Description Logics for Documentation

Carlo Meghini

Consiglio Nazionale delle Ricerche
Istituto di Scienza e Tecnologie della Informazione, Pisa, Italy
carlo.meghini@isti.cnr.it

Abstract. Much of the activity in a digital library revolves around collecting, organizing and publishing knowledge about the resources of the library, in the form of metadata records. In order to document such activity, digital librarians need to express knowledge about the metadata records they produce. This knowledge, which we call *documentation knowledge,* may express *e.g.*, provenance, trustability, or access restrictions of the records. Today, documentation knowledge is mostly represented in digital libraries via RDF. We propose a new type of information system, called documentation system, as a basic component of a digital library allowing to represent and reason about both domain and documentation knowledge in an expressive language such as OWL.

1 Introduction

In a digital library, it is often necessary to represent and reason about two different kinds of knowledge. One kind is *domain* knowledge, which in is typically embodied in metadata and ontologies, used by the end-users, for instance to discover and access the resources of the library. The other kind of knowledge concerns domain knowledge and is used by digital librarians in order to manage the resources of the library. For instance, a digital librarian might want to describe the provenance, or the degree of trust or the access policy of a metadata record. We call this latter kind of knowledge as *documentation* knowledge, since documentation is one of the main reasons that brings it into life. Documentation knowledge is a primary matter dealt with by curators in libraries and archives. In general, it shows up in any organization that devotes resources to the documentation of artifacts, events, users, services and in general any resource that is of value to the organization. Its scope of application is therefore quite ample.

Documentation knowledge consists of factual and ontological statements about individuals, concepts and relations of the domain of dicourse, and as such it can be expressed and reasoned upon using standard logics, such as OWL [10]. However, a problem arises if documentation knowledge is to be used *together with* domain knowledge, as it happens in digital libraries. The problem is due to the fact that the individuals of documentation knowlegde are domain knowledge statements, and in order to express knowledge about resources and about the statements used to describe such resources *in the same language,* one needs very powerful languages, whose expressive power goes beyond that of first-order logic.

Such languages, though, are hardly usable in digital libraries, because their negative computational properties are inadequate to digital library requirements.

The Resource Description Frameowrk (RDF) [6] is a case in point. RDF allows to express metadata records as statements having the described resource as subject and metadata elements as properties. It also allows to express a certain amount of documentation knowledge, by allowing properties as subjects in statements. However, as soon as the expressivity of the language goes beyond that of RDF Schema [3] by including constructs from Description Logics, serious computational problems arise. Indeed, the combination of RDF with Description Logics yields the language OWL Full [7], which is undecidable in spite of the decidability of its two constituents.

In this study, we tackle the problem of representing and reasoning about both domain and documentation knoweldge by means of languages significantly more expressive than RDF, but we follow a different approach than RDF. Our approach is based on the *simultaneous* usage of two different logics: the *object* logic devoted to represent domain knowledge, and the *documentation* logic (or *doc*-logic for short) devoted to represent documentation knwoledge. We view the resulting information system, which we call *documentation system,* as the backbone of a digital library, used for representing and reasoning about domain and documentation knowledge both in an independent and in a joint way, the latter option offering an innovative query functionality. As such, the present study extends the model presented in [8], which also allows to represent documentation knowledge, but whose expressive power does not go beyond that of RDF.

In order to illustrate our approach in general terms, we choose the Description Logic $\mathcal{ALCO}$ [1] as the object logic, relying on the abstract syntax of DLs, which is more concise than the official OWL notation [4].

2 Motivation

Much of the activity of digital librarians revolves around *metadata records.* A metadata record can be viewed as consisting of three separate parts: a description; a described object, the *subject* of the metadata record; and the attribution of the description to the subject. A description is a set of features. For instance, a description in a metadata record may consist of two features: (1) being a book and (2) being titled "Waverley". Each one of these features may be understood as describing a set of resources. In particular, the former feature describes anything that is a book, while the latter describes anything that is titled "Waverley". Taken together, the two features form a description that describes any resource satisfying *both* of them. In Description Logics (DLs for short), descriptions are represented by concepts, and in fact the two terms "description" and "concept" are synonymous; we will follow the same convention. In the DL $\mathcal{ALCO}$, the above description may be represented by the concept $\mathsf{Book} \sqcap \exists \mathsf{Title}.\{\mathit{Waverley}\}$ The attribution of a description to a particular subject is the act of asserting that that subject does indeed possess *simultaneously* all the features that make up the description. This attribution produces a metadata record. In a

DL, attribution is realized by concept assertion, therefore in $\mathcal{ALCO}$, the metadata record as a whole can be expressed by the following concept assertion: $b : (\mathsf{Book} \sqcap \exists \mathsf{Title}.\{\mathit{Waverley}\})$ where b is the DL individual that stands for the subject, a book in this case. In a DL knowledge base (KB for short), such an assertion is placed in the ABox, the component of the KB holding factual knowledge. The axioms of DLs can be used for modelling ontologies, understood as vocabularies that establish, via terminological axioms, the meaning of the concepts used in descriptions [5]. A terminological axiom in our example, could be the following: $\mathit{Book} \sqsubseteq (\exists \mathsf{Title.Literal} \sqcap \exists \mathsf{Author.Person})$ stating that every book has a title that is a literal, and an author that is a person. The ontology underlying an ABox is placed in the other component of a DL KB, the TBox.

The discussion so far seems to indicate that DLs are adequate representation languages for the knowledge contained in a digital library. Indeed, OWL [10], the W3C recommendations that stand for DLs in the semantic web architecture [2], is known and used in digital libraries. However, there is a fundamental requirement of digital libraries that DLs are not able to capture. This requirement comes from the fact that documentation, a central activity in a digital library, may be a *recursive* process; it may not be solely confined to the domain of discourse, but it may also concern the descriptions and metadata records used for documenting the resources of the domain of discourse. For example, a digital librarian may need to represent the fact that a metadata record has a certain provenance, (*i.e.*, it has been created by a certain person, on a certain date, as a result of a certain activity), or a certain degree of trustability; or that the record is subject to a certain set of access restrictions, or to a certain billing policy. In general, the same requirements arise in every information system that deals with the documentation of the individuals of the domain of discourse, be these individuals users, devices, or events.

In order to cope with the recursiveness of documentation, we argue that a knowledge representation language is needed that offers descriptions as first class citizens, that is as individuals on their own right, endowed with an identity and a structure. In addition, the language in question should offer the machinery to attribute a description to a certain individual, which may itself be a description, thereby coping with the recursiveness of documentation. DLs, in spite of their name, fall short of this requirement. They offer a rich machinery to create descriptions, yet these descriptions are not denotable as individuals, and therefore it is not possible to state any knowledge on them, other than assertions and axioms. As such, the support that DLs offer to documentation is practically limited. The situation, however, is not unrecoverable. The machinery of DLs can be used to remedy at this inconvenient, by slightly shifting the focus of representation from the individuals, concepts and roles in the domain of discourse, to descriptions made up of these. The shift is a form of reification and leads to a logic, the doc-logic, that allows the expression of documentation knowledge, in contrast to the object-logic that is used for representing domain knowledge.

In order to show how this can be done, in the next Section we introduce a doc-logic corresponding to the object-logic $\mathcal{ALCO}$. Our doc-logic will be doc-$\mathcal{ALCO}$, more simply called as *alco*.

3 Introducing *alco*

Suppose we wish to create the above description for the Waverley, and we also want such description to be identified as the individual d. In order to achieve our goal, in *alco* we describe the structure of d as the conjunction of two concepts, which we choose to identify as d_1 and d_2. We use the *alco* role CCId (for *Concept Conjunction Identification*) for associating the identifier of a conjunction to the identifiers of the conjuncts, as follows: $\mathsf{CCId}(d, d_1), \mathsf{CCId}(d, d_2)$ The role CCId binds an identifier to a concept *part*, and therefore we will call it a *binding* role.

Next, we need to state that d_1 identifies an atomic concept, say *Book*. For this, we use the *alco* role ACId (for *Atomic Concept Identification*) as follows: $\mathsf{ACId}(d_1, Book)$ An assertion on the role ACId is in fact an assigment of an identifier to a whole concept, not just to a part of it, as in the case of binding roles; therefore will call ACId a *concept identifying role,* or more simply an *identifying role.* Of course, we could choose to use an atomic concept as an identifier of itself (*e.g.*, the *alco* individual *Book* as an identifier of the $\mathcal{ALCO}$ concept *Book*) to make a doc-KB more readable; but we prefer to use different names in order to avoid confusion. We also introduce in *alco* the role CNId (for *Concept Negation Identification*) for identifying negation concepts as follows: $\mathsf{CNId}(d, e)$ assigns the identifier d to the negation of the concept identified by e. Also CNId is an identifying role.

As seen in the previous section, the title feature is expressed in $\mathcal{ALCO}$ as the concept $\exists Title.\{Waverley\}$, known in the DL world as *existential quantification.* In order to reduce this concept to something denotable as the individual d_2, we follow the same style followed above for concept conjunction, as follows:

- we introduce the identifying role ARId (for *Atomic Role Identification*), and use it to identify the $\mathcal{ALCO}$ role *Title* as d_3 by the assertion $\mathsf{ARId}(d_3, Title)$;
- we introduce the identifying role SCId (for *Singleton Concept Identification*) and use it to identify the $\mathcal{ALCO}$ singleton concept $\{Waverley\}$ as d_4 by the assertion $\mathsf{SCId}(d_4, Waverley)$;
- finally, we introduce two identifying roles ERId (for *Role Identification in an Existential Quantification*) and ECId (for *Concept Identification in an Existential Quantification*) and use them to bind d_2 to its constituent parts by the assertions: $\mathsf{ERId}(d_2, d_3), \mathsf{ECId}(d_2, d_4)$

Now we have completed the description of d, and can create the desired metadata record by attributing d to the book b that we want to describe. The obvious way to state the association between d and b, is to introduce the *alco* role CAss (for *Concept Assertion*) and use it as follows: $\mathsf{CAss}(d, b)$. The first argument of a CAss assertion is always a description identifier, whereas the second argument

identifies the subject, which can be any resource. The CAss role allows to represent in *alco* an $\mathcal{ALCO}$ concept assertion. Yet, it is not adequate to document metadata record, because there is no individual that denotes the resulting metadata record in a CAss assertion. An alternative way of proceeding is to coin a name, say m, for the metadata record that we want to create, and then connect m to d and b by using appropriate role assertions, namely: $\mathsf{MRD}(m, d), \mathsf{MRS}(m, b)$. There is an obvious relation between the just introduced roles and CAss. This relation will be captured by an axiom, introduced in next Section.

So far we have simply used a rather cumbersome notation for expressing a concept assertion. However, this notation has given us identifiers for the description and the metadata record that we have created, therefore we are in the position of representing documentation knowledge about them. Suppose we want to state that d was created by *John*. The latter feature can be expressed as a description identified by e as follows: $\mathsf{ARId}(e_1, Author), \mathsf{SCId}(e_2, John), \mathsf{ERId}(e, e_1), \mathsf{ECId}(e, e_2)$ The description e can be attributed to d by stating $\mathsf{CAss}(e, d)$.

Analogously we can document the metadata record m by specifying a creation time, a creation place and an author for it. We first create a description h for the $\mathcal{ALCO}$ concept $\exists Author.\{Sue\}$, using the *alco* roles (for brevity, we do not detail h); finally, we add the assertion: $\mathsf{CAss}(h, m)$ to the ABox of the doc-KB.

In order to make *alco* a full doc-logic, we need a role for representing (at the doc-level) concept subsumptions in an object-TBox. To this end, we introduce CSAx (for *Concept Subsumption Axiom*) with the intended meaning that: $\mathsf{CSAx}(e, d)$ states that the concept identified by e is a sub-concept of the concept identified by d.

4 Semantics

As a DL, *alco* has a well-defined semantics, based on the notion of interpretation. This notion, however, turns out to license undesired situations. In this section we discuss these undesired situations, and introduce axioms for ruling them out. These axioms are intended to be in the TBox of *any* doc-KB, since they express the semantics of the *alco* roles. In order to ease the expression of the axioms, we introduce some atomic concepts:

- OInd, OACon and OARol, for representing individuals, atomic concepts and atomic roles of the object-DL, respectively; these concepts are defined as follows (as customary, $\exists R$ abbreviates $\exists R.\top$):

$$\mathsf{OInd} \equiv \exists \mathsf{SCId}^{-}; \ \mathsf{OACon} \equiv \exists \mathsf{ACId}^{-}; \ \mathsf{OARol} \equiv \exists \mathsf{ARId}^{-}$$

- Concepts denoting the different types of concept and role identifiers:

$$\mathsf{AtomId} \equiv \exists \mathsf{ACId}; \ \mathsf{ConjId} \equiv \exists \mathsf{CCId}; \ \mathsf{NegId} \equiv \exists \mathsf{CNId}$$
$$\mathsf{SomeId} \equiv \exists \mathsf{ERId}; \ \mathsf{SingId} \equiv \exists \mathsf{SCId}; \ \mathsf{RoleId} \equiv \exists \mathsf{ARId}$$

- The concept ConcId denoting concept identifiers:

$$\mathsf{ConcId} \equiv \mathsf{AtomId} \sqcup \mathsf{ConjId} \sqcup \mathsf{NegId} \sqcup \mathsf{SomeId} \sqcup \mathsf{SingId}$$

Identification axioms These axioms capture the proper behaviour of the binding and the identifying roles. First, identifying roles must behave like functions (in OWL terms, they are functional object properties [9]).

$$(\geq 2\ \mathsf{ACId}) \sqsubseteq \bot; \quad (\geq 2\ \mathsf{ARId}) \sqsubseteq \bot; \quad (\geq 2\ \mathsf{CNId}) \sqsubseteq \bot$$
$$(\geq 2\ \mathsf{ERId}) \sqsubseteq \bot; \quad (\geq 2\ \mathsf{ECId}) \sqsubseteq \bot; \quad (\geq 2\ \mathsf{SCId}) \sqsubseteq \bot$$

Concerning the binding role CCId, every identifier used as a first argument in an assertion on this role, must appear as a first argument also in at least another assertion on the same role, because a conjunction has at least two conjuncts. We express this condition as follows: $(= 1\ \mathsf{CCId}) \sqsubseteq \bot$.
Moreover, for every $\mathsf{ERId}(d, e)$ assertion, there must be exactly one $\mathsf{ECId}(d, e')$ assertion, for any individuals e, e'; and viceversa. In order to capture this constraint, which we call *pairing constraint,* it is not sufficient to include the axiom $\exists\mathsf{ERId} \equiv \exists\mathsf{ECId}$ in the TBox. This is due to the fact that a DL KB is interpreted under the Open World Assumption. As a consequence, a KB whose ABox contains *only* the assertion $\mathsf{ERId}(d, R)$ and whose TBox contains the axiom $\exists\mathsf{ERId} \equiv \exists\mathsf{ECId}$ is not inconsistent; rather, the KB is understood as implicitly stating that there exists some unknown individual z such that $\mathsf{ECId}(d, z)$ is true in every interpretation. We cannot therefore capture the pairing constraint as a TBox axiom; we will do it in a different way, illustrated in the next Section.

Finally, binding and identifying roles must all together satisfy the obvious constraint that the domain of each one of them be disjoint from the domain of each of the others, otherwise it may happen that the same identifier be used to identify two concepts of different kinds. Assuming the above mentioned pairing constraints are in place, we need to consider only one of ERId and ECId because the domains of these two roles are made equivalent by the pairing constraint. So, overall we need to declare mutual disjointness of the domain of six roles; this requires fifteen axioms, all of the same type. For brevity, we only state the five axioms stating the disjointness of atomic concept identifiers from the other types of concept identifiers:

$$(\mathsf{AtomId} \sqcap \mathsf{ConjId}) \sqsubseteq \bot; (\mathsf{AtomId} \sqcap \mathsf{NegId}) \sqsubseteq \bot; (\mathsf{AtomId} \sqcap \mathsf{SomeId}) \sqsubseteq \bot;$$
$$(\mathsf{AtomId} \sqcap \mathsf{SingId}) \sqsubseteq \bot; (\mathsf{AtomId} \sqcap \mathsf{RoleId}) \sqsubseteq \bot$$

Syntactic axioms Syntactic axioms are those making sure that the syntax of the object-DL concepts is properly captured by the assertions of the doc-DL.

A basic constraint of every DL, is that the symbols used for individuals, atomic concepts and roles come from three disjoint alphabets.

$$(\mathsf{OACon} \sqcap \mathsf{OARol}) \sqsubseteq \bot; (\mathsf{OACon} \sqcap \mathsf{OInd}) \sqsubseteq \bot; (\mathsf{OARol} \sqcap \mathsf{OInd}) \sqsubseteq \bot$$

We also must make sure that concepts are properly formed in a doc-KB. To exemplify, if in the doc-KB there is the assertion that d identifies a negation, *i.e.*, $\mathsf{CNId}(d, d_1)$, then d_1 must identify something, and a concept in particular. The second part of this constraint can be captured via an axiom, imposing that

anything that appears as a second argument in a CNId assertion be a concept identifier, and the same for all the other concept constructors. This can be done by introducing the following axioms:

$$\top \sqsubseteq (\forall \mathsf{CCId}.\mathsf{ConcId}); \top \sqsubseteq (\forall \mathsf{CNId}.\mathsf{ConcId}); \top \sqsubseteq (\forall \mathsf{ECId}.\mathsf{ConcId})$$

However, the first part of the constraint, namely that there be in the KB an explicit assertion binding d_1 to a concept, cannot be formalized as an axiom, again due to the Open World Assumption. Similarly to pairing constraints, we have therefore to find a different way of expressing this kind of constraints, which we call the *syntactic constraints.*

Metadata axioms Metadata axioms concern identifiers of metadata records. Our intended notion of a metadata record requires that a metadata record concern exactly one individual. In order to capture this intention, we introduce the following axiom: $(\geq\ 2\ \mathsf{MRS}) \sqsubseteq \bot$. Moreover, we need to enforce a pairing constraint for MRD and MRS, in the sense that for every assertion of the kind $\mathsf{MRD}(m, d)$, we want the KB to contain at least one assertion of the form $\mathsf{MRS}(m, i)$, for some individual i; and viceversa. For the reasons given above, also this kind of pairing constraint has to be captured in a different way. Finally, we include an axiom for capturing the previously mentioned relation between the role MRD and MRS from one side, and the role CAss from the other side. In fact, every time we use MRD and MRS for structuring a metadata record m, as in $\mathsf{MRD}(m, d)$ and $\mathsf{MRS}(m, b)$, we are implicitly asserting that the d describes b, that is, that $\mathsf{CAss}(d, b)$. In order to make this connection happen in a doc-KB, we introduce the axiom:

$$\mathsf{MRD}^{-} \circ \mathsf{MRS} \sqsubseteq \mathsf{CAss} \tag{1}$$

where $\circ$ is the role composition operator. Notice that this axiom leaves the freedom of inserting CAss assertions without the corresponding MRD and MRS assertions. In other words, it is possible to create concept assertions that are not metadata records, such as for instance the last assertion of the previous example.

Inference Axioms Inference axioms are required in order to model the proper behavior of the CAss and the CSAx roles.

Concerning the CAss role and returning to our example, from the assertions $\mathsf{CAss}(d, b)$ and $\mathsf{CCId}(d, d_1)$ it should follow the assertion $\mathsf{CAss}(d_1, b)$, because d identifies a concept conjunction and d_1 identifies one of the conjuncts. By the same argument, it also should follow $\mathsf{CCId}(d, d_2)$. More generally, modelling the semantics of CAss means laying down all the rules that capture implied concept assertions, ultimately leading to the axiomatization of (object-level) instance checking in the doc-logic.

Likewise, modelling the semantics of the CSAx role amounts to axiomatize (object-level) concept subsumption in the doc-logic. In other words, the doc-logic does not only have to capture the syntax of the corresponding object-logic, but also its inference mechanism.

The proof theory of DLs provides us with sound and complete inference methods for both instance checking and submsumption. These methods can be encoded in *alco* by means of axioms and semantic conditions, exactly in the same way the syntax rules of the object-logic are encoded. As a result, the users of the doc-KB would be able to exploit the implicit domain knowledge, typically for performing ask operations. In particular, in order to check whether an individual i is an instance of an object-concept c, it suffices to identify c via an individual d and then ask whether $\mathsf{CAss}(d, i)$ logically follows from the doc-KB. The same machinery also allows to do some consistency checking, for instance checking whether an individual is an instance of the $\bot$ concept, or of a contradictory description. However, the price to be paid for achieving this goal would be very high, as the encoding of inference would make the doc-logic very complex. In what follows, we will present a much simpler method for attaining the same goal.

In order to preserve the semantics of the *alco* roles, we assume that the doc TBox does not contain any axiom concerning these roles other than those given above.

5 Strong consistency

Suppose we want to add to the KB the assertion that b is not a book. In *alco,* the complext concept $\neg Book$ must first be created and identified, say as the individual f, by using the *alco* roles, and then f must be declared to be the negation of the atomic concept *Book*. There is already an identifier for the latter concept, namely d_1, therefore all our replacement librarian must do, is to state: $\mathsf{CNId}(f, d_1)$ so that he can finally assert that b is an instance of f, by using the *alco* role assertion: $\mathsf{CAss}(f, b)$. By adding the last two assertions to those introduced in the Section 3, the consistency of the doc-KB is not broken. However, the represented knowledge is clearly inconsistent from an intuitive point of view, since b is asserted to be a book and not a book at the same time.

One way to capture pairing and syntacting constraints while at the same time ruling out the last kind of inconsistency, is to try to transform a consistent doc-KB into its corresponding object-KB. In order to do so, $\mathcal{ALCO}$ concepts must be extracted from the assertions of the doc-ABox. If the doc-KB suffers from a pairing or a syntactic inconsistency, then this extraction is not possible because the assertions in the doc-ABox do not conform to the syntax of the doc-logic. Otherwise the extraction is possible, and it allows to determine, for each concept identifier d in the doc-ABox, the concept identified by d, that we denote as $\nu_D(d)$ ($\nu(d)$ for simplicity). Once the function ν is determined, the doc-KB can be transofrmed into its corresponding object-KB, and the obtained object-KB can be checked for consistency. If this check succeeds, then the doc-KB satisfies all the intuitive consistency criteria. This approach gives us a simple method for checking consistency of the doc-KB. A further advantage of it, is that we no longer need to model object level instance checking or subsumption as implicit CAss and CSAx assertions, respectively; we can transform the doc-KB and perform these inferences on the resulting object-KB, by relying on well-known

$\nu(d)$	if the doc-ABox contains
A	$\mathsf{ACId}(d, A)$
R	$\mathsf{ARId}(d, R)$
$\nu(e_1) \sqcap \ldots \sqcap \nu(e_n)$	$\mathsf{CCId}(d, e_1), \ldots, \mathsf{CCId}(d, e_n)$, n maximal
$\neg\nu(e)$	$\mathsf{CNId}(d, e)$
$\exists\nu(e).\nu(f)$	$\mathsf{ERId}(d, e)$ and $\mathsf{ECId}(d, f)$
$\{o\}$	$\mathsf{SCId}(d, o)$
ω	otherwise

Table 1. Assignment of object-DL concepts to identifiers

algorithms. This is the route that we will follow in the rest of the paper. To this end, we will first define how to determine the function ν from a given doc-KB D. Based on the existence of $\nu(d)$ for each concept identifier d, we will define a stronger consistency criterion for a doc-KB. Next, we state the transformation ϕ from a doc-KB to its corresponding object-KB and define the strongest consistency criterion of a doc-KB based on the consistency of its corresponding object-KB. The domain of the function ν is the set of concept identifiers:

$$dom(\nu) = \{d \mid (T, A) \models d : \mathsf{ConcId}\}$$

and can be efficiently determined as the set of identifiers that occur as first arguments in a binding or in an identifying role assertion. For each $d \in dom(\nu)$, the value of $\nu(d)$ is recursively defined in Table 1. By iterating this recursive computation on $dom(\nu)$, the function ν can be efficiently computed.

Intuitively, if $D = (T, A)$ is a consistent doc-KB, then exactly one of the conditions on the right column of Table 1, excluding the last row, is met by the ABox A, that is:

$$\nu(d) \neq \omega \text{ for all } d \in dom(\nu),$$

in other words ν *is total on* D. Based on this consideration we define a doc-KB $D = (T, A)$ to be *fully consistent* if $\nu(d)$ is total on D.

Let us now consider how to define the transformation ϕ. The DL *alco* offers the roles CSAx and CAss for representing the terminological axioms and the assertions of the object-KB, respectively. Therefore, it is natural to use assertions on the former role in order to derive the axioms in the object-TBox, and assertions on the latter role in order to derive the axioms in the object-ABox. Formally, given a doc-KB $D = (T, A)$, we have $\phi(D) = (\mathcal{T}, \mathcal{A})$ where:

$$\mathcal{T} = \{\nu(d) \sqsubseteq \nu(e) \mid \mathsf{CSAx}(d, e) \in A\}$$
$$\mathcal{A} = \{i : \nu(d) \mid (T, A) \models \mathsf{CAss}(d, i)\}$$

Some explanations are in order concerning the translation of CAss assertions. Any such assertion has the form $\mathsf{CAss}(d, i)$ where d is a description identifier and i is either a description identifier (such as e in the example above) or an individual denoting any other kind of resource (such as the book b in the example

above). In the latter case, i does not have to be translated, as there would be nothing to translate it into. In the former case, i is a concept identifier and d another concept identifier that describes i. If we translate both identifiers into the corresponding concepts, the result will be an assertion like $C : D$ where both C and D are object-concepts. In our example, the translation would look like:

$$(Book \sqcap \exists Ttile.\{Waverley\}) : \exists Author.\{John\}$$

The last expression reads "John created the description *Book titled 'Waverley'*", which is precisely what we want to say. However, the above is not a valid assertion in any DL, and this is the very reason why we set out to define the doc-DL. Therefore, we have no choice but to translate only the second identifier. This will result in the concept assertion: $d : \exists Author.\{John\}$ This means that d will appear in the object-KB, but its connection with the description that it identifies (*i.e.*, $(Book \sqcap \exists Ttile.\{Waverley\})$) is lost in the object-KB. Notice that we require $\mathsf{CAss}(d, i)$ to be *not only* explicitly asserted, but also implicitly present in the KB, typically as a consequence of the creation of a metadata record. Table 2 shows the translation of the doc-KB of our example.

Based on these considerations, we say that a doc-KB $D = (T, A)$ is *strongly consistent* if D is fully consistent and $\phi(D)$ is a consistent $\mathcal{ALCO}$ KB. Moreover, we define a *documentation system* $\mathcal{S}$ to be a pair $\mathcal{S} = \langle D, O \rangle$ where D is a doc-KB and O is an object-KB, such that $O = \phi(D)$.

6 Querying a documentation system

The two KBs in a DS can be queried individually, based on the kind of knwoledge they store. As customary, we will denote the answer to an object query C_o against an object-KB O as $\textsc{Ask}_o(C_o, O)$, or simply $\textsc{Ask}_o(C_o)$, and define: $\textsc{Ask}_o(C_o) = \{\ i \mid O \models i : C_o\}$ Likewise, we will use $\textsc{Ask}_d(C_d)$ for denoting the result of asking doc-query C_d to a doc-KB and define: $\textsc{Ask}_d(C_d) = \{\ i \mid O \models i : C_d\}$

doc-ABox	ν	object-ABox
$\mathsf{CCId}(d, d_1)$, $\mathsf{CCId}(d, d_2)$	$d \mapsto \nu(d_1) \sqcap \nu(d_2)$	
$\mathsf{ACId}(d_1, Book)$	$d_1 \mapsto Book$	
$\mathsf{ERId}(d_2, d_3)$, $\mathsf{ECId}(d_2, d_4)$	$d_2 \mapsto \exists\nu(d_3).\nu(d_4)$	
$\mathsf{ARId}(d_3, Title)$	$d_3 \mapsto Title$	
$\mathsf{SCId}(d_4, Waverley)$	$d_4 \mapsto \{Waverley\}$	
$\mathsf{CAss}(d, b)$		$b : Book \sqcap$ $\exists Title.\{Waverley\}$
$\mathsf{MRD}(m, d)$, $\mathsf{MRS}(m, b)$		
$\mathsf{ERId}(e, Author)$, $\mathsf{ECId}(e, e_1)$	$e \mapsto \exists Author.\nu(e_1)$	
$\mathsf{SCId}(e_1, John)$	$e_1 \mapsto \{John\}$	
$\mathsf{CAss}(e, d)$		$d : \exists Author.\{John\}$
$\mathsf{ERId}(h, Author)$, $\mathsf{ECId}(h, h_1)$	$h \mapsto \exists Author.\nu(h_1)$	
$\mathsf{SCId}(h_1, Sue)$	$h_1 \mapsto \{Sue\}$	
$\mathsf{CAss}(h, m)$		$m : \exists Author.\{Sue\}$

Table 2. Summary of the running example

There is a third category of queries, which we call *mixed* queries, that can be asked to a DS. Mixed queries involve both domain and documentation knowledge, and can be of one of two kinds: (1) *mixed-object* queries, asking for the resourses of the object-KB that satisfy some property expressed in the doc-KB; for instance, a mixed-object query may ask for the editions of the *Waverley* that have been described by John; (2) *mixed-doc* queries, asking for the resourses of the doc-KB that satisfy some property expressed in the object-KB; for instance, a mixed-object query may ask for the metadata records of the *Waverley* that have been created by Sue. In order to express mixed queries, we need to address both the object- and the doc-KB. One way of doing so, is to use the operators $\textsc{Ask}_o$ and $\textsc{Ask}_d$ *within* queries. Using these two operators, the editions of the *Waverley* described by John can be expressed as follows:

$$Book \sqcap \exists Title.\{Waverley\} \sqcap \textsc{Ask}_d(\exists \mathsf{CAss}^{-}.\textsc{Ask}_o(\exists Author.\{John\}))$$

This is a concept denoting the individuals in the object-KB that are known to be books titled *Waverley,* and that are known in the doc-KB to be described by a description authored by *John.* Notice the double nesting of the Ask operator, without which it would not be possible to express the query. This implies that asking a doc-KB may require asking a object-KB. Let us now consider the mixed-doc query asking for the metadata records of the *Waverley* authored by Sue. Using the same ask operators introduced above, this query can be expressed as:

$$\exists \mathsf{MRD}.(\exists \mathsf{CAss}.\textsc{Ask}_o(\exists Title.\{Waverley\})) \sqcap \textsc{Ask}_o(\exists Author.\{Sue\})$$

This is a conjunction of two concepts: the first concept denotes the metadata records whose description describes an individual known in the object-KB to be titled *Waverley.* The second concept denotes the individuals known in the object-KB to be authored by Sue.

The syntax of our query language, which we call Documentation Query Language (DQL for short), is given by the following rules:

$$\begin{aligned} C &::= C_o \mid C_d \\ C_o &::= \text{any object-concept} \mid \textsc{Ask}_d(C_d) \\ C_d &::= \text{any doc-concept} \mid \textsc{Ask}_o(C_o) \end{aligned}$$

In order to give the semantics of DQL, the intepretations of the doc- and the object-DL are combined in a rather obvious way. For simplicity, we omit such specification, pointing out that it allows us to reduce query answering on a DS to query answering on a single KB whose TBox is given by the union of the TBoxes of the doc- and the object-KBs, and the same for the ABox. In order to show the effects of this result, let us now return to our example queries. From a semantical point of view, the former query above: $Book \sqcap \exists Title.\{Waverley\} \sqcap \textsc{Ask}_d(\exists \mathsf{CAss}^{-}.\textsc{Ask}_o(\exists Author.\{John\}))$ stated against our example DS in Table 2, is equivalent to the query $Book \sqcap \exists Title.\{Waverley\} \sqcap \exists \mathsf{CAss}^{-}.\exists Author.\{John\}$ stated against the union of the KBs shown in the first and third column of the table. The union contains the assertions:

$$b : Book \sqcap \exists Title.\{Waverley\}\ \mathsf{CAss}(d, b)\ d : \exists Author.\{John\}$$

clearly implying that b is in the answer to the query. Analogously, the latter query above: $\exists \mathsf{MRD}.(\exists \mathsf{CAss}.\textsc{Ask}_o(\exists \mathit{Title}.\{\mathit{Waverley}\})) \sqcap \textsc{Ask}_o(\exists \mathit{Author}.\{\mathit{Sue}\})$ stated against $\mathcal{S} = (D, O)$, is equivalent to the query $\exists \mathsf{MRD}.(\exists \mathsf{CAss}.\exists \mathit{Title}.\{\mathit{Waverley}\}) \sqcap \exists \mathit{Author}.\{\mathit{Sue}\}$ stated against $D \cup O$. The assertions:

$$\mathsf{MRD}(m, d) \;\; \mathsf{CAss}(d, b) \; b : \mathit{Book} \sqcap \exists \mathit{Title}.\{\mathit{Waverley}\} \; m : \exists \mathit{Author}.\{\mathit{Sue}\}$$

are in $D \cup O$, therefore m is in the answer to the query.

7 Conclusions

We have presented documentation systems as constituents of digital libraries, using description logics to represent and reason about domain and documentation knowledge at the same time. In particular, the KB of a documentation system consists of an object-KB and of a doc-KB. In the object-TBox, it is possible to express domain ontologies as terminological axioms, whereas in the object-ABox it is possible to express metadata records as DL concept assertions. In the doc-ABox, it is possible to make assertions involving descriptions and the metadata records of the object-KB Abox, thereby representing documentation knowledge. As such, a documentation system significantly extends the expressive capabilities of current digital libraries, mostly based on RDF.

References

1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2nd edition, 2003.
2. Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American Magazine*, May 2001.
3. Dan Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, WWW Consortium, February 2004.
4. Fabien Gandon and Guus Schreiber. RDF 1.1 XML syntax. Technical report, W3C Recommendation, 10 February 2004 25 February 2014.
5. N. Guarino. Formal ontology in information systems. In *Proceedings of FOIS 98*, pages 3–15. IOS Press, Amsterdam, 1998. Amended version.
6. Frank Manola and Eric Miller. RDF Primer. W3C Recommendation, WWW Consortium, February 2004.
7. Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, February 2004.
8. Carlo Meghini, Nicolas Spyratos, Tsuyoshi Sugibuchi, and Jitao Yang. A model for digital libraries and its translation to rdf. *Journal on Data Semantics*, 3(2):107–139, June 2014. ISSN 1861-2032.
9. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language structural specification and functional-style syntax (second edition). W3C recommendation, W3C, December 2012.
10. W3C OWL Working Group. OWL 2 Web Ontology Language document overview (second edition). W3C recommendation, W3C, December 2012.