the addition of the sizes of the variable substitutions required to move from the anti-unification of two terms to each of these terms. This measure is related to the refinement-based approaches in [20] and [18], but is more coarse grained.

RIBL (Relational Instance-Based Learning) is an approach to apply lazy learning techniques while using Horn clauses as the representation formalism [9]. An earlier similarity measure related to RIBL was that of Bisson [5]. Horváth et al [11] presented an extension of RIBL that is able to deal with lists and terms. The downside of the RIBL approach is that specialized measures have to be defined for different types of data, while other approaches, such as the ones based on refinement operators, do not have this downside.

Bergmann and Stahl [4] present a similarity metric 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 decomposition" idea as RIBL, and limiting the approach to tree representations.

SHAUD, presented by Armengol and Plaza [1], is another similarity measure following the "hierarchical decomposition" approach but designed for feature terms. SHAUD also assumes that the terms do not have cycles, and in the same way as RIBL and Bergmann and Stalh's it can handle numerical values by using specialized similarity measures for different data types.

7 Conclusions and Future Work

This paper has presented a new approach to assess similarity between individuals in Description Logics. Our approach is based on first converting the individuals to conjunctive queries, and then assessing the similarity between the queries. Converting individuals to queries has several advantages with respect to converting individuals to concepts, and then assessing the similarity between the concepts: first, the conjunctive query language is shared among different DLs, and thus, our similarity measure is more generic (although in this paper we focused on the $\mathcal{EL}$ logic). Second, search in the space of queries is more efficient than search in the space of concepts, thus gaining a computational advantage. Our empirical results show that the resulting measure obtains similar results as previous comparable measures, but at a much lower computational cost.

As part of our future work, in addition to evaluation with larger and more complex datasets, we would like to fully explore the applicability of our similarity measure for more expressive description logics. Specifically, we would like to investigate the tradeoffs between relaxing the requirement of having a complete refinement operator (reducing the search space, and thus the computational complexity), and the performance of the resulting similarity measure.

References

1. Armengol, E., Plaza, E.: Relational case-based reasoning for carcinogenic activity prediction. Artif. Intell. Rev. 20(1-2), 121–141 (2003)
2. Ashburner, M.: Gene ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, New York, NY, USA (2003)
4. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: Proc. European Workshop on Case-Based Reasoning, EWCBR-98. pp. 8–13. Lecture Notes in Artificial Intelligence, Springer Verlag (1998)
5. Bisson, G.: Learing in FOL with a similarity measure. In: Proceedings of AAAI 1992. pp. 82–87 (1992)
6. Bodenreider, O., Smith, B., Kumar, A., Burgun, A.: Investigating subsumption in SNOMED CT: An exploration into large description logic-based biomedical terminologies. Artif. Intell. Med. 39, 183–195 (March 2007), `http://portal.acm.org/citation.cfm?id=1240342.1240604`

7. Cojan, J., Lieber, J.: 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 (2010)
8. d'Amato, C., Staab, S., Fanizzi, N.: 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 (2008)
9. Emde, W., Wettschereck, D.: Relational instance based learning. In: Saitta, L. (ed.) Machine Learning - Proceedings 13th International Conference on Machine Learning. pp. 122 – 130. Morgan Kaufmann Publishers (1996)
10. González-Calero, P.A., Díaz-Agudo, B., Gómez-Albarrán, M.: Applying DLs for retrieval in case-based reasoning. In: In Proceedings of the 1999 Description Logics Workshop (DL'99) (1999)
11. Horváth, T., Wrobel, S., Bohnebeck, U.: Relational instance-based learning with lists and terms. Machine Learning 43(1-2), 53–80 (2001)
12. Hutchinson, A.: 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 (1997)
13. van der Laag, P.R.J., Nienhuys-Cheng, S.H.: Completeness and properness of refinement operators in inductive logic programming. Journal of Logic Programming 34(3), 201–225 (1998)
14. Larson, J., Michalski, R.S.: Inductive inference of VL decision rules. SIGART Bull. 63(63), 38–44 (1977)
15. Lehmann, J., Hitzler, P.: A refinement operator based learning algorithm for the LC description logic. In: Blockeel, H., Ramon, J., Shavlik, J.W., Tadepalli, P. (eds.) ILP. Lecture Notes in Computer Science, vol. 4894, pp. 147–160. Springer (2007)
16. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. Ph.D. thesis, Univesitat Karlsruhe (TH), Karlsruhe, Germany (January 2006)
17. Ontanón, S., Plaza, E.: Similarity Measures over Refinement Graphs. Machine Learning 87, 57–92 (2012)
18. Ontañón, S., Plaza, E.: On similarity measures based on a refinement lattice. In: Wilson, D., McGinty, L. (eds.) Proceedings of ICCBR-2009, pp. 240 – 255. No. 5650 in Lecture Notes in Artificial Intelligence, Springer-Verlag (2009)
19. Sánchez-Ruiz-Granados, A.A., González-Calero, P.A., Díaz-Agudo, B.: 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 (2009)
20. Sánchez-Ruiz-Granados, A.A., Ontañón, S., González-Calero, P.A., Plaza, E.: Measuring similarity in description logics using refinement operators. In: ICCBR. pp. 289–303 (2011)
21. Ullman, J.D.: Information integration using logical views. Theor. Comput. Sci. 239(2), 189–210 (2000)
22. Winkler, W.E., Thibaudeau, Y.: An application of the Fellegi-Sunter model of record linkage to the 1990 U.S. decennial census. In: U.S. Decennial Census. Technical report, US Bureau of the Census (1987)