Morph Moulder: Teaching Software for HPSG and Description Logics

EKATERINA OVCHINNIKOVA, *University of Osnabrück, Institute of Cognitive Science, E-mail: eovchinn@uni-osnabrueck.de*

FRANK RICHTER, *University of Tübingen, Seminar für Sprachwissenschaft, E-mail: fr@sfs.uni-tuebingen.de*

Abstract

The graphical software Morph Moulder (MoMo) presented here was originally created for teaching the logical foundations of Head-Driven Phrase Structure Grammar (HPSG) in an e-Learning environment. It has then been extended to a treatment of description logics (DL), which are at present the standard formalism for building ontologies. With MoMo, students can construct interpretations of sets of formulae and check whether their interpretations model these formulae (called *theory* in HPSG and *axioms* in DL). MoMo also supports reasoning such as the construction of well-formed interpretations in the feature logic of HPSG and the automatic extraction of subsumption hierarchies in DL. It has been used successfully in several courses on HPSG linguistics, on computational grammar implementation and on the logical foundations of constraint-based grammar frameworks.

Keywords: feature logic, description logics, visualization, reasoning, interactive teaching software, graphical teaching software, ontology, model-theoretic grammar, HPSG

1 Motivation

MoMo was developed as an interactive educational tool for teaching the mathematical foundations of the Head-Driven Phrase Structure Grammar (HPSG, [4]) based on Relational Speciate Re-entrant Language (RSRL, [5]). Novices in constraint-based linguistic theories with little background in mathematical logic often have problems understanding the essential relationship between grammars as sets of logical statements and their model-theoretic interpretation. To overcome these difficulties, the main task of the software was to project the underlying mathematical concepts onto a graphical level, where they could be grasped much more intuitively than in the form of symbolic definitions. Some early inspiration for the design of MoMo came from the introductory logic textbook [3], and in particular from one of its software tools, *Tarski's World*.[1] Encouraged by MoMo's success with students in its first application domain, MoMo was extended to the visualization of description logics (DL, [2]), a widely used family of logics designed for representing ontological knowledge.

Particular emphasis was placed on providing students with hands-on experience constructing interpretations of logical formulae. Central topics are (1) restrictions on interpretations imposed by logical signatures, and (2) properties of the satisfaction relation between logical formulae and objects in interpretations. With interactive

[1] For illuminating observations about the advantages of learning the meaning of logical languages by studying their models, see [3, pp. 13–14].

graphical software, students quickly learn the relevant concepts without having to read a significant number of complex mathematical definitions first, and they see and construct many more examples than they would in traditional classroom lectures. By the time they get to explicit definitions, they have already understood the essential ideas and are in a much better position to appreciate their theoretical explanation. Working with visual representations of the founational logical concepts of linguistic theory or ontologies, students gain a much deeper understanding of the subject and are ultimately much better prepared to analyze problems in the application domains.

We will give a brief overview of MoMo's structure, the graphical interface and the most important features. In Section 2 we discuss the general functions which are available in both modules of the system, the initial RSRL module and the subsequently added DL module; Sections 3 and 4 introduce functions which are specific to RSRL and to description logics, respectively. Section 5 illuminates the relationship between the two logical systems implemented in MoMo. Section 6 concludes our overview with general observations on the use of the software and on future developments.

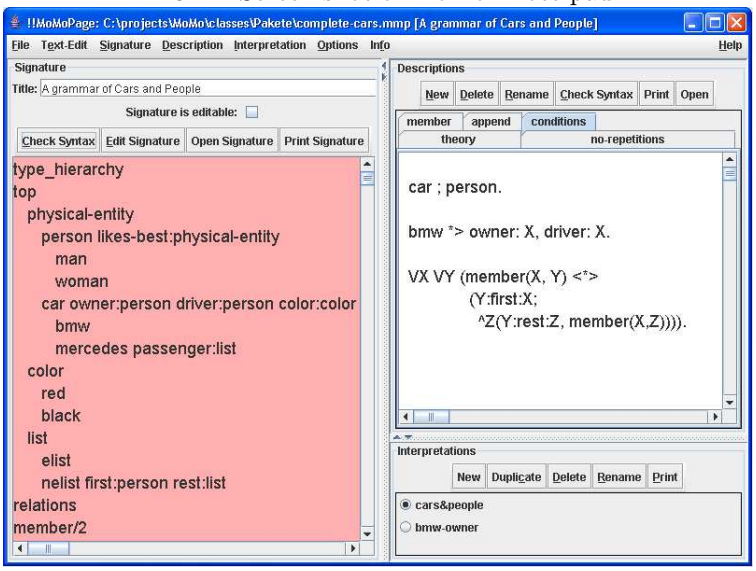
2 MoMo: Visualization and Reasoning

MoMo's most important functions visualize the relationships between sets of logical formulae and their model-theoretic interpretations, and support reasoning about interpretations of logical statements. Here we will not present definitions of the formal languages of MoMo and of the denotations of their expressions. Instead, we will use examples as they are displayed in MoMo to explain the formalisms. Essential to our discussion will be the fact that our logical systems comprise logical formulae and an interpretation function defining the denotation of every logical formula as a set of objects. In MoMo, users can (1a) declare arbitrary sets of basic non-logical syntactic symbols and organize them into a subsumption hierarchy; (1b) write sets of logical formulae; and (2) construct interpretations of a given set of formulae as labeled (possibly cyclic) directed graphs. The fundamental division between the declaration and use of syntactic symbols (1a-b) and the structures employed to interpret these symbols (2) is reflected by the graphical user interface: MoMo has two main windows, a *note pad window* and a *graph window*, which correspond to the two fundamental components of the formalisms, to their syntax and to their semantics.

The note pad window, shown in Fig. 1, is divided into three areas: a *Descriptions* area containing a set of logical formulae, which can either be typed in or imported from an external file; a *Signature* area with a hierarchy of symbols which can be typed in, imported from an external file or inferred from a set of logical formulae (in DL mode); and an *Interpretations* area with a list of interpretations (which are created and manipulated in the graph window).

Every signature in MoMo starts with the key word `type_hierarchy` and ends with a period in the last line. These hierarchies are alternately called *type hierarchy* or *sort hierarchy* in the literature, and the symbols in the hierarchy are called *types* and *sorts* (in RSRL) or *concept names* (in DL). Every sort declared in the hierarchy is printed on a separate line. In Fig. 1, the first symbol in each line of the signature area between the key words `type_hierarchy` and `relation` declares a sort and its place in the sort hierarchy. All subsorts of a supersort S are below S and are further indented than S. Each sort in MoMo hierarchies may have more than one immediate

FIG. 1. Screen shot of MoMo: Note pad



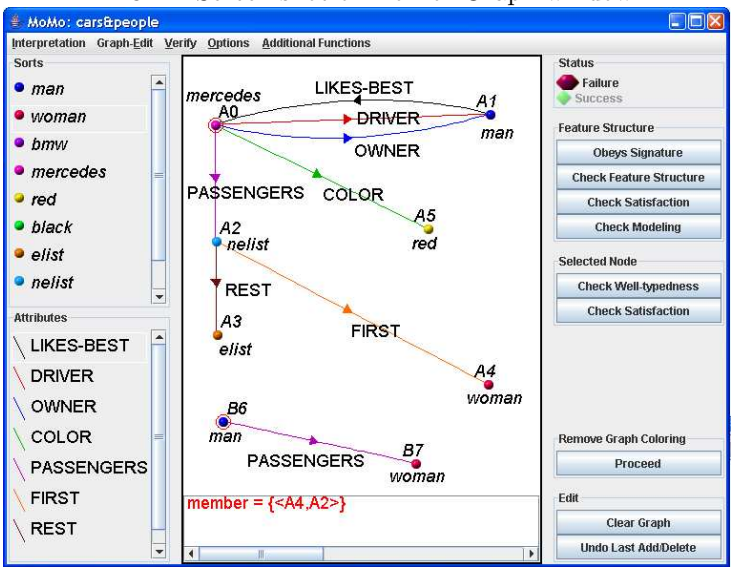
supersort, in which case it occurs more than once in the signature declaration. A symbol in the hierarchy may be followed by a sequence of *attribute* (or *role*) symbols and their values. They play different roles in RSRL and DL and will thus be explained below in the context of the respective formalisms.

The syntax of the logical languages is slightly different in RSRL and in DL. In both formalisms, however, the logical languages comprise the usual logical connectives. Conjunction is expressed by the symbol ",", and disjunction by ";". The symbol "$\sim$" stands for negation; "`*>`" represents implication and "`<*>`" equivalence; "$\hat{}$" and "`V`" denote existential and universal quantification, respectively. The equation symbol "=" is available for equating variables. Variables are printed with initial capital letters, all other non-logical symbols in lower case letters.

The graph window (Fig. 2) contains a toolbar in the left panel for drawing interpretations. Objects in interpretations are depicted as labeled nodes. The node labels come from the elements in the type hierarchy, the *sorts* of RSRL and the *concept names* of DL. Relationships between the objects are depicted as labeled arrows. The arrow labels are taken from the second set of non-logical symbols of the signature, the *attributes* of RSRL and the *roles* of DL. In RSRL mode the user usually works with *feature structures* (see Section 3). Feature structures in MoMo are rooted connected graphs. Root nodes are circled in red. A *relation field* at the bottom of the graph window (shown in Fig. 2 with the relation `member`) is available only in RSRL mode.

By pressing the buttons in the right panel of the graph window, the user can investigate various relationships between signatures or formulae and the configuration of nodes and arrows. The system responses to the user queries (described in more detail below) are shown graphically on the configurations themselves and by two status lights in the upper right corner of the window (the *status panel*). The green light *Success* indicates that all structures in the graph window have passed a test;

FIG. 2. Screen shot of MoMo: Graph window



otherwise the red *Failure* light lights up. MoMo is also equipped with a message window which informs about the results of user actions and provides links to Web resources such as an on-line tutorial. For syntax checks of formulae the message window gives detailed error reports, and similar error analyses are provided for well-formedness checks of configurations relative to a signature or a set of formulae.

By pressing the *Check Modeling* button, the user can check whether the structures in the graph window model the set of formulae in the note pad. An interpretation models a set of formulae iff every object in the interpretation satisfies every formula. If all the structures in the graph window model the set of formulae, then they will be outlined in red. If not, only those nodes in the interpretations are outlined in red that satisfy all the formulae in the note pad. All other nodes receive black circles, as illustrated in the screen shot on the right-hand side in Fig.3.

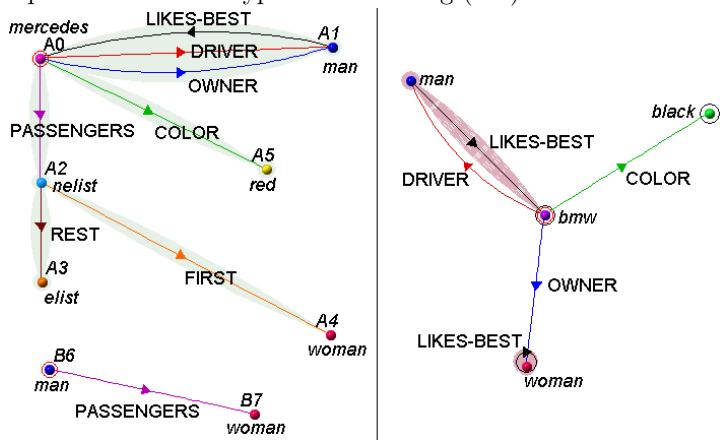
3 Relational Speciate Re-entrant Language

RSRL is a very expressive language belonging to the family of feature logics and designed to capture, as precisely as possible, the mathematical foundations of HPSG.[2] Its syntax uses feature paths, path equations, the standard boolean connectives, relational expressions (such as `member` and `append`) and a special kind of existential and universal quantification with restricted quantification domains (as required by HPSG). The syntax of RSRL is implemented in MoMo as a syntactic parallel to the feature logic descriptions of the HPSG grammar implementation environment TRALE[3] in order to establish a direct link to grammar implementations. More precisely, the syn-

[2] See [5] for mathematical details and a thorough explanation of the use of RSRL in HPSG.

[3] `www.sfs.uni-tuebingen.de/hpsg/archive/projects/trale/`

FIG. 3. Graph window: Well-typedness checking (left) and model checking (right)



tax of MoMo is an extension of the feature logic descriptions of the TRALE system. MoMo can read in TRALE's parsing output and generate interpretations of parse results.[4]

RSRL descriptions depend on *signatures* consisting of a set of sorts, a partial order on the sorts (the sort hierarchy), a set of attributes, attribute appropriateness conditions, and a set of relation symbols. The sorts, attributes and relation symbols provide the non-logical symbols of the language. Attribute appropriateness conditions declare certain attributes appropriate to certain sorts, and assign a second sort to each of these sort-attribute pairs as value. They impose restrictions on *well-formed* interpretations by requiring the presence of certain attribute arcs and of certain values of these arcs. An example of appropriateness conditions can be seen in the signature displayed in the signature area of Fig. 1: Each sort symbol in the hierarchy may be followed by a sequence of attribute-value pairs that are appropriate to it. Each attribute-value pair consists of an attribute followed by a colon and the sort that is appropriate for this attribute at the sort in the hierarchy stated at the beginning of the line. Attribute-value pairs are inherited by subsorts. In our example, the attribute LIKES-BEST is appropriate to the sort *person*, and the value of LIKES-BEST at person must be a *physical-entity*. Since *man* and *woman* are subsorts of *person*, they inherit these appropriateness specifications from *person*.

The symbols declared in the signature are used in descriptions of the formal language. The description area of the note pad in Fig. 1 shows three well-formed statements using our signature. Paraphrased in natural language, the three formulae say roughly the following: (1) It (i.e. the object being described) is a car or a person. (2) If it is a *bmw*, then the values of the attributes OWNER and DRIVER are identical. (3) Two objects x and y are in the `member` relation iff either x is the first element on list y or there is a tail z of list y and x is in the `member` relation with that tail z.

The effect of a signature on interpretations becomes clear when we consider graphs

[4] MoMo is not a complete implementation of RSRL as presented in [5], since it does not comprise the syntax and semantics of a construct called *chains*. Chains are an idiosyncratic construct of RSRL introduced for handling certain highly specialized uses of lists in the arguments of relational expressions in HPSG.

in the graph window. MoMo provides two kinds of interpretations for RSRL: (1) interpretations populated by totally well-typed and sort-resolved (concrete) feature structures with designated root nodes (feature structure semantics), and (2) more general interpretations which do not obey all algebraic restrictions of feature structure models (standard semantics). For simplicity we will concentrate on the feature structure semantics. The important underlying intuition about RSRL interpretations is the idea that the structures in the denotation of RSRL theories correspond to – or represent – structured objects in the real world which are described by logical theories.

The two feature structures shown in Fig. 2 illustrate interpretations of signatures in RSRL. The signature is taken from Fig. 1. Nodes in interpretations are labeled by maximally specific sorts only, i.e. sorts from the sort hierarchy which do not have any proper subsorts. Each node has an outgoing arc labeled with an attribute that is declared appropriate to it in the signature. Each of these attribute arcs points to a node which is either equal to the sort σ declared appropriate for the value of this attribute in the signature, or a subsort of σ. For any structure the user creates in the graph window, he can check (using the button *Obeys Signature*) whether the configurations on the canvas are well-formed with respect to the appropriateness requirements of the signature. The left-hand side of Fig. 3 shows the result of such a well-formedness test on the configuration in Fig. 2: The feature structure with root node A0 (of sort *mercedes*) is almost completely outlined in green, signaling that the colored part is well-formed. The exception is the node *woman* which lacks the attribute LIKES-BEST. The second feature structure (with root node B6 of sort *man*) is entirely ill-formed: An attribute arc labeled LIKES-BEST is missing from both nodes in the feature structure, and PASSENGER is not appropriate to *man*. The message window, not shown in the figure, gives the user exact information on the ill-formedness of the feature structures.

Check Satisfaction implements the RSRL notion of constraint satisfaction: Informally, a particular node n in an interpretation satisfies a description iff its substructure (i.e. the configuration of nodes reachable from n by following attribute arcs) is well-formed relative to the signature and it is in the denotation of the description. As described in Section 2, the notion of modeling builds on satisfaction: In RSRL, a feature structure models a description iff it satisfies it and so does every node which is accessible from the root node by following a sequence of arcs in the feature structure. The notion of substructure employed in this definition is also important in the definition of universal and existential quantification in RSRL. Unlike in first order logic, quantification in RSRL is not over the entire universe of objects, but rather over the substructures of feature structures, or over the nodes that we can reach from a given node by following the sequences of arcs accessible from the node. As a consequence, existence statements in terms of modeling are statements about the existence of nodes with certain properties within all substructures of a given feature structure. Similarly, universal statements are statements about properties of all nodes within all substructures of a given feature structure (or set of feature structures). The statement 'there exists an x such that x is of sort *woman*' is satisfied by each feature structure that contains a node labeled *woman*. It is modeled by each feature structure in which the substructure of each node in the feature structure contains a node labeled *woman*.

The right-hand side of Fig. 3 shows the result of model checking for a feature structure representing a *bmw*. It is well-formed with respect to the signature, but

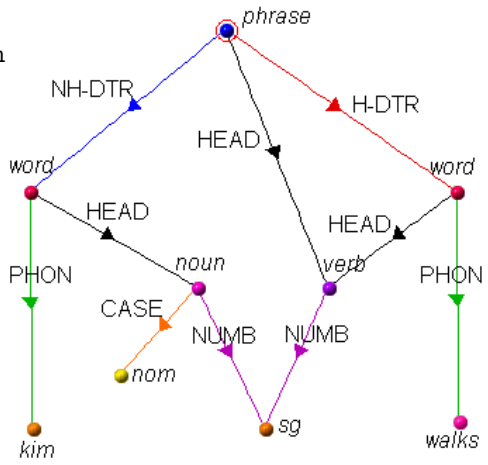
it does not model the theory in Fig. 1 because the BMW is owned and driven by different persons, and the node labeled *black* violates the constraint '`car ; person`'.

The relation field at the bottom of the graph window (see Fig. 2) contains tuples of nodes (chosen by the user) which are in the relations given by the signature. In order to visualize nodes in relations, each node on the canvas is automatically assigned a letter-integer combination by which it can be addressed. In Fig. 2 the node A4 with label *woman* stands in the `member` relation with the non-empty list node A2, which corresponds to the intuition that the woman is the only element on the list of passengers in the feature structure with root node A0 depicted in the graph window. The concrete feature structure with root node A0 and the `member` relation depicted in the relation field are thus a model of the definition of the `member` relation, the third logical statement in the description area of the note pad in Fig. 1.[5] For each node in the feature structure, any two nodes x and y in its substructure which are described by the right-hand side of the equivalence are in the `member` relation in the relation field. Alternatively, we can say that no nodes in the feature structure with root node A0 besides A4 and A2 fulfill the conditions stated for the `member` relation.

FIG. 4. Signature, theory and interpretation

Signature:

```
top
   sign head:head
        phrase h-dtr:sign nh-dtr:sign
        word  phon:phonstring
   head
      verb numb:numb
      noun numb:numb case:case
   case
      nom
      acc
   numb
      sg
      pl
   phonstring
      kim
      walks
```



Theory:

```
phrase*> head:Head,h-dtr:head:Head.
phrase*> h-dtr:head:(verb,numb:X),nh-dtr:head:(noun,numb:X,case:nom).
word*> (phon:kim,head:(noun,numb:sg));(phon:walks,head:(verb,numb:sg)).
```

Fig. 4 shows a simplified linguistic example of the use of RSRL in HPSG-style grammars. The example has three simple grammar principles: The first one is a version of a HEAD FEATURE PRINCIPLE (see [4]) saying that the HEAD value of each

[5] Modulo the fact that the feature structure is not entirely well-formed with respect to the signature because the node A4 is missing an appropriate attribute.

phrase must equal the HEAD value of the head daughter.[6] The second principle says that the head daughter of each phrase is a verb, and its NUMBER value must equal the NUMBER value of the non-head daughter, which is a noun. The last principle is a simple version of a WORD PRINCIPLE. It defines the words in the lexicon: If something is a word, it is either a singular noun with phonology *Kim*, or it is a verb with singular agreement features and phonology *walks*. Under modeling, this grammar licenses structures like the one depicted in Fig. 4. In other words, the logical statements we listed are true of every node in this feature structure. For most of the nodes the statements are trivially true, since they are neither words nor phrases. For the two nodes labeled *word* and the one node labeled *phrase* it is easy to verify that the consequents of the implicational descriptions hold.

Additional functions of MoMo in RSRL mode allow the manipulation of interpretations: MoMo can map concrete feature structures on their corresponding abstract feature structure representations, which are an elegant encoding of equivalence classes of isomorphic concrete feature structures. Partial feature structures can be extended automatically to complete feature structures which obey all requirements of a given signature. For a more comprehensive description of these and other features, the reader is referred to the User's Manual[7] and to [6].

4 Description Logics

Description logics are a widely used class of model-theoretic logics, designed for the representation of terminological knowledge (ontologies) and for reasoning based on this knowledge (see [2]). Logical formulae in DL can be translated into first order logic (FOL) or a slight extension thereof. For a subset of description logics called $\mathcal{ALCN}$ there is a close, illuminating correspondence to a variant of RSRL (see Section 5). Due to this relationship the graphical interface of MoMo is well-suited for the visualization of the central concepts of $\mathcal{ALCN}$. The implementation of DL in MoMo is expected to be useful for knowledge engineering purposes as well as for teaching the mathematical foundations of knowledge bases.[8] Here we will focus on those aspects of description logics which are directly relevant to the implementation of DL in MoMo.

An ontology expressed in a description logic represents a terminological knowledge base which makes it possible to draw inferences and thus gain new knowledge. A knowledge base in DL consists of a *terminology TBox* representing a vocabulary and application domain and an *ABox* containing *assertions* about the concrete individuals (objects) in the denotation of the concepts from the vocabulary. In what follows we will focus on TBoxes, because at present ABoxes are not implemented in MoMo.

A TBox contains *terminological axioms* of the form $C \sqsubseteq D$ and $C \equiv D$ (written in MoMo as C`*>`D and C`<*>`D), where C and D are *concept descriptions*. Concept descriptions are constructed from *concept names* (denoting sets of individuals) and *role names* (denoting binary relations between individuals) by syntactic rules which are interpreted model-theoretically. An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is the domain of interpretation and $\cdot^{\mathcal{I}}$ is an interpretation function. For every atomic

[6] Variables in descriptions which are not explicitly bound by a quantifier are assumed to be bound by an implicit existential quantifier taking wide scope.

[7] User's Manual: `milca.sfs.uni-tuebingen.de/A4/Course/Momo/manual/momo-manual-par.pdf`

[8] This work was carried out in the project *Adaptive Ontologies on Extreme Markup Structures*, concerned with the automatic extension of ontologies. See `tcl.sfs.uni-tuebingen.de/tt/english.html` for further information.

concept $A : A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and for every role R: $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. At the moment MoMo implements $\mathcal{ALCN}$-DL (see [2] for definition). The syntax, semantics and MoMo notation of the concept descriptions in the logic are defined in Table 1. R depicts role names and C depicts concept descriptions.

TABLE 1. Description logic $\mathcal{ALCN}$: Syntax, semantics and MoMo notation

DL Syntax	**Semantics**	**MoMo Notation**
$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$	*top*
$\bot$	$\bot^{\mathcal{I}} = \{\}$	$\sim top$
$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	C_1, C_2
$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	$C_1; C_2$
$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\sim C$
$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y : \langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$	$\hat{}R.C$
$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y : \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	$\mathrm{V}R.C$
$\leq nR$	$(\leq nR)^{\mathcal{I}} = \{x \mid \lvert\{(x, y) \in R^{\mathcal{I}}\}\rvert \leq n\}$	$<= nR$
$\geq nR$	$(\geq nR)^{\mathcal{I}} = \{x \mid \lvert\{(x, y) \in R^{\mathcal{I}}\}\rvert \geq n\}$	$>= nR$

An interpretation $\mathcal{I}$ *satisfies* an axiom $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ satisfies $C \equiv D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ is called a *model* of a TBox iff it satisfies all axioms in the TBox. The intuition is that interpretations in DL represent networks of relations between objects that instantiate the concepts defined in the ontology.

In MoMo the user can construct interpretations as graphs and check if they model a TBox contained in the note pad. Nodes in a graph represent individuals belonging to the interpretation of the atomic concepts listed as node labels. Arrows represent relations between objects. In contrast to interpretations in RSRL, individuals may be instances of several atomic concepts in DL.

The example in Fig. 5 contains an ontology that describes family relations. The taxonomy is automatically inferred by MoMo from the ontology. The ontology in Fig. 5 can be paraphrased as follows: (1) *If a person has children or spouses then the children and spouses are persons.* (2) *A woman is a person.* (3) *A man is a person who is not a woman, and vice versa.* (4) *A parent is a person who has a child, and vice versa.* (5) *A grandmother is a woman who has a child that is a parent, and vice versa.* (6) *A wife is a woman who has a spouse who is a man, and vice versa.* (7) *A husband is a man who has a spouse who is a woman, and vice versa.*

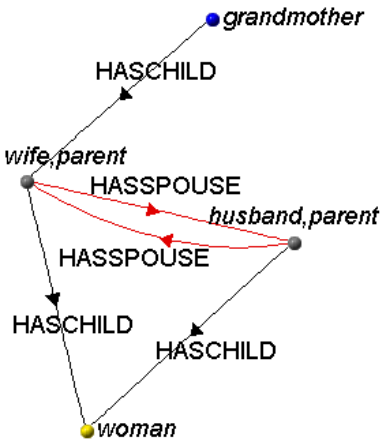
The interpretation in Fig. 5 shows a small family that consists of a grandmother with a daughter who is married to a man and has a daughter with him.

One important DL inference task carried out by MoMo is subsumption hierarchy extraction, resulting in a taxonomy of the concept names in the knowledge base. Fig. 5 shows the extracted subsumption hierarchy for the TBox in our example. Note that the binary relations of DL correspond to the attributes of RSRL, but unlike in RSRL there are no type restrictions in DL on the arguments of relations. In the hierarchy they are simply declared appropriate to the root type *top*, and their values are again of type *top*. The presence of the type *top* is obligatory in every hierarchy; it is the

FIG. 5. TBox, interpretation and extracted taxonomy

Ontology:

```
person*>VhasChild.person,VhasSpouse.person.
woman*>person.
man<*>person,~woman.
parent<*>person,^hasChild.top.
grandmother<*>woman,^hasChild.parent.
wife<*>woman,^hasSpouse.man.
husband<*>man,^hasSpouse.woman.
```



Inferred taxonomy:

```
top hasChild:top hasSpouse:top
  person
      woman
        grandmother
        wife
      man
        husband
      parent
        grandmother
```

top concept subsuming all other concepts.

The subsumption hierarchy helps to keep the interpretation graphs compact. If a node is labeled with an atomic concept C, it is assumed that the node belongs to the interpretation of all superconcepts of C in the subsumption hierarchy. It follows that the interpretation in Fig. 5 models the TBox under the subsumption hierarchy, although the nodes labeled *grandmother* and *wife, parent* do not explicitly belong to the interpretation of *person, woman*, the node *husband, parent* is not labeled *person, man*, and the node *woman* is not labeled *person*. At present only simple structural subsumption has been implemented.[9] There are plans to integrate the KAON2 plug-in that reasons over OWL-Lite and SWRL ontologies.[10]

5 On the Relationship between RSRL and DL

The fact that RSRL and DL live side by side in MoMo naturally prompts questions about their suggestive model-theoretic similarities. However, RSRL and DL were made for substantially different application areas. RSRL is a feature logic designed for the description of the structure of linguistic signs, whereas description logics describe networks of relations between objects in the world. DL can be translated into FOL (the converse is false) and constitutes a relatively weak formalism. RSRL theories cannot in general be expressed in FOL, nor vice versa.[11]

Crucial mathematical differences between RSRL and description logics are salient: (1) RSRL interprets attributes as partial functions, while the corresponding DL roles are relations. (2) RSRL theories comprise an elaborate signature that imposes restric-

[9] See [2] for further discussion and explanations.

[10] `http://kaon2.semanticweb.org/`, `www.kaon2.semanticweb.org`, `www.w3.org/2004/OWL/`, and `www.w3.org/Submission/SWRL/`

[11] RSRL theories using only the SRL fragment of the formalism can be translated into FOL (see [1]).

tions on interpretations: Well-typedness entails that only totally well-typed structures can model a theory. Nothing like this is required in description logics. (3) In RSRL exactly one (maximally specific) sort is assigned to every object in the interpretation. In DL an object may simultaneously be in the denotation of several atomic concepts.

To see the relationship between DL and RSRL it is useful to relax several constraints on RSRL interpretations. Let RSRL* be a variant of RSRL in which interpretations are not required to be well-typed, attributes are relational instead of functional, and sorts are mapped to sets of objects. In addition we demand that each RSRL* signature have a top sort subsuming all other sorts in the signature, and we call this sort *top*. With these modifications in place, Table 2 shows how $\mathcal{ALCN}$-DL (defined in Section 4) and its sublogics can be translated into RSRL*.[12] The translation is based on the MoMo syntax of RSRL informally described in Section 3.

TABLE 2. Translation of the description logic $\mathcal{ALCN}$ into RSRL*

DL	**RSRL***
$\top$	top
$\bot$	$\sim top$
$C_1 \sqcap C_2$	$C_1\ ,\ C_2$
$C_1 \sqcup C_2$	$C_1\ ;\ C_2$
$\neg C$	$\sim C$
$\exists R.C$	$\hat{}X(R : X, X : C)$
$\forall R.C$	$VX(R : X \texttt{ *> } X : C)$
$\leq nR$	$VX_1...VX_{n+1}((R : X_1, ..., R : X_{n+1})\texttt{*>}\ (X_1 = X_2; ...; X_n = X_{n+1}))$
$\geq nR$	$\hat{}X_1...\hat{}X_n((R : X_1, ..., R : X_n),\ (\sim (X_1 = X_2), ..., \sim (X_{n-1} = X_n))$

The translation indicated in Table 2 is defined in such a way that every model of a DL ontology is a model of the corresponding RSRL* theory, and vice versa. While we will not prove this result here, we want to draw the reader's attention to an important property of the interpretation of quantifiers. RSRL employs non-standard quantification over restricted quantification domains, whereas DL uses standard FOL quantification ranging over the entire set of objects in a given universe. Given that, the question arises how quantificational expressions can be translated correctly. The answer lies in the actual use of quantification in DL. When we translate DL expressions into FOL[13] we observe that quantification is always over variables occurring in the second arguments of roles, i.e. in terms of RSRL it is over feature values. Thinking in terms of the graph representation of the DL models, it is clear that for every node n in the interpretation graph, terminological definitions from the ontology are verified with respect to the substructure of n. But this means that terminological definitions are modeled by a variant of feature structures. As a result, while the relationship between RSRL and DL is intricate it is close enough to express a DL ontology in the syntax of RSRL and to obtain corresponding feature structure models in RSRL*. In fact, this correspondence is at the heart of the MoMo implementation of DL.

[12] The translation in Table 2 ignores the necessary uniqueness of variable names.

[13] The translation of DL expressions into FOL is similar to the one shown in Table 2 (see [2]).

6 Conclusions

MoMo has been used successfully both in HPSG grammar implementation courses as well as in teaching courses in HPSG, both in face-to-face teaching and in e-Learning. In classes that focused on linguistic problems MoMo functioned as a quick and informal primer on the logical foundations of HPSG; it served as the introduction to mathematically oriented courses on the logical foundations of constraint-based grammar formalisms, where the use of MoMo preceded an in-depth study of feature logics; and it was used in classes on grammar engineering, where it served to explain the declarative meaning of logical grammatical constraints and to complement and strengthen the students' understanding of the computational tasks carried out by parsing systems for constraint-based grammars.

MoMo is a central interactive software element in a web-based course on constraint-based grammar formalisms and parsing (see [6] and references therein for a description of the course design and goals). In this course, MoMo serves to visualize the explanations and examples of the electronic textbook in the introductory section on the feature logic of HPSG (five to six weeks of course work). The textbook provides MoMo files with exercises that students complete at regular intervals and submit for evaluation. The software gives immediate feedback in self-paced study and guides students toward correct solutions. Evaluation questionnaires of the web-based course showed that MoMo was very popular with the students. The use of the software substantially improves their grasp of the relationship between a constraint-based grammar and its meaning in terms of sets of feature structures compared to traditional face-to-face teaching, which provides less opportunity for working through a sufficient number of examples for each new concept and does not adapt as well to differences in the background knowledge of students. In the near future, we also plan to use MoMo in seminars on DL ontologies.

The next inference task to be implemented in MoMo is the automatic construction of models satisfying not only a given signature but also a theory. This functionality is important both in HPSG and for DL frameworks. Since the general problem is undecidable in RSRL, this is a non-trivial task which cannot receive a general and fully automatic solution. We are, however, confident that we can implement a satisfactory solution for an interesting class of cases.

References

[1] Aldag, B. A proof theoretic investigation of prediction in HPSG. Magisterarbeit, Universität Tübingen, 1997.

[2] Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds. *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[3] Etchemendy, J. and J. Barwise. *Language, Proof and Logic*. Stanford: CSLI Publications, 1999.

[4] Pollard, C. and I. A. Sag. *Head-Driven Phrase Structure Grammar*. U. of Chicago Press, 1994.

[5] Richter, F. A mathematical formalism for linguistic theories with an application in Head-driven Phrase Structure Grammar. Dissertation (2000), Universität Tübingen, 2004.

[6] Richter, F., E. Ovchinnikova, B. Trawiński, D. Meurers. Interactive Graphical Software for Teaching the Formal Foundations of Head-Driven Phrase Structure Grammar. In Jäger, G., P. Monachesi, G. Penn, S. Wintner, eds. *Proceedings of Formal Grammar 2002*, pp. 137–148, 2002.

Received 31 May 2007.