Kernels for $\mathcal{EL}^{++}$ Description Logic Concepts

Łukasz Józefowski[1], Agnieszka Ławrynowicz[1], Joanna Józefowska[1], Jedrzej Potoniec[1], and Tomasz Łukaszewski

Institute of Computing Science, Poznan University of Technology, ul. Piotrowo 2, 60-965 Poznan, Poland
{jjozefowski,alawrynowicz,jjozefowska,jpotoniec,tlukaszewski}@cs.put.poznan.pl

1 Introduction

Various kernel functions have been introduced for structured data such as sequence data, graphs, text, images or vectors [1]. This paper deals with structured data in the form of *description logic (DL)* [2] knowledge bases. Description logics (DLs) are an important formalism as they provide theorethical foundations for the formal ontologies expressed in *Web Ontology Language (OWL)*[1]. Kernels for DLs may be exploited to solve a multitude of tasks that may be dealt with similarity-based methods, especially in the context of so-called Semantic Web applications. However, so far only very few kernels have been introduced for DLs, and no kernels for $\mathcal{EL}^{++}$ in particular.

The contribution of this paper is in proposing two kernel functions for measuring similarity of concepts represented in the $\mathcal{EL}^{++}$ description logic [3] that are both based on the proposed canonical form of $\mathcal{EL}^{++}$ concepts. $\mathcal{EL}^{++}$ is an interesting fragment of DL languages family. It has been chosen as an underlying formalism for OWL 2 EL standard due to its practical computational features (tractable reasoning), and in the same time as a language that embraces the expressivity of several major real-life ontologies, especially from life sciences domain, such as SnoMed, Gene Ontology, or most of GALEN.

2 Knowledge representation language

2.1 Description logics as knowledge representation formalism

Basic elements in DLs are: *atomic concepts* (denoted by A), and abstract *atomic roles* (denoted by R, S). *Complex descriptions* (denoted by C, D) are built by using concept and role *constructors*. Furthermore, let by N_C, N_R, N_I denote the sets of *concept names*, *abstract role names* and *individual names* respectively.

A DL *knowledge base*, KB, is formally defined as: $KB = (\mathcal{T}, \mathcal{A})$. $\mathcal{T}$ is called a TBox, and it contains axioms dealing with how concepts and roles are related to each other. In particular, we will further refer to axioms such as: $C \sqsubseteq D (R \sqsubseteq S)$ or $C \equiv D (R \equiv S)$. Axioms of the first kind are called *inclusions*, while axioms of the second kind are called *equalities*. An equality whose left-hand side is an

[1] http://www.w3.org/TR/owl-features

Table 1: Transformation rules for a concept C to an $\mathcal{EL}^{++}$ canonical form.

TR1 if $C = A \wedge C \equiv D \in \mathcal{T}$ then $C \leftarrow D$ (replace C by D)
TR2 if $C = A \wedge C \sqsubseteq D \in \mathcal{T}$ then $C \leftarrow C \sqcap D$
TR3 if $C = \{a\} \wedge KB \models A(a)$ then $C \leftarrow C \sqcap A$

atomic concept is a *definition*. An inclusion whose left-hand side is an atomic concept is a *partial definition*. The atomic concepts occurring in $\mathcal{T}$ are divided into two sets: the symbols that occur on the left-hand side of definition axioms, called *defined* concepts, and the symbols that occur only on the right-hand side of the axioms, called *primitive* concepts. Importantly, in this work we require that a terminological part $\mathcal{T}$ of our KB is *acyclic*. $\mathcal{A}$ is called an ABox. It contains assertions about individuals such as $C(a)$ (a is an instance of the concept C) and $R(a, b)$ (a is R-related to b).

DLs may also support reasoning with *concrete datatypes* such as strings or integers. Within this work, we are interested in *concrete roles* P which are interpreted as binary relations $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}}$. We will further denote by N_P a set of concrete role names.

2.2 Canonical Form for $\mathcal{EL}^{++}$ Concepts

In order to compute the kernels proposed in this paper to measure the similarity of given concepts, we require these concepts to be previously transformed to a normal form. We assume that all the atomic concepts appearing in considered $\mathcal{EL}^{++}$ concept descriptions are primitive concepts. To ensure this, we envisage a pre-processing step that consist on expanding all defined concepts in a given concept description. In order to further exploit the semantics of a KB, we envisage also other pre-processing rules.Let by C denote a component of a given concept description that is to be transformed, and assume to operate on a TBox $\mathcal{T}$ in normal form [3]. Then the rules from Tab. 1 are exhaustively applied. Note that by an application of the transformation rules, we avoid such situation, where, for example, a defined concept compared by our measures with a concept formed by its right-hand side of a definition would be assessed as totally dissimilar.

In order to define the canonical form, let us first denote by $\mathsf{primn}(C)$ the set of all the primitive and nominal concepts that occur at the top-level conjunction of C, and by $\mathsf{ex}_R(C)$ denote the set of the concept descriptions C' that occur in existential restrictions $\exists R_i.C'$ at the top-level conjunction of C.

Definition 1 ($\mathcal{EL}^{++}$ canonical form). *A concept description C is in $\mathcal{EL}^{++}$ canonical form iff $C \equiv \top$ or $C \equiv \bot$ or if*

$$C = \prod_{B_i \in \mathsf{primn}(C)} B_i \sqcap \prod_{\substack{R_k \in N_R \\ C' \in \mathsf{ex}_R(C)}} \exists R_k.C' \sqcap \prod_{P_l \in N_P} \exists P_l.f$$

where for any $R_k \in N_R$, every concept description C' in $\mathsf{ex}_R(C)$ is in canonical form. In general, each of the higher level intersection may be also replaced by $\top$.

3 Kernels for concepts in $\mathcal{EL}^{++}$ canonical form

It may be observed that the canonical form for concepts apart from top concept $\top$ and bottom concept $\bot$ is a composite structure that can consist of three basic parts: primitive and nominal concepts part, abstract role part and concrete role part, denoted by PN, $ABST$, $CONC$, respectively. Without the loss of generality let us also assume that $\top$ and $\bot$ belong to PN. Each concept in the canonical form consists of at most three different basic parts. Let C be a concept in $\mathcal{EL}^{++}$ that is in the canonical form. Further, let W_{PN}, W_{ABST} and W_{CONC} denote the sets of all proper words in $\mathcal{EL}^{++}$ that can appear respectively in PN, $ABST$ and $CONC$ parts of any concept of that language. Let us assume that in case where any of the parts: PN, $ABST$, $CONC$ is empty, it is represented by special $EMPTY$ word that does not belong to the language $\mathcal{EL}^{++}$. Let us define the following new parts $PN' = PN$ if PN part is not empty and $PN' = EMPTY$ otherwise, $ABST' = ABST$ if $ABST$ is not empty and $ABST = EMPTY$ otherwise, finally $CONC' = CONC$ if $CONC$ part is not empty and $CONC' = EMPTY$ otherwise. Let us define the following sets: $W'_{PN} = W_{PN} \cup \{EMPTY\}$, $W'_{ABST} = W_{ABST} \cup \{EMPTY\}$ and $W'_{CONC} = W_{CONC} \cup \{EMPTY\}$. Let us now transform each concept C that is in canonical form into a 3-tuple in $W'_{PN} \times W'_{ABST} \times W'_{CONC}$. Let us define K_{PN}, K_{ABST}, K_{CONC} to be a kernel for respectively, primitive and nominal part, abstract role part and concrete role part of concept C. The similarities between any two concepts C_1 and C_2 in the $\mathcal{EL}^{++}$ canonical form, can be calculated using the following convolution kernel [4, 6]

$$K_{\oplus,R}(C_1, C_2) = K_{PN}(C_1, C_2) + K_{ABST}(C_1, C_2) + K_{CONC}(C_1, C_2) \tag{1}$$

K_{PN} is a kernel for primitive and nominal part of C that is defined on $W_{PN} \times W_{PN}$. This kernel may by extended using *zero extension* to the kernel on $W'_{PN} \times W'_{PN}$ by defining $K_{PN}(x, y) = 0$ if either x or y is not in W_{PN} i.e. if x or y is the EMPTY word. Kernel K_{ABST} defined on $W_{ABST} \times W_{ABST}$ can be *zero extended* to the kernel on $W'_{ABST} \times W'_{ABST}$ by defining $K_{ABST}(x, y) = 0$ if either x or y is not in W_{ABST} i.e. if x or y is the EMPTY word. Finally, kernel K_{CONC} defined on $W_{CONC} \times W_{CONC}$ can be *zero extended* to the kernel on $W'_{CONC} \times W'_{CONC}$ by defining $K_{CONC}(x, y) = 0$ if either x or y is not in W_{CONC}.Now the K_{PN} kernel for PN parts of two concepts C_1 and C_2 in the $\mathcal{EL}^{++}$ canonical form can be defined as follows using set intersection kernel [1].

$$K_{PN}(C_1, C_2) = k_{\cap}(\mathsf{primn}(C_1), \mathsf{primn}(C_2)) \tag{2}$$

Observe now that the $ABST$ role part of concept C is a composite structure that can be represented by a set of tuples of the form (R_i, C'), where R_i is a role in $ABST$ parts and C' is a filler of the role R_i. For such structure the decomposition relation [6] R_{ABST} can be defined in the following way.

Definition 2. *Let X be a set of tuples of part $ABST$ of concept C and $\boldsymbol{x} = x_{ABST} \in ABST$ define elements of X. The relation that $\boldsymbol{x}$ is a part of X can be represented by relation R_{ABST} on set $X \times X$, where $R_{ABST}(\boldsymbol{x}, x)$ is true iff $\boldsymbol{x}$ is an element of X.*

The kernel K_{ABST} for $ABST$ role parts of two concepts C_1 and C_2 in the $\mathcal{EL}^{++}$ canonical form is defined using convolution kernel in the following way.

$$K_{ABST}(C_1, C_2) = \sum_{x \in R^{-1}_{ABST}(C_1)} \sum_{y \in R^{-1}_{ABST}(C_2)} K_r(x, y) \tag{3}$$

where K_r is a kernel defined on tuples Tu_1, Tu_2 of the form (R_i, C'), where R_i is a role in $ABST$ parts and C' is a filler of the role R_i. The K_r kernel can be defined as a convolution kernel in the following way, which exploits additional semantical information coming from the KB, namely role hierarchies.

$$K_r(Tu_1, Tu_2) = K_{co-rooted}(R_1, R_2) K_{\oplus,R}(C'_1, C'_2) \tag{4}$$

$K_{co-rooted}(R_1, R_2)$ is the all co-rooted tree kernel [1] for two role hierarchies of roles R_1 and R_2 that are part of $ABST$ and both are rooted in top object property. Since the fillers C'_1 and C'_2 of the roles R_1, R_2 respectively can be any valid concepts in the canonical form the kernel defined in Formula (1) is called recursively.

Let us now define the kernel for the $CONC$ role part of the concept C. Analogously, the $CONC$ role part of concept C is a composite structure that can be represented by a set of tuples of the form (P_i, f), where P_i is a role in $CONC$ parts and f is a concrete value argument of the role P_i. For such structure the decomposition relation R_{CONC} can be defined analogously to the R_{ABST}. Now the kernel K_{CONC} for $CONC$ role parts of two concepts C_1 and C_2 can be defined as follows.

$$K_{CONC}(C_1, C_2) = \sum_{x \in R^{-1}_{CONC}(C_1)} \sum_{y \in R^{-1}_{CONC}(C_2)} K_l(x, y) \tag{5}$$

where K_l is a kernel defined on tuples Tu_1, Tu_2 of the form (P_i, f), where P_i is a role in $CONC$ parts and f is a concrete value argument of the role P_i. The K_l kernel can be defined as a convolution kernel in the following way.

$$K_l(Tu_1, Tu_2) = K_{co-rooted}(P_1, P_2) K_{type}(f_1, f_2) K_T(f_1, f_2) \tag{6}$$

where $K_{co-rooted}(P_1, P_2)$ is the all co-rooted tree kernel for two role hierarchies of roles P_1 and P_2 rooted in top data property, and K_T is a type specific kernel for values f_1 and f_2. Since the fillers f_1, f_2 of the roles P_1, P_2 respectively may be of various types, by using the K_{type} kernel we assure that values of the same types are compared.

The second kernel makes use of the fact that $\mathcal{EL}^{++}$ concepts can be represented as directed labelled trees, similarly as described in [7, 8]. The $\mathcal{EL}^{++}$ *concept description tree* $T = (V, E)$ is a directed labelled tree, where V is the finite set of *nodes*, and $E \subseteq V \times N_{R|P} \times V$ is the set of *edges*. The root of the tree is labelled with either $\top$, $\bot$ or all atomic/nominal concepts occurring in $\mathsf{primn}(C)$. For each existential restriction $\exists R_k.C'$ occurring at the top level of C, it has an R_k-labelled edge to the root of a subtree corresponding to C'. For each $\exists P_l.f$ restriction at the top level of C, it has an P_l-labelled edge to a leaf

corresponding to value f (the nodes corresponding to concrete values are not expanded further). An empty label in a node is equivalent to $\top$.

This kernel takes the structure of the directed labeled tree as well as the semantics of its nodes and edges into account. It is mainly based on the random walk graph kernel proposed in [9], where given two labelled graphs G_1 and G_2, the number of matching labelled random walks is counted. The value of the random walk kernel for two graphs G_1 and G_2 is calculated as the sum over the kernel values of all pairs of walks within these two graph.

Unfortunately in that kernel attributes of two nodes v_1 of graph G_1 and w_2 of graph G_2 are considered similar if they are completely identical which is rather unusual for concept description trees of two $\mathcal{EL}^{++}$ concepts. For that reason we decide to use similar approach to that presented for protein graphs kernel in [5] and we analogously redefine the random walk kernel for the concept description tree, where the kernel for each step in the random walk is the product of the kernel of the original node, the destination node and the edge between them.

Definition 3 (Step kernel). *For $i \in \{1, \ldots, n-1\}$ the step kernel if defined as*
$k_{step}((v_i, v_{i+1}), (w_i, w_{i+1})) = k_{node}(v_i, w_i) k_{node}(v_{i+1}, w_{i+1}) \cdot$
$k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1}))$*, where k_{edge} is defined as follows*
$k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1})) = k_{type}((v_i, v_{i+1}), (w_i, w_{i+1})) \cdot$
$k_{edge\ labels\ abstract|concrete}((v_i, v_{i+1}), (w_i, w_{i+1}))$
and for $i \in \{1, \ldots, n\}$, k_{node} is defined as follows
$k_{node}(v_i, w_i) = k_{type}(v_i, w_i) k_{node\ labels\ concepts\ set|concrete}(v_i, w_i)$.

In the Definition 3 of step kernel we used three basic types of kernels: type kernel [5], node labels kernel and edge labels kernel. The purpose of the type kernel is to assure that only nodes and edges of the same types are compared. In our case, there are four basic types: 'concept set node' (corresponding to nodes labelled with a set of concepts from $\mathsf{primn}(C)$), 'concrete value node' (corresponding to nodes labelled with a concrete value f), 'abstract edge' (labelled with an abstract role R_k), and 'concrete edge' that is those labelled with a concrete role P_l.

Now we will define kernels to be used within the same type. In case of nodes labelled with sets of primitive/nominal concepts, we will use the intersection kernel [1]. Consider two sets $\mathcal{C}_1, \mathcal{C}_2$ of primitive or nominal concepts, then:

$$k_{node\ label\ concept\ set}(\mathcal{C}_1, \mathcal{C}_2) = k_{\cap}(\mathcal{C}_1, \mathcal{C}_2) \tag{7}$$

The node label kernel for two nodes of concrete value types can be evaluated using any kernel suitable for both concrete value types. In order to assure that only values of the same type are compared we use also the type kernel. $k_{node\ labels}$ for two values nodes f_i and f_i for $i \in \{1, \ldots, n\}$ is defined as follows

$$k_{node\ label\ concrete\ value}(f_i, f_i) = k_{type}(f_i, f_i) k_T(f_i, f_i) \tag{8}$$

where k_T is a kernel defined for particular type of values of nodes.

There are basically two types of edge kernels, namely the kernel for an abstract role and for a concrete role. For each R_k role as well as for each P_l role, we can consider role hierarchy, rooted respectively in top object property and

top data property. Using this idea the edge label kernel for R_k role and for P_l role are defined below respectively.

$$k_{edge\ labels\ abstract}(R_i, R_j) = k_{co-rooted}(R_i, R_j) \tag{9}$$

$$k_{edge\ labels\ concrete}(P_i, P_j) = k_{co-rooted}(P_i, P_j) \tag{10}$$

4 Conclusions and Future Work

In this paper, we have introduced two new kernel functions for computing similarity between $\mathcal{EL}^{++}$ description logic concepts. It should be noted, however, that our approach is suitable as well for comparing DL individuals. In particular, individuals may be lifted to concept descriptions by an exploitations of a most specific concept (msc) reasoning service or its approximation.

In the future, we will implement the proposed kernels within the framework of an ontology-based data mining extension to Rapid Miner[2], RMonto[3], we develop at the Poznan University of Technology. A supplementary version of this work (submitted to the ECML/PKDD'2011 CoLISD workshop) may be found at `http://www.cs.put.poznan.pl/alawrynowicz/EL_kernels.pdf`.

Acknowledgements. We acknowledge the support from the Polish Ministry of Science and Higher Education (grant N N516 186437) and from European Community 7th framework program ICT-2007.4.4 (grant 231519 "e-LICO: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science"). We also thank Adam Woznica for the references on labelled graph kernels.

References

1. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)
3. Baader, F., Brand, S., Lutz, C.: Pushing the el envelope. In: In Proc. of IJCAI 2005, Morgan-Kaufmann Publishers (2005) 364–369
4. De Raedt, L.: Logical and Relational Learning. Springer Verlag (2008)
5. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S.V.N., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. In: ISMB (Supplement of Bioinformatics). (2005) 47–56
6. Haussler, D.: Convolution kernels on discrete structures. Technical report (1999)
7. Baader, F., Molitor, R., Tobies, S.: Tractable and decidable fragments of conceptual graphs. In Tepfenhart, W.M., Cyre, W.R., eds.: ICCS. Volume 1640 of Lecture Notes in Computer Science., Springer (1999) 480–493
8. Lehmann, J., Haase, C.: Ideal downward refinement in the el description logic. In: Proceedings of the 19th international conference on Inductive logic programming. ILP'09, Berlin, Heidelberg, Springer-Verlag (2010) 73–87
9. Gaertner, T., Flach, P., Wrobel, S.: S.: On graph kernels: Hardness results and efficient alternatives. In: In: Conference on Learning Theory. (2003) 129–143

[2] http://rapid-i.com/
[3] http://semantic.cs.put.poznan.pl/RMonto/doku.php?id=start