Converting Mikrokosmos frames into Description Logics

P.J. Beltrán-Ferruz and **P.A. González-Calero** and **P.Gervás**
GAIA - Applied Artificial Intelligence Group
Dep. Sistemas Informáticos y Programación
Universidad Complutense de Madrid
C/ Juan del Rosal, 8. 28040 Madrid (Spain)
pablo@fdi.ucm.es, {pedro,pgervas}@sip.ucm.es

Abstract

Mikrokosmos contains an ontology plus a number of lexicons in different languages that were originally developed for machine translation. The underlying representation formalism for these resources is an ad-hoc frame-based language which makes it difficult to inter-operate Mikrokosmos with state-of-the-art knowledge-based systems.

In this paper we propose a translation from the frame-based representation of Mikrokosmos into Description logics. This translation allows us to automatically transform Mikrokosmos sources into OWL and thus provide a powerful ontology in the formalism of the semantic web. Furthermore, the reasoning mechanisms of Description Logics may also support knowledge acquisition and maintenance as well as its application in natural language processing systems.

1 Introduction

The Mikrokosmos project was originally an interlingual system for Knowledge-Based Machine Translation (KBMT) (Nirenburg, 1987) developed in the Computing Research Laboratory from New Mexico State University. Although KBMT was conceived for translation of domain specific texts, no further restrictions are imposed in the contents of the text. Therefore the creators of Mikrokosmos built a rich ontology that contains a lot of general concepts, more than 4.700 concepts that are connected with an average of other 14 concepts using attributes and relations (de Quesada, 2001).

KBMT is an expensive approach that requires a big effort on knowledge acquisition, and it has been considered impractical by some authors. For that reason, the creators of Mikrokosmos were especially concerned about developing real-size systems that would demonstrate the feasibility of their approach. Generating contents for the ontology was their first concern, while the use of a rigorous formalism for knowledge representation was not considered a priority (Moreno-Ortiz et al., 2002).

The work presented here is an effort to port Mikrokosmos into Description Logics (DL) in order to incorporate this resource into the systems we are developing. Our work on natural language generation integrates ontologies and case-based reasoning (CBR) (Diaz-Agudo et al., 2002), an approach that heavily relies on classification-based reasoning from DL.

Representing Mikrokosmos in DL should bring several benefits. Since DL is the underlying knowledge representation approach in the Semantic Web, a big number of supporting tools are being developed for acquiring and maintaining ontologies represented in some version of DL, such as OWL (Bechhofer et al., 2004). Giving a well-founded formal representation to Mikrokosmos should improve its quality by uncovering inconsistencies. Finally, porting Mikrokosmos to a formalism as popular as OWL will definitively increase its potential user community.

There are other efforts that convert ontologies from different representations to Description Logics languages. OntoMap is a web-site that provides access to upper-level ontologies and handcrafted mappings between them (Kiryakov et al., 2001). The initial set of ontologies contains Cyc, EuroWordnet and Mikrokosmos ontology, but it only deals with the top-level hierarchy. In case of Mikrokosmos it only contains 13 concepts. Their main effort involves the mapping between different ontologies. They also provide the resources in DAML+OIL. The project's goal is to facilitate easy access, understanding, and reuse of general-purpose ontologies and upper-level models.

The rest of this paper runs as follows. Next section describes the frame-based language used in Mikrokosmos ontology, and Section 3 describes the DL which is the target of the translation. Section 4 is dedicated to the mapping process and Section 5 evaluates this process. Section 6 points out future work, and finally Section 7 concludes the paper.

Concept	*Slot*	*Facet*	*Filler(s)*
REPLACEMENT-FOR	DEFINITION	VALUE	”when x is a replacement for y”
	IS-A	VALUE	PHYSICAL-OBJECT-RELATION, EVENT-RELATION
	INVERSE	VALUE	REPLACED-BY
	DOMAIN	SEM	EVENT, OBJECT
	RANGE	SEM	EVENT, OBJECT

Table 1: Example frame: REPLACEMENT-FOR

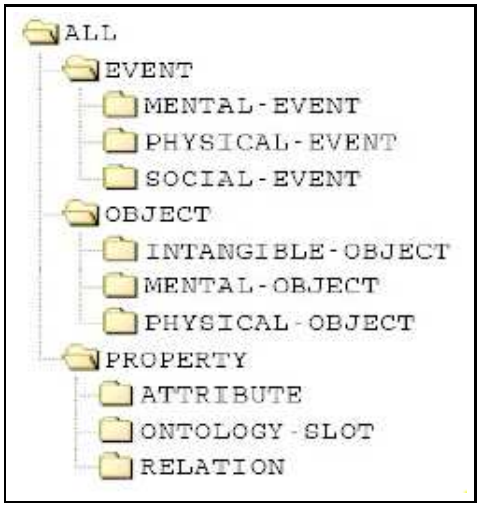

Figure 1: Mikrokosmos top hierarchy

2 Mikrokosmos ontology

In Mikrokosmos, ontology lists the definitions of concepts that are understood as corresponding to classes of thing and events in the world. Concepts are primitive symbols of a world model which includes objects, events and properties organized in a complex hierarchy of language-independent concepts (See top hierarchy of Mikrokosmos in Figure 1). The concepts are constructed following super ordinates, or hyponymy relations (IS-A links). In addition to its organization into a taxonomy via IS-A links, the ontology contain numerous other links between concepts, such as links using properties (Lonergan, 2001). For example DECEMBER has a relation with WINTER using the property PART-OF-OBJECT.

Each concept that makes up the ontology is language independent and is represented using frames. For example we can see the frame for concept REPLACEMENT-FOR in Table 1.

The format of Mikrokosmos Ontology is described in detail in (Nirenburg and Raskin, 2004). It formally introduces the syntax and the semantics of the ontology using a BNF grammar. We are most interested in how we can access to this information.

Ontology is saved in a text file using Spencer notation that is based on XML. There is another notation called Beale notation that is based on Lisp, but we will focus on Spencer notation.

In the XML based format we have the whole ontology represented in a list of RECORD entries. Definition of one CONCEPT requires one or more of these RECORD entries. Each entry contains four fields, that are: CONCEPT, SLOT, FACET, and FILLER.

The CONCEPT field can be filled by any *Name* of a concept of the ontology.

The second field in each entry is SLOT. This field can be filled with PROPERTY or any of its subclasses using IS-A links. There are two kind of *slot fillers*. One type are descendants of ATTRIBUTE or RELATION, that represent links between concepts in the hierarchy. The other type are descendants of ONTOLOGY-SLOT. We will call them *special slots*, and all of them have the sense of determining the structure of the ontology. Possible descendants of ONTOLOGY-SLOT are: DEFINITION, DOMAIN, INSTANCES, INVERSE, IS-A, RANGE, SUBCLASSES and some others that are less important; later in this section we will explain them in detail.

The third field is FACET, and it describes some finer distinctions between the possible fillers of the slot. Possibles FACET fillers are: VALUE, SEM, DEFAULT, INV, NOT, DEFAULT, DEFAULT-MEASURE and RELAXABLE-TO.

The last field is FILLER, and its value depends on the other fields, but generally it contains either a *Name* of a concept of the ontology or an instance.

Initially we can think that there are no restrictions in these representations, but there are some special slots that limit expressiveness. All CONCEPT frames have non-special and special slots. Special slots for all kinds of concepts are:

- DEFINITION: Definition in English of the concept.
- IS-A: It is used for asserting parents in the hierarchy.
- SUBCLASSES: It is used for listing concept children.
- INSTANCES, SPANISH1, ENGLISH1: They are only used in the leaves of OBJECT and

EVENT, and contains words of the dictionary.

Special slots which can only be present in all PROPERTY and only in PROPERTY concept frames are:

- DOMAIN: It has fillers usually filled with EVENTs[1] and/or OBJECTs and it determines whether a CONCEPT can have it as a SLOT.
- RANGE: It is used in RELATIONs and ATTRIBUTEs. In RELATIONs the RANGE slot has only the SEM facet. The fillers of the SEM facet are the names of concepts that are in the range of this RELATION. In ATTRIBUTEs the RANGE slot has only a VALUE facet. The VALUE facet is filled by all the possible literal or numerical values permissible for that ATTRIBUTE. The filler can also be a numerical range specified using appropriate mathematical comparison operators (such as $>$, $<$, ...).
- INVERSE: It is defined only for RELATIONs. It is mandatory for all RELATION frames. The INVERSE slot has only the Value facet which is filled by the name of the RELATION which is the Inverse of the given RELATION.
- MEASURED-IN: It is defined only for the descendants of the SCALAR-ATTRIBUTE concept frame. The MEASURED-IN slot is used to add a measuring unit for the number or scalar range that fills facets of the RANGE slot in SCALAR-ATTRIBUTE concept frames. The facet fillers of the MEASURED-IN slot are the daughters of the MEASURING-UNIT concept. The MEASURED-IN slot is used only in those SCALAR-ATTRIBUTE frames where MEASURING-UNIT has physical sense (e.g. for SIZE, AGE, etc.).

3 Description logics language: $\mathcal{SHIQ}$

DL are a family of logical formalisms that originated in the field of artificial intelligence as a tool for representation of conceptual knowledge. Since then, DLs have been successfully used in a wide range of application areas such as knowledge representation, reasoning about class-based formalisms (e.g. conceptual database models and UML diagrams), and ontology engineering in the context of the semantic web. The basic syntactic entities of DL are *concepts*, which are constructed from concept names (unary predicates) and role names (binary predicates) using the set of concept and role constructors provided by a particular DL (Lutz, 2003).

Our interest in Mikrokosmos ontology is to map its contents to a DL language. We have chosen $\mathcal{ALCQHI}_{\mathcal{R}^+}$ also known as $\mathcal{SHIQ}$ (Horrocks et al., 2000).

$\mathcal{SHIQ}$ is the basic logic $\mathcal{ALC}$ augmented with qualifying number restrictions, role restrictions, role hierarchies, inverse roles, and transitive roles.

$\mathcal{ALC}$ comprises concepts —denoting sets— as well as roles —denoting binary relations. Unlike roles, concepts can be compound. Compound concepts are constructed by the following operators: intersection $\sqcap$, union $\sqcup$, complementation $\neg$ —taking concepts as arguments—, and the value restrictions $\forall$, and $\exists$ —taking a role and a concept as their arguments. Formally, $\mathcal{ALC}$ is given by the following formation rules, where *c* denotes a concept symbol and *r* a role symbol (Schild, 1991):

$$C, D \longrightarrow c \mid \top \mid C \sqcap D \mid \neg C \mid \forall R.C$$
$$R \longrightarrow r$$

DL $\mathcal{SHIQ}$ is implemented in the RACER system (Haarslev and Moller, 2003). This makes it a desirable target representation for our ontology. For describing our ontology in $\mathcal{SHIQ}$ we will use the notation explained in Table 2, that contains denotational semantics for our language translation.

4 Mikrokosmos mapping to $\mathcal{SHIQ}$

Once we have identified DL language we want to use —$\mathcal{SHIQ}$— and we have described the Mikrokosmos ontology, we can proceed to map it.

The first step is to determine whether a concept is a class or a slot. Although in the Mikrokosmos ontology everything is a concept we need to distinguish between Mikrokosmos concepts that correspond to unary predicates —which map to DL classes— and Mikrokosmos concepts that correspond to binary predicates —which map to DL relations. EVENT, OBJECT and all of their subclasses will be unary predicates so they will be classes. Meanwhile PROPERTY and all its hierarchy except ONTOLOGY-SLOTs (see Figure 1) will be binary predicates so they will be slots. There are a few exceptions: concept ALL is **top** in DL and ONTOLOGY-SLOT and all of their subclasses are not mapped to DL language because they have the

[1] In this paper when we say a concept name in plural we are refering to this concept and his children, using links IS-A defined in the ontology.

[2] $\sigma(C)$ is the interpretation of a concept. Interpretation of a concept is the set of all individuals in the domain that satisfies description of the concept.

class-def (**primitive** | **defined**) CN	$\texttt{CN}(\sqsubseteq \mid \doteq)\top$
subclass-of $C_1 \ldots C_n$	$\sqcap\sigma^2(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
slot-constraint$_1$	$\sqcap\sigma(\textit{slot-constraint}_1)$
$\vdots$	$\vdots$
slot-constraint$_m$	$\sqcap\sigma(\textit{slot-constraint}_m)$
top | **thing** | **bottom**	$C \sqcup \neg C \mid C \sqcup \neg C \mid C \sqcap \neg C$
(C_1 **and** ... **and** C_n)	$(\sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n))$
(C_1 **or** ... **or** C_n)	$(\sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n))$
(**not** C)	$(\neg\sigma(C))$
(**one-of** i_1 ... i_n)	$(P_{i_1} \sqcup \ldots \sqcup P_{i_n})$
slot-constraint SN	$\top$
has-value $C_1 \ldots C_n$	$\sqcap \exists\ \texttt{SN}.\sigma(C_1) \sqcap \ldots \sqcap \exists\texttt{SN}.\sigma(C_n)$
value-type $C_1 \ldots C_n$	$\sqcap \forall\ \texttt{SN}.\sigma(C_1) \sqcap \ldots \sqcap \forall\texttt{SN}.\sigma(C_n)$
max-cardinality n C	$\sqcap \leq n\ \texttt{SN}.\sigma(C)$
min-cardinality n C	$\sqcap \geq n\ \texttt{SN}.\sigma(C)$
cardinality n C	$\sqcap \geq n\ \texttt{SN}.\sigma(C) \sqcap \leq n\texttt{SN}.\sigma(C)$
has-filler d	$\sqcap \exists\ \texttt{SN}.\sigma(\mathrm{d})$
slot-def SN	
subslot-of $\texttt{SN}_1 \ldots \texttt{SN}_\texttt{n}$	$(\texttt{SN} \sqsubseteq \texttt{SN}_1) \ldots (\texttt{SN} \sqsubseteq \texttt{SN}_\texttt{n})$
domain $C_1 \ldots C_n$	$\exists\ \texttt{SN}.\top \sqsubseteq \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
range $C_1 \ldots C_n$	$\top \sqsubseteq \forall\ \texttt{SN}.\sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
inverse RN	$(\texttt{SN}^- \sqsubseteq \texttt{RN})(\texttt{RN}^- \sqsubseteq \texttt{SN})$
properties transitive	$\texttt{SN} \in \texttt{S}_+$
properties symmetric	$(\texttt{SN} \sqsubseteq \texttt{SN}^-)(\texttt{SN}^- \sqsubseteq \texttt{SN})$
properties functional	$\top \sqsubseteq \leq 1\texttt{SN}$
disjoint $C_1\ C_2 \ldots C_n$	$(\sigma(C_1) \sqsubseteq \neg\sigma(C_2))$
covered C **by** $C_1 \ldots C_n$	$\sigma(C) \sqsubseteq \sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n)$
disjoint-covered C **by** $C_1 \ldots C_n$	$(\sigma(C_1) \sqsubseteq \neg\sigma(C_2))$
	$(\sigma(C) \sqsubseteq \sigma(C_1) \sqcup \ldots \sqcup \sigma(C_n))$
equivalent $C\ C_1 \ldots C_n$	$(\sigma(C) = \sigma(C_1)) \ldots (\sigma(C_{n-1}) = \sigma(C_n))$
instance-of $i\ C_1 \ldots C_n$	$P_i \sqsubseteq \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n)$
related SN $i\ j$	$P_i \sqsubseteq \exists\ \texttt{SN}.P_j$

Table 2: Denotational semantics for language definition

sense of structuring the ontology. ONTOLOGY-SLOT and all of their subclasses encode the structure of the Mikrokosmos ontology. They are not mapped as DL classes or slots. Instead they are incorporated into the DL definition of the Mikrokosmos concepts that they refer to.

Mikrokosmos has some information that can not be mapped to a DL language. We will address this problem in two ways. First we will make some annotations to class and slots that are not supported by DL language, but which could be provided by RDFS based languages. Second, extra information about slots that is not supported by DL language will be stored in special concepts created from the corresponding slots.

4.1 Building DL classes

Now we will discuss how we extract information stored in the XML based file to build classes in DL language.

The information that has to be extracted is:

class-def (**primitive** \| **defined**) CN
subclass-of $C_1 \ldots C_n$
slot-constraint$_1$
$\vdots$
slot-constraint$_m$

Having identified the set of DL classes we need to identify their superclasses and *slot-constraints*. Information about superclasses is encoded in XML records of the form shown in Figure 2. Additional sources of information about superclasses —such as RECORDs where CN appears as FILLER and SUBCLASSES appears as SLOT— actually encode redundant information and are therefore discarded.

```
<RECORD>
    <CONCEPT> CN </CONCEPT>
    <SLOT>IS-A</SLOT>
    <FACET>VALUE</FACET>
    <FILLER> Ci </FILLER>
</RECORD>
```

Figure 2: XML encoding of superclass information

Information about *slot-constraints* is encoded in records having PROPERTYs as a slot. But there are also some ONTOLOGY-SLOT used in class definition and we will assign them a representation.

We collect information about *slot-constraints* from XML records of the form shown in Figure 3:

<RECORD>
<CONCEPT> CN </CONCEPT>
<SLOT> SN </SLOT>
<FACET> $FACET$ </FACET>
<FILLER> C </FILLER>
</RECORD>

Figure 3: XML encoding for *slot-constraints*

We obtain different information depending on the value of *FACET*

- If *FACET* = DEFAULT-MEASURE
  CN **slot-constraint** SN **value-type** C is added to the corresponding class definition.
- If *FACET* = DEFAULT. This information is stored as an annotation
- If *FACET* = INV. This information comes from another slot, that it is inverse to SN. There is no need to handle this information here because DL has automatic handling for such type of information.
- If *FACET* = NOT. This entry appears when we restrict inheritance of one SLOT in the hierarchy. Information contained in Mikrokosmos about these is affirmative information and negative information, DL only uses affirmative information to handle it, so we do nothing with this information.
- If *FACET* = RELAXABLE-TO. This information is stored as an annotation
- If *FACET* = SEM
  CN **slot-constraint** SN **value-type** C is added.
- If *FACET* = VALUE
  CN **slot-constraint** SN **has-value** C is added.

Additional information encoded in terms of records with ONTOLOGY-SLOTS —as slots—, must be handled and incorporated into the corresponding class definitions.

The ONTOLOGY-SLOTs to be identified are DEFINITION, SPANISH1 and ENGLISH1.

- If SLOT = DEFINITION. We will make an annotation in class definition.
- If SLOT = SPANISH1 or ENGLISH1. We create two SLOTs called SPANISH1 and ENGLISH1, so we can assert:
  slot-constraint $ENGLISH1$ **has-filler** d. [3]

4.2 Building DL relations

Information required to build DL relations is encoded in XML records with ONTOLOGY-SLOTS in their SLOT field of the form shown in Figure 4

<RECORD>
<CONCEPT> SN </CONCEPT>
<SLOT>*SLOT*</SLOT>
<FACET>*FACET*</FACET>
<FILLER> X </FILLER>
</RECORD>

Figure 4: XML encoding of slot information

Possible relevant fillers of the ONTOLOGY-SLOTS are:

- DEFINITION, IS-A and SUBCLASSES: This information is handled for DL relations in the same way as for DL classes.
- INVERSE: It can be used with SEM and VALUE *FACET* and represents inverse slots.
  slot-def SN **inverses** X is added.
- DOMAIN: As before when there is a restriction in inheritance Mikrokosmos asserts affirmative and negative information so there is a *FACET* NOT that is rejected, and has no translation to DL language. There are more possibilities for filling the FACET: VALUE, DEFAULT, RELAXABLE-TO and SEM, we make no distinction among them:
  slot-def SN **domain disjoint** $X_1 \ldots X_n$ is added.
- RANGE: FACET NOT is treated as above. When we have other FACETs there are two possible kinds of FILLERs: CONCEPTS or numeric ranges. For CONCEPTS
  slot-def SN **range disjoint** $X_1 \ldots X_n$ is added.
  For numeric range we create a subclass of Numeric-Range (See Figure 5 and example in Figure 6).
- MEASURED-IN: This information is considered the same as RANGE. It can only have SEM or DEFAULT FACETs.
  slot-def SN **range** X is added.

[3] These slots encode cross indexing with lexical information. Another possible mapping would have been to add them as instances, but this would result in loss of this cross indexing information.

```
class-def primitive Numeric-Range
    slot-constraint Left-Range-Margin
        max-cardinality 1 int
    slot-constraint Right-Range-Margin
        max-cardinality 1 int

slot-def Numeric-Left-Margin
    range int

slot-def Numeric-Right-Margin
    range int

class-def defined Numeric-Right-Range
    subclass-of Numeric-Range
    slot-constraint Right-Range-Margin
        min-cardinality 1 int

class-def defined Numeric-Left-Range
    subclass-of Numeric-Range
    slot-constraint Left-Range-Margin
        min-cardinality 1 int

class-def defined Numeric-Closed-Range
    subclass-of Numeric-Right-Range
    subclass-of Numeric-Left-Range
```

Figure 5: Range definitions

```
<RECORD>
    <concept>VISCOSITY</concept>
    <slot>RANGE</slot>
    <facet>SEM</facet>
    <filler>(<;>;0 1)</filler>
    <uid>256</uid>
</RECORD>

class-def VISCOSITY
    subclass-of Numeric-Range
    slot-constraint Left-Range-Margin
        has-filler 0
    slot-constraint Right-Range-Margin
        has-filler 1
```

Figure 6: Example of range restriction

4.3 Building Mikrokosmos PROPERTYs as DL classes

As we have seen in previous subsection, not all information about PROPERTYs can be mapped easily to slots. Because of that we have decided to include an extra hierarchy of concepts created from PROPERTYs.

For each slot we create a class that inherits from CLASS-SLOT called CLASS-<*PROPERTY-NAME*>. These classes contain all information about the PROPERTYs that we could not represent in a DL relation.

For each SLOT applied to a CONCEPT we will create a class that inherits from CLASS-SLOT-CONCEPT called CLASS-<*PROPERTY-NAME*>-<*CONCEPT-NAME*>. These classes have slot-constraints in order to define information not captured in the respective concept.

With this structure of classes we do not lose any information about slots and slot-constraints but almost all information stored in that way is not useful for reasoning in current tools like RACER (Haarslev and Moller, 2001).

5 Evaluation of the translation process

DL provide the way to carry out complex inference and reasoning tasks. In order to achieve this goal our DL language is less expressive than Mikrokosmos. Among all restrictions in the expressiveness of DL languages we will mention two. DL languages are not able to reason with default values for the restrictions. And they do not manage finite domains such as enumerates or sets.

These differences in expressivity between Mikrokosmos and our DL language has as a result some interesting points in the translation process. There were two possible solutions to this problem. First one was to discard all information that has not a direct mapping to our DL language. And second one —which we have chosen— is to make some artifices in order to preserve all information, but obviously we cannot reason with this information.

There are two places where we have made this kind of artifices:

- Default values: Mikrokosmos is able of managing default values for restrictions while DL is not. So we have decided to store it as an annotation.
- Numeric restrictions: For example in Mikrokosmos we can restrict the age of a person to be plus than 0 and minus that 120, but our DL language is not capable. Because of that we have created the complex structure of *Numeric-Range* concepts.

So we can say that we have no loss of information in the translation process. But we were incapable to use all information contained in Mikrokosmos for reasoning and inference tasks.

6 Applications of Mikrokosmos and future work

One of the distinguishing features of the original Mikrokosmos resources for machine translation was the explicit isolation between the pseudo-taxonomical structure used to represent the concepts on one hand, and the particular lexical information

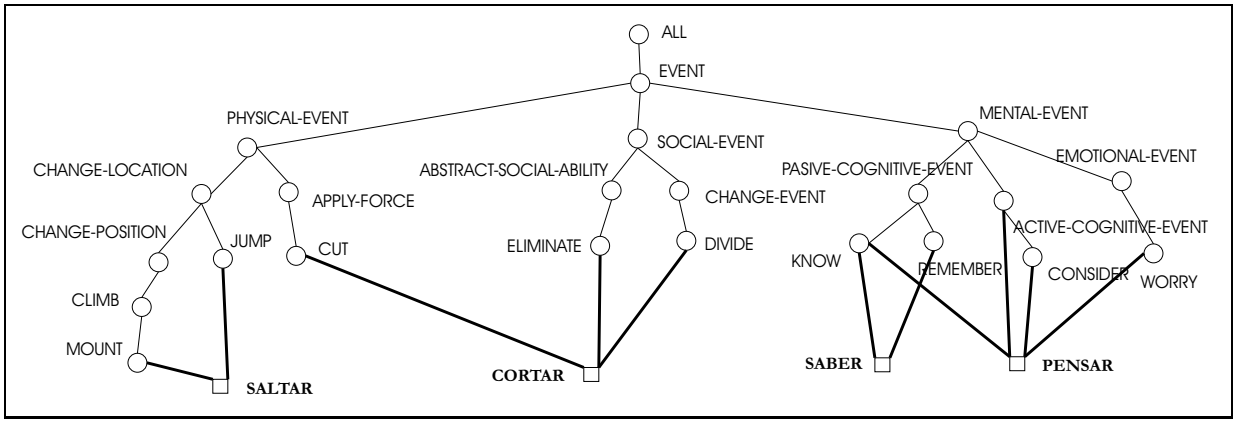

Figure 7: Mikrokosmos ontology with some instances

that was associated with those concepts for realization in a particular language. This peculiarity allowed relatively easy bidirectional translation between different languages via an intermediate conceptual representation.

Subsequent uses and/or transformations of these resources must take into account this peculiarity. In our case, the work carried out so far in transporting the Mikrokosmos ontology to OWL has been restricted to the part of the ontology concerned with the conceptual representation. Although this transformation already opens up avenues of research for knowledge representation for problem solving (Díaz-Agudo and González-Calero, 2002), the number of useful applications of the results of this process in the field of natural language processing will increase greatly once the corresponding lexicons —there are currently versions in Spanish and English— are also transformed into OWL.

For instance, use of this resource provides the means for intelligently substituting a given word for a different one - as required for example in our poetry generation system (Diaz-Agudo et al., 2002) during the adaptation of the structure of poem from the case base to obtain a verse approximation of a user query. Assuming that a structure such as:

> Sabed que en mi perfecta edad y armado
> con mis ojos abiertos me he rendido
> al niño que sabéis ciego y desnudo.

needs to be adapted, and the adaptation requires the substitution of the verb "sabed" for a related one out of a list of candidates - possibly obtained from the given user query - such as "pensad", "cortad" and "saltad". By conuslting the structure of the ontology —see extract in Figure 7 for illustration— the system may correctly select "pensad" as a preferable candidate in view of its proximity in the ontology to the original word.

Our future lines of research in this field will focus in a deeper study of which concepts are primitive and which ones are defined. Now we have decided that all concepts having any restriction are defined concepts. This decission was taken in orden to reason with the ontology but it is necessary to examine it in detail.

7 Conclusions

Mikrokosmos ontology is a rich and extensive knowledge resource that was developed with a proprietary formalism, with a weak theoretical foundation. We have analysed the contents of the ontology which have lead us to propose a possible translation into description logics.

All this effort of understanding Mikrokosmos ontology and mapping it to a description logics language has resulted in a concrete implementation. We have chosen OWL —an RDFS based language—, in its version idOWL DL. This version implements reasoning using JENA (McBride, 2001) and the DIG interface (Bechhofer et al., 2003). There are two inference engines that implement the DIG interface: RACER and FaCT[4]. As part of this implementation we have developed an *import plugin* for Protégé 2.0 (See Figure 7).

With this work we can profit from all the knowledge stored in the Mikrokosmos ontology for other tasks related to Artificial Intelligence. These tasks are Natural Language Processing and Knowledge-Intensive Case-Based Reasoning. We still have to translate the Mikrokosmos lexicon in order to fully exploit the original resource.

Acknowledgements

The first author is supported by a FPI Predoctoral Grant form Universidad Complutense, Madrid. The

[4]http://dl-web.man.ac.uk/dig/

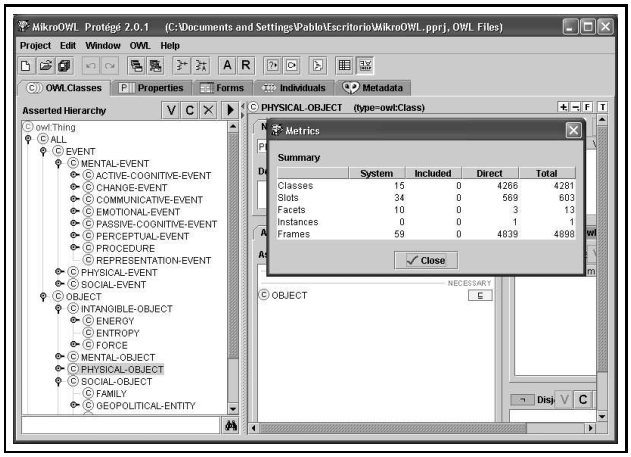

Figure 8: Screen capture of Protégé 2.0 with Mikrokosmos ontology.

work was partially funded by the Spanish Committee of Science & Technology (TIC2002-01961).

References

Sean Bechhofer, Ralf Moller, and Peter Crowther. 2003. The DIG description logic interface. In *Description Logics 2003*, CEUR Workshop Proceedings.

Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Andrea Stein. 2004. Owl web ontology language reference, February. W3C http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Mercedes García de Quesada. 2001. Estructura definicional terminogrfica en el subdominio de la oncologa clínica. In *Estudios de Linguistica Española*, volume 14.

Belén Díaz-Agudo and Pedro Antonio González-Calero. 2002. CBROnto: a task/method ontology for CBR. In S. Haller and G. Simmons, editors, *Procs. of the 15th International FLAIRS'02 Conference (Special Track on CBR)*, pages 101–106. AAAI Press.

B. Diaz-Agudo, P. Gervás, and P A. Gonzalez-Calero. 2002. Poetry generation in COLIBRI. In S. Craw and A. Preece, editors, *ECCBR 2002, Advances in Case Based Reasoning*, pages 73–102. Springer. Lecture Notes in Artificial Intelligence.

Volker Haarslev and Ralf Moller. 2001. Description of the RACER system and its applications. In *Proceedubgs International Workshop on Description Logics (DL-2001)*, pages 132–142, Stanford, USA.

Volker Haarslev and Ralf Moller, 2003. *RACER User s Guide and Reference Manual Version 1.7.7*. Concordia University and Univ. of Appl. Sciences in Wedel, November. http://www.sts.tu-harburg.de/∼r.f.moeller/racer/racer-manual-1-7-7.pdf.

I. Horrocks, U. Sattler, and S. Tobies. 2000. Reasoning with individuals for the description logic SHIQ. In David MacAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, number 1831, pages 482–496, Germany. Springer Verlag.

Atanas K. Kiryakov, Marin Dimitrov, and Kiril Iv. Simov. 2001. Ontomap - the guide to the upper-level. In *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001*. http://www.ontotext.com/publications/swws01.pdf.

E. Lonergan. 2001. Lexical knowledge engineering: Mikrokosmos revisited. In *PACLING2001 - Pacific Association for Computational Linguistics 2001*, Kitakyushu, Japan.

C. Lutz. 2003. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. King's College Publications.

Brian McBride. 2001. Jena: Implementing the rdf model and syntax specification. In *Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001. Hongkong, China, May 1, 2001*. http://SunSITE.Informatik.RWTH-Aachen.de/Publications/CEUR-WS/Vol-40/mcbride.pdf.

Antonio Moreno-Ortiz, Victor Raskin, and Sergei Nirenburg. 2002. New developments in ontological semantics. In *Proceedings of LREC-02, Spain, June.*

Sergei Nirenburg and Victor Raskin, 2004. *Ontological Semantics*, chapter 7, pages 159–207. The MIT Press, September.

S. Nirenburg. 1987. Knowledge-based machine translation, the cmu approach. In *Machine Translation: theoretical and methodological issues, Studies in Natural Language Processing*, pages 68–69, Cambridge. Cambridge University Press.

Klaus Schild. 1991. A correspondence theory for terminological logics: preliminary report. In *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*, pages 466–471, Sidney, AU.