DL Meet FL: A Bidirectional Mapping between Ontologies and Linguistic Knowledge*

Hans-Ulrich Krieger and **Ulrich Schäfer**
Language Technology Lab
German Research Center for Artificial Intelligence (DFKI)
{krieger|ulrich.schaefer}@dfki.de

Abstract

We present a transformation scheme that mediates between description logics (DL) or RDF-encoded ontologies and type hierarchies in feature logics (FL). The *DL-to-FL* direction is illustrated by an implemented offline procedure that maps ontologies with large, dynamically maintained instance data to named entity (NE) and information extraction (IE) resources encoded in typed feature structures. The *FL-to-DL* translation is exemplified by a (currently manual) translation of so-called MRS (Minimal Recursion Semantics) representations into OWL instances that are based on OWL classes, generated from the the type hierarchy of a deep linguistic grammar. The paper will identify parts of knowledge which can be translated from one formalism into the other without loosing information and parts which can only be approximated. The work described here is important for the Semantic Web to become a reality, since semantic annotations of natural language documents (DL) can be automatically generated by shallow and deep natural language parsing systems (FL).

1 Introduction and motivation

Ontologies on the one hand and resources for natural language processing (lingware) on the other hand, though closely related, are often maintained independently, thus constituting a duplication of work.

In the *first* part of this paper, we describe an implemented offline procedure that can be used to map concepts and instance information from ontologies to lingware resources for named entity recognition and information extraction systems. The approach (i) improves NE/IE precision and recall in closed domains, (ii) exploits linguistic knowledge for identifying ontology instances in texts more robustly, (iii) gives full access to ontology instances and concepts in natural language processing results, and (iv) avoids duplication of work in development and maintenance of ontologies and lingware. The advantages of this approach for Semantic Web and natural language (NL) processing-based applications come from a *cross-fertilization* effect. While ontology instance data can improve precision and recall of, e.g., named entity recognition (NER) and information extraction (IE) in closed domains, linguistic knowledge contained in NER and IE components can help to recognize ontology instances (or concepts) occurring in text, e.g., by taking into account inflection, anaphora, and context. Furthermore, (Haghighi and Klein, 2009) and others have shown that incorporating finer-grained semantic information on entities occurring in text (e.g., for antecedent filtering) helps to improve performance of coreference resolution systems.

If both resources would be managed jointly at a single place (in the ontology), they could be easily kept up-to-date and in sync, and their maintenance would be less time-consuming. When ontology concepts and instances are recognized in text, their name or ID can be used by applications to support subsequent queries, navigation, or inference in the ontology using an ontology query language (e.g., SPARQL). The procedure we describe here, preserves hierarchical concept information and links to ontology concepts and instances. Applications are, e.g., hybrid deep-shallow question answering (Frank et al., 2007), automatic typed hyperlinking (Busemann et al., 2003) of instances and concepts occurring in documents, or other innovative applications that combine Semantic Web and NL processing technologies, e.g., for semantic search (Schäfer et al., 2008).

The *second* part of this paper outlines the inverse transformation from feature logics (FL) into description logics (DL). Walking along this direction has the big advantage of potentially applying subsequent description logic reasoners to the lexical semantics of natural language input text in order to infer new knowledge, e.g., in interactive natural language ques-

*The work described in this paper has been carried out in the TAKE project (Technologies for Advanced Knowledge Extraction), funded by the German Federal Ministry of Education and Research under contract number 01IW08003.

Coling 2010: Poster Volume, pages 588–596,
Beijing, August 2010

tion answering. As an example, we will carefully develop the (approximate) translation of so-called robust minimal recursion semantic (RMRS) structures (Copestake, 2003) into OWL descriptions (McGuinness and van Harmelen, 2004). RMRS structures are the semantic output of various NL processing engines, encoded in typed feature structures (TFS). Since NL processors (e.g., taggers, chunkers, deep parsers) only build up structure, subsequent processing steps are either not realized or implemented in ad hoc way,

- dealing with merging & normalization of RMRS,
- infering new knowledge (e.g., w.r.t. the foregoing dialog),
- taking into account extralinguistic knowledge for reasoning.

Now, by moving from a specialized "designer language" (RMRS) to OWL, we can take advantage of years of solid theoretical and practical work in logic, especially in description logics. Since OWL is an instance of the description logics family and the defacto language for the Semantic Web, we can utilize the built-in reasoning capabilities of OWL and (rule-based) description logic reasoners.

The structure of this paper is as follows. In the next section, we outline the relationship between description logics and feature logics, trying to make clear what they have in common, but at the same time explaining their differences. Section 3 describes the syntactic mapping process from the ontology to feature structure descriptions. In Section 4, we present an example where recognized named entities enriched with ontology information are used in hybrid NL processing and subsequent applications. After that, Section 5 explains the mapping of RMRS structures into OWL descriptions. Finally, Section 6 shows that a subsequent description logic reasoner can utilize these descriptions to infer new knowledge.

2 The relationship between description and feature logics

Description logics (DL) (Baader et al., 2003) and *feature logics* (FL) (Carpenter, 1992) have been pursued independently for quite a while. Their close relationship was recognized by (Nebel and Smolka, 1990). Instances of both families of knowledge representation formalisms are usually decidable two-variable fragments of first-order predicate logic. Even though DL dialects usually have an intractable worst-case complexity, average-case reasoning is usually fast, due to the availability of highly-optimized tableaux reasoners. When adding seemingly easy constructs such as "role-value maps" (the analog to reentrancies), the underlying logical calculus becomes undecidable.

From an abstract viewpoint, both DL and FL employ unary and binary predicates for which the two communities invented different names (we only list some of them):

arity	description logic	feature logic
unary	concept, class	type, category
binary	role, property	feature, attribute

Though these names are different, both representation families (usually) vary in further, not so subtle details:

description logic	feature logic
open world assumption	closed world assumption
full Boolean concept logic	only conjunctions
relational properties	functional properties
role-value maps forbidden	reentrancies allowed

Let us be more verbose here to see the descriptional consequences of both approaches in terms of a mutual translation. We note here that we take OWL (McGuinness and van Harmelen, 2004) as an instance of DL and $\mathcal{TDL}$ (type description language) (Krieger and Schäfer, 1994) as an example of FL. OWL, the outcome of the DAML+OIL standardization, is regarded to be the de-facto language for the Semantic Web. OWL still makes use of constructs from RDF and RDFS, but restricts the expressive power of RDFS, thereby ensuring decidability of the standard inference problems. Compared to RDF(S), OWL provides more fine-grained modelling constructs, such as `intersectionOf` or `unionOf`.

Within the Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) paradigm in modern computational linguistics (CL), $\mathcal{TDL}$ is a language that has been employed in various implemented systems, such as PAGE, LKB, PET, or *SProUT*.

Before going into the details of our approximate transformation schema, let us quickly explain how to *atomize* a typed feature structure (TFS) in terms of description logic primitives, using OWL. Consider the following TFS which is a gross simplification of the *Head-Feature Principle* in HPSG. In terms of the "one-dimensional" line-based $\mathcal{TDL}$ notation, we write

```
phrase1 := phrase &
  [HEAD #h1, HEAD-DTR|HEAD #h1],
```

or as a two-dimensional AVM (attribute-value matrix) notation, we have

$$\text{phrase1} \equiv \left[\begin{array}{l} \textit{phrase} \\ \text{HEAD } \boxed{\text{h1}} \\ \text{HEAD-DTR|HEAD } \boxed{\text{h1}} \end{array}\right]$$

Assuming that this is an individual of class *phrase*, we can obtain a meaning-preserving OWL representation (we assume that HEAD and HEAD-DTR are functional OWL object properties):

```
<owl:Thing rdf:ID="h1"/>

<rdf:Description rdf:about="hdtr1">
  <rdf:type rdf:resource="owl:Thing"/>
  <HEAD rdf:resource="h1"/>
</rdf:Description>

<rdf:Description rdf:about="phr1">
  <rdf:type rdf:resource="phrase"/>
  <HEAD rdf:resource="h1"/>
  <HEAD-DTR rdf:resource="hdtr1"/>
</rdf:Description>
```

Note that only the top-level structure is explicitly typed (*phrase*); every other substructure thus is assigned the most general type, which translates into the OWL class `owl:Thing`. Note also the sharing of information under paths `HEAD` and `HEAD-DTR|HEAD`—this is realized by refering to the the name `h1` in the above RDF/OWL description for `phr1` and `hdtr1`.

Given a set of OWL descriptions, obtaining the inverse direction from DL to FL should now be clear. It is important here to group statements that are related to a specific class, viz., inheritance information (e.g., `intersectionOf`) together with property information about roles that are "introduced" on a given class (as given by the value of `rdfs:domain`). In Section 3, we focus on this inverse direction (DL-to-FL), whereas Section 5 exemplifies the FL-to-DL direction.

Let us finally elaborate fundamental differences between the DL and FL families that can only be approximated in terms of "less expressive" constructs.

Open vs. closed world assumption. Typed feature logics usually "live" in a closed world, meaning that if two types t_1 and t_2 do not share a common subtype (having a greatest lower bound), the unification (conjuntion) is assumed to be the bottom type (OWL: `owl:Nothing`), meaning that no individual exists which is of both t_1 and t_2 at the same time. This is totally different to the DL point of view: *what can not proven to be true* (whether the conjunction of t_1 and t_2 denotes the empty set) *is not believed to be false*. Thus we either have to introduce a new type t on the FL side, abbreviating the conjunction of t_1 and t_2 ($\mathcal{TDL}$: t := t_1 & t_2.), or to close the subclass hierarchy on the DL side: $\bot \equiv t_1 \sqcap t_2$ (OWL: `disjointWith`). This decision clearly depends on the direction of the transformation.

Boolean vs. conjunctive description logic. Typed feature logics rarely provide more than *conjunctions* of feature-value constraints. This is due to the fact that disjunctive descriptions render almost linear (conjunctive) unification exponential. A full Boolean calculus, such as OWL DL, even has an NEXPTIME complexity. Thus it is clear that the direction from DL to FL can only be approximated. The inverse direction is clearly trivial with the notable exception of *reentrancies* (see below).

To flesh out our point, consider the DL axiom *human* $\equiv$ *man* $\sqcup$ *woman* that fully determines ($\equiv$) *human* in terms of the union of the concepts *man* and *woman*. Given the syntax of $\mathcal{TDL}$, we can approximate parts of the intended meaning of the description by `man :< human` and `woman :< human`, since the above DL axiom entails that *man* $\sqsubseteq$ *human* and *woman* $\sqsubseteq$ *human* is the case. This is exactly specified by the above two $\mathcal{TDL}$ type definitions. Further, not so trivial approximations can be found in (Flickinger, 2002). The idea here is that foreseeable disjunctions of DL concepts can be emulated by introducing additional FL types (in the worst case, exponentially-many new types, however). Even negated concepts can be simulated this way, since FL lives in a closed world (see above).

Relational vs. functional properties. By default, roles in DL are relational properties, meaning that for a fixed individual in the domain of a given role, the number of individuals in the range needs not to be 0 or 1. DL further allows to impose *cardinality (or number) restrictions* on roles, so that we might write ≥ 0 *livingParents* $\sqcap \leq 2$ *livingParents* which says that one can have at least 0 and at most 2 living parents. This is in sharp contrast to FL which usually assume functional roles (so-called features), making such roles essentially partial functions. A partial workaround has been proposed in CL systems by using (ordered) difference lists to collect information. Other systems, such as *SProUT* (Krieger et al., 2004), come up with bags (or multisets) that even violate the foundational axiom (a set must not contain itself) in order to achieve runtime efficiency.

Summarizing, the FL-to-DL direction of translating features into roles is easy, since features in FL can be easily defined as functional roles in DL (OWL even provides the `owl:FunctionalProperty` characteristics). The inverse direction is only a gross approximation in that cardinality constraints can not be

stated on the FL side.

Role-value maps & reentrancies. The above Head-Feature Principle example seems to indicate that role-value maps can be easily represented in DL, simply by using the name of an individual to specify identity. In fact, this is true, but only for the ABox of a knowledge base, i.e., only for the set of individuals (or instances). However, the notion of role-value maps in DL or reentrencies in FL refers to the TBox and the set of concept definitions, resp. Thus, one can not intensionally specify identity of information for a potentially infinite number of individuals via a class axiom in DL, but needs to extensionally specify identity of information for each individual in the ABox.

3 OntoNERdIE: from OWL to $\mathcal{TDL}$

In this section, we describe an instantiation of the DL-to-FL mapping. OntoNERdIE is an offline procedure that maps ontology concept and instance information to lingware resources (Schäfer, 2006). The approach has been implemented for the language technology ontology that backs up the LT World web portal (http://www.lt-world.org), but can be easily adapted to other domains and ontologies, since it is fully automated, except for the choice of relevant main concepts and properties that are going to be mapped which is a matter of configuration.

The target named entity recognition and information extraction tool we employ here is *SProUT* (Drożdżyński et al., 2004), a shallow multilingual, multi-purpose NL processor. The advantage of *SProUT* in the described approach for named entity recognition and information extraction is that it comes with (1) a type system and typed feature structures as the basic data type, (2) a powerful, declarative rule mechanism with regular expressions over typed feature structures, and (3) a highly efficient gazetteer module with fine-grained, customizable classification of recognized entities.

SProUT provides additional modules such as morphology or a reference resolver that can be exploited in the rule system, e.g., to use context or morphological variation for improved NER. Through automatically generated mappings, *SProUT* output enriched with ontology information can be used for robust, hybrid deep-shallow parsing, and semantic analysis.

In this section, we describe the offline processing steps of the OntoNERdIE approach. The online part in applications is described in Section 4. The approach heavily relies on XSLT transformations (Clark, 1999) of the XML representation formats, both in the offline mapping and in the online application.

3.1 RDF preprocessing

Input to the mapping procedure is an OWL ontology file, containing both concept and instance descriptions. The RDF file is pre-processed with a generic XSLT stylesheet sorting and merging `rdf:Description`s that are distributed over the file but which belong together. We use XSLT's `key` and `generate-id` functions. Depending on the application, the next two processing stages take a list of concepts as filter because it will typically not be desirable to extract all concepts or instances available in the ontology. In both cases, resource files are generated as output that can be used to extend existing named entity recognition resources. E.g., while general rules can recognize domain-independent named entities (e.g., any person name), the extended resource contains specific, and potentially more detailed information for domain-specific entities.

3.2 Extracting inheritance

The second stylesheet converts RDFS `subClassOf` statements from output step 1 (Section 3.1) into a set of $\mathcal{TDL}$ type definitions that can be immediately imported by the *SProUT* named entity recognition grammar. Currently 1,260 type definitions for the same number of `subClassOf` statements in the LT World ontology are generated, e.g.,

```
NL_Parsing := Written_Language &
  Language_Analysis.
```

This is of course a lossy conversion because not all relations supported in an OWL ontology (such as `unionOf`, `disjointWith`, `intersectionOf`) are mapped. However, we think that for NE classifications, the `subClassOf` taxonomy mappings will be sufficient. Other relations could be formulated as direct (though slower) ontology queries using the `OBJID` mechanism described in the next step. If the target of OntoNERdIE is a NER system different from *SProUT* and without a type hierarchy, this step can be omitted. The `subClassOf` information can always be gained by querying the ontology appropriately on the basis of the concept name.

3.3 Generating gazetteer entries

The next stylesheet selects statements about instances of relevant concepts via the `rdf:type` information and converts them to structured gazetteer source files for the *SProUT* gazetteer compiler (or into a different format in case of another NER system). In the following example, one of the approximately 20,000 converted entries for LT World is shown.

```
Bernd Kiefer | GTYPE: lt_person |
  SNAME: "Kiefer" | GNAME: "Bernd" |
  CONCEPT: Active_Person |
  OBJID: "obj_62893"
```

The attribute CONCEPT contains a $\mathcal{TDL}$ type generated in step 2 (described in Section 3.2). For convenience, several ontology concepts are mapped (defined manually as part of the configuration of the stylesheet) to only a few named entity classes (under attribute GTYPE). For the LT World ontology, these classes are person, organization, event, project, product, and technology. The advantage of this simplification is that NER context rules from existing *SProUT* named entity grammars can be re-used for improved robustness and disambiguation.

The rules, e.g., recognize name variants with title like Prof. Kiefer, Dr. Kiefer, or Mr. Kiefer with or without a first name. Moreover, context (e.g., prepositions with location names, verbs), morphology and reference resolution information can be exploited in these rules.

The following *SProUT* rule lt-event (extended $\mathcal{TDL}$ syntax) simply copies the slots of a matched gazetteer entry for events (e.g., a conference) to the output as a recognized named entity.

```
lt-event :> gazetteer &
  [GTYPE lt_event, SURFACE #name,
   CONCEPT #concept, OBJID #objid,
   GABBID #abbrev]
  ->
  ne-event & [EVENTNAME #name,
    CONCEPT #concept, OBJID #objid,
    GABBID #abbrev].
```

OBJID contains the object identifier of the instance in the ontology. It can be used as a link back to the full knowledge stored in the ontology, e.g., for subsequent queries, like *Who else participated in project* [*with OBJID obj_4789*]*?*.

In case multiple instances with same names but different object IDs occur in the ontology (which actually happens to be the case in LT World), multiple alternatives are generated as output which is probably the expected and desired behavior (e.g., for frequent names such as John Smith). On the other hand, if product or event names with an abbreviated variant exist in the ontology, they both point to the same object ID (provided they are stored appropriately in the ontology).

4 Application to hybrid deep-shallow parsing

We now describe and exemplify how the named entities enriched with ontology information are employed in a robust, hybrid deep-shallow architecture, combining domain-specific shallow named entity recognition with deep, broad-coverage, domain-independent, unification-based parsing for generating a semantic representation of the meaning of parsed sentences. An application of this scenario is deep question analysis for question answering of structured knowledge sources, encoded as an OWL ontology (Frank et al., 2007).

The output of *SProUT* for a recognized named entity is a typed feature structure in XML containing the instantiated RHS of the recognition rule as shown in step 3 (Section 3.3) with the copied structured gazetteer data, plus some additional information like character span, named entity type, etc. The mapping of recognized named entities to generic lexicon entries of the deep grammar, in this case the English Resource Grammar (Flickinger, 2002), for hybrid processing are performed through an XSLT stylesheet, automatically generated from the *SProUT* type hierarchy. Analogous mappings are supported for other grammars available in the DELPH-IN repository (see http://www.delph-in.net). The mapping basically transports the surface string, a character span, and a generic lexicon type of the deep grammar for a chart item to be generated in an XML format, readable by the deep parser. A sample output of the semantic representation generated by the deep parser is shown in Figure 1. The semantic representation format, called RMRS, is described in (Copestake, 2003) and in Section 5.3 below.

In addition to the basic named entity type mapping for default lexicon entries, the recognized concepts are also useful for constraining the semantic sort in the deep grammar in a more fine-grained way (e.g., for disambiguation). The deep parser's XML input chart format foresees "injection" of such types into deep structures. Here, OBJID and other structured information, like given name and surname, can be preserved in the representation. The advantage of the RMRS format is that it can also be combined *ex post* with analyses from other deep or shallow NLP components, e.g., with partial analyses when a full parse fails.

5 (R)MRS2OWL: from $\mathcal{TDL}$ to OWL

This section is devoted to the translation of MRSs which are encoded as TFSs into a set of OWL expressions. An example of a variant of MRS, a so-called robust MRS (RMRS) has already been depicted in Figure 1. RMRS will be explained in more detail in Section 5.3.

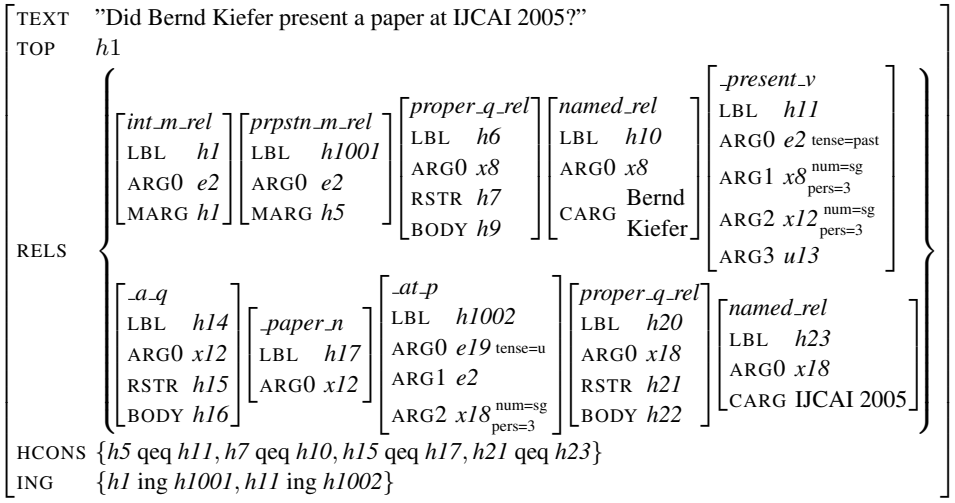

Figure 1: RMRS generated through hybrid parsing.

5.1 Some words on MRSs

There exist good linguistic reasons for assuming that the semantics of a sentence like *Kim ate a cookie* is not $past(eat(kim', cookie'))$, but instead something like $\exists e \ . \ eating(e) \wedge subject(e, kim') \wedge object(e, cookie') \wedge before(e, now)$. This approach to NL semantics is often called *Event* or *Davidsonian* semantics (named after the American philosopher Donald Davidson). HPSG has incorporated ideas from event semantics by defining so-called Minimal Recursion Semantics (MRS) structures (Copestake et al., 2005) that are constructed in parallel with the syntactic structure. MRS as such provides a flat compositional semantics and maximizes splitting using equality constraints. Structural ambiguities, as can be found in the famous sentence *Every farmer who owns a donkey beats it*, are not spelled out, but instead quantifier scope is underspecified. By imposing constraints on the scope, specific analysis trees can be reconstructed. Robust MRS (RMRS) (Copestake, 2003), derived from MRS, was designed as an abstract language that supports the integration of partial and total analysis results from deep and shallow processors and provides a good tradeoff between robustness and accuracy (see (Frank et al., 2004) for an example).

5.2 Why the translation is useful

NL processors (e.g., tokenizer, POS tagger, shallow chunk parser, deep parser, etc.) that are geared towards (R)MRSs (or another common language) have the potential of combining their output on the level of semantics. However, these engines do *not* provide any form of reasoning, i.e., they only build up structure.

Consider, for instance, a deep unification-based parser that might return analyses represented as typed feature structures, where both syntax and semantics (the MRS) has been constructed with the help of unification. Now, to bring structures together and to perform deductive and abductive forms of reasoning, subsequent computational steps are necessary, but these steps strictly go beyond the power of ordinary parsing.

In order to perform these subsequent steps, we need a concrete implemented (and hopefully standardized) representation language for which editing, displaying, and reasoning tools are available. Exactly OWL accomplishes these requirements. Hence we think that the described below translation process from (R)MRSs into OWL is worthwhile, especially when one is interested in interfacing linguistic knowledge (the (R)MRSs) with extralinguistic ontologies for specific domains.

5.3 The translation process

In order to explain the translation process, we will analyze the RMRS depicted in Figure 1. The RMRS was derived from the MRS of the deep unification-based parser. We see that an RMRS contains four distinguished attributes (the `TEXT` attribute is only added for illustration):

1. `TOP`: a handle (pointer) to the top-level structure.

2. `RELS` (relations): a set of so-called *elementary predications* (EP), encoded as TFSs, each expressing an atomic semantic unit that can not be

further decomposed; due to the lack of sets, TFS grammars use a list here.

3. HCONS (handle constraints): a set of so-called *qeq constraints* (equality modulo quantifiers); the left side of a qeq constraints (a handle h in an argument position) is always related to a label l of an EP, (i) either directly ($h = l$) or (ii) indirectly, in case h dominates a quantifier q, such that $\texttt{BODY}(q) = l$ or again another quantifier, where condition (ii) is recursively applied again.

4. ING (in group): a set of relations used to express a conjunction of EPs from the set RELS.

Giving this information, it should now be clear that the TFS from Figure 1 must be realized as an instance of the OWL class RMRS and that the features TOP and RELS must be implemented as roles in OWL, all defined on RMRS through the use of rdfs:domain:

```
<owl:Class rdf:ID="RMRS"/>
<owl:ObjectProperty rdf:ID="TOP">
  <rdf:type rdf:resource=
     "&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#RMRS"/>
  <rdfs:range rdf:resource=
     "#HandleVar"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="RELS">
  <rdfs:domain rdf:resource="#RMRS"/>
  <rdfs:range rdf:resource="#EP"/>
</owl:ObjectProperty>
```

TOP takes exactly one argument, hence we use OWL's FunctionalProperty characteristics mechanism here. Since RELS (as well as HCONS and ING, see below) might take more than one argument, we do not impose a property restriction here, so they are relational by default. TOP maps to a special variable class (see below), and RELS to EPs.

TOP. The TOP property always takes a handle variable; other variable classes, such as label vars are used for restricting properties:

```
<owl:Class rdf:ID="Var"/>
<owl:Class rdf:ID="HandleVar">
  <rdfs:subClassOf rdf:resource=
     "#Var"/>
</owl:Class>
<owl:Class rdf:ID="LabelVar">
  <rdfs:subClassOf rdf:resource=
     "#Var"/>
</owl:Class>
```

Actually, this modelling is mere window-dressing and clearly verbose, since an OWL instance of class RMRS is always assigned a name (<RMRS rdf:ID="...">), and in fact, this name can be taken to be the TOP handle. This means that we can in principle forbear from the TOP property. However, if we want to utilize morpho-syntactical information in subsequent inference steps, we have to enrich the above variable classes with further properties/roles, such as tense, pers, or num (see, e.g., the "structured" variables in the structure for _present_v in Figure 1).

RELS. Elements of RELS, i.e., concrete EPs are essentially "slimed" instances of feature structure types. Overall, this means that we have to represent the relevant types of the linguistic type hierarchy and their subsumption relationship as OWL classes. As shown in Section 3, this process can be automated and only some guidance from a knowledge engineer is necessary to mark the features that should *not* be taken over to the DL side.

HCONS and ING. HCONS essentially specifies a ternary relation, but since OWL (and DL in general) are restricted to unary and binary relations, one way to model a qeq constraint is to define a binary property, consisting of a left-hand and a right-hand side. From what has been said above, the left-hand side is a handle and the right-hand side a label, hence we have the following declaration for qeq:

```
<owl:ObjectProperty rdf:ID="qeq">
  <rdfs:domain rdf:resource=
     "#HandleVar"/>
  <rdfs:range rdf:resource=
     "#LabelVar"/>
</owl:ObjectProperty>
```

Given this way of modelling, it is now *impossible* to define a property HCONS (as well as ING) on class RMRS, since properties can only take instances of classes, but not instances of other *properties*. However, since we assume that our variables (instances of class Var) are always unique at runtime, it is in principle not necessary to group the qeq constraint *inside* an (R)MRS—note that there is still a connection between EPs and qeq constraints through the use of variables. However, if we want to talk about/want to access the qeq constraints of a specific (R)MRS instance directly, this kind of modelling is somewhat unhandy.

To overcome this seemingly wrong representation (we are neutral about this), we have to "reify" or "wrap" qeq property instances. This would mean that qeq would no longer be a property, but instead becomes a class, say QEQ, consisting of a right-hand and a left-hand side. With this in mind, we can easily model, e.g., the first qeq constraint qeq1 from the above figure:

```
<RMRS rdf:ID="rmrs1">
  <TOP  rdf:resource="#h1"/>
  <RELS rdf:resource="#ep1"/>
  <HCONS rdf:resource="#qeq1"/>
```

```
  ...
</RMRS>
<QEQ rdf:ID="qeq1">
  <LHS rdf:resource="#h5"/>
  <RHS rdf:resource="#h11"/>
</QEQ>
<HandleVar rdf:ID="h1"/>
<HandleVar rdf:ID="h5"/>
<LabelVar rdf:ID="h11"/>
<int_m_rel rdf:ID="ep1">
  <LBL rdf:resource="#h1"/>
  <ARG0 rdf:resource="#e2"/>
  <MARG rdf:resource="#h1"/>
</int_m_rel>
```

What we have said about qeq constraints so far do hold for in-group constraints as well.

6 DL reasoning: a small example

We have already said that the OWL representation of RMRS structures are a good starting point to implement some useful forms of reasoning. Consider the sentence *Did Bernd Kiefer present a paper at IJCAI 2005?* from Figure 1. From the resulting EPs and with the help of an in-group constraint, we can infer the fact that Bernd Kiefer was (physically) at IJCAI 2005, assuming he has presented a paper (which he did). The inference rule achieving this can be stated informally as *presenting a paper at a conference entails being at the conference*. A more formal representation in terms of feature structures is given in Figure 2.

Clearly this rule can be rewritten to operatate on OWL expressions (as is proposed in SWRL (Horrocks et al., 2004)) or on the underlying RDF triple notation (which, for instance, OWLIM (Kiryakov, 2006) assumes). Note the use of logical variables in the above rule in order to formulate the transport of information from the LHS to the RHS. The above rule abstract away from concrete persons and locations through the use of logic variables $?p$ and $?l$. Note further that the resulting RHS output structure is no longer a RMRS but a domain-specific representation (somewhat simplified in this example) that can be queried for or can be employed in subsequent reasoning tasks.

In (Frank et al., 2007), an implemented approach is described that utilizes an *additional* frame representation layer (Ruppenhofer et al., 2006) in which rules of the above kind are applied, using the term rewriting system of (Crouch, 2005).

7 Summary

Our paper returned to mind that there exists a close relationship between feature logics as used in computational linguistics and description logics employed in the Semantic Web community. This relationship can be utilized to obtain more and better semantic annotations through information extraction and deep parsing of text documents. We have indicated that specific language constructs in FL and DL can be mutally transformed without losing any meaning, whereas others can only be approximated (esp., role-value maps/reentrencies and functional features/relational roles).

We have described an implemented procedure that maps ontology instances and concepts to named entity recognition and information extraction resources. As argued in the paper, the benefits for minimized domain-specific and linguistic knowledge engineering are manifold. An application using hybrid shallow and deep NL processing on the basis of the mapped ontology data has been successfully implemented for question answering. This application (Frank et al., 2007) employs an additional frame semantics layer (cf. Section 6) on which light forms of reasoning take place. In order to make this additional layer superflous, we have described a transformation scheme that maps (R)MRS into OWL descriptions. Given these descriptions, rules of the above kind (Section 6) can directly operate on OWL, and no additional translation is necessary to query the instance data, encoded in RDF/OWL.

References

Baader, Franz, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. 2003. *The Description Logic Handbook*. Cambridge University Press, Cambridge.

Busemann, Stephan, Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Hans Uszkoreit, and Feiyu Xu. 2003. Integrating Information Extraction and Automatic Hyperlinking. In *Proceedings of the Interactive Posters/Demonstration at ACL-03*, pages 117–120.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge.

Clark, James, 1999. *XSL Transformations (XSLT)*. W3C, http://w3c.org/TR/xslt.

Copestake, Ann, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(4):281–332, 12. DOI 10.1007/s11168-006-6327-9.

Copestake, Ann. 2003. Report on the Design of RMRS. Technical Report D1.1b, University of Cambridge, Cambridge, UK.

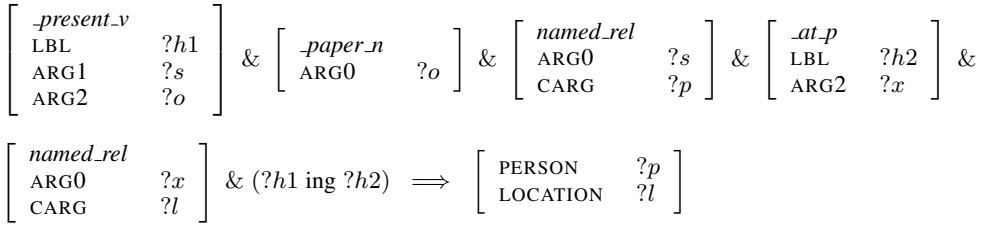

Figure 2: RMRS rule over EPs and in-group constraint.

Crouch, Richard. 2005. Packed rewriting for mapping semantics to KR. In *Proceedings of the International Workshop on Computational Semantics (IWCS) 6, Tilburg*.

Drożdżyński, Witold, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2004. Shallow Processing with Unification and Typed Feature Structures—Foundations and Applications. *KI*, 04(1):17–23.

Flickinger, Dan. 2002. On building a more efficient grammar by exploiting types. In Oepen, S. D. Flickinger, J. Tsuji, and H. Uszkoreit, editors, *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*, pages 1–17. CSLI Publications.

Frank, Anette, Kathrin Spreyer, Witold Drożdżyński, Hans-Ulrich Krieger, and Ulrich Schäfer. 2004. Constraint-Based RMRS Construction from Shallow Grammars. In Müller, Stefan, editor, *Proceedings of the HPSG04 Conference Workshop on Semantics in Grammar Engineering*, pages 393–413. CSLI Publications, Stanford, CA.

Frank, Anette, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, and Ulrich Schäfer. 2007. Question answering from structured knowledge sources. *Journal of Applied Logics, Special Issue on Questions and Answers: Theoretical and Applied Perspectives*, 5(1):20–48.

Haghighi, Aria and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1152–1161.

Horrocks, Ian, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission.

Kiryakov, Atanas. 2006. OWLIM: balancing between scalable repository and light-weight reasoner. Presentation of the Developer's Track of WWW2006.

Krieger, Hans-Ulrich and Ulrich Schäfer. 1994. *TDL*—A Type Description Language for Constraint-Based Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, pages 893–899.

Krieger, Hans-Ulrich, Witold Drożdżyński, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2004. A Bag of Useful Techniques for Unification-Based Finite-State Transducers. In *Proceedings of KONVENS 2004*, pages 105–112.

McGuinness, Deborah L. and Frank van Harmelen. 2004. OWL Web Ontology Language Overview. Technical report, W3C. 10 February.

Nebel, Bernhard and Gert Smolka. 1990. Representation and reasoning with attributive descriptions. In Bläsius, K.-H., U. Hedtstück, and C.-R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, pages 112–139. Springer, Berlin. Also available as IWBS Report 81, IBM Germany, September 1989.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, Chicago.

Ruppenhofer, Josef, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. FrameNet II: extended theory and practice. Technical report, International Computer Science Institute (ICSI), University of California, Berkley. http://framenet.icsi.berkley.edu/book/book.pdf.

Schäfer, Ulrich, Hans Uszkoreit, Christian Federmann, Torsten Marek, and Yajing Zhang. 2008. Extracting and querying relations in scientific papers on language technology. In *Proceedings of LREC-2008*.

Schäfer, Ulrich. 2006. OntoNERdIE – mapping and linking ontologies to named entity recognition and information extraction resources. In *Proceedings of the 5th International Conference on Language Resources and Evaluation LREC-2006*, pages 1756–1761, Genoa, Italy.