Morph Moulder: Graphical Software for Teaching and Visualizing RSRL and Description Logics

Ekaternina Ovchinnikova and Frank Richter
University of Tübingen

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 recently been extended to a treatment of Description Logics (DL). With MoMo, students can construct interpretations of sets of formulae and check whether their interpretations model these formulae. MoMo 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.

1 Motivation

MoMo was originally developed as an interactive educational tool for teaching the mathematical foundations of HPSG based on Relational Speciate Re-entrant Language (RSRL, [3]). To overcome the difficulties novices in constraint-based linguistic theories often have in understanding the essential relationship between grammars as sets of logical statements and their model-theoretic interpretation, 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 [2], and in particular from one of its software tools, *Tarski's World*.

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.

In Section 2 we will discuss the general functions of MoMo; Sections 3 and 4, respectively, will introduce functions which are specific either to RSRL or to description logics. Section 5 will conclude with a few general observations.

2 MoMo: Visualization and Reasoning

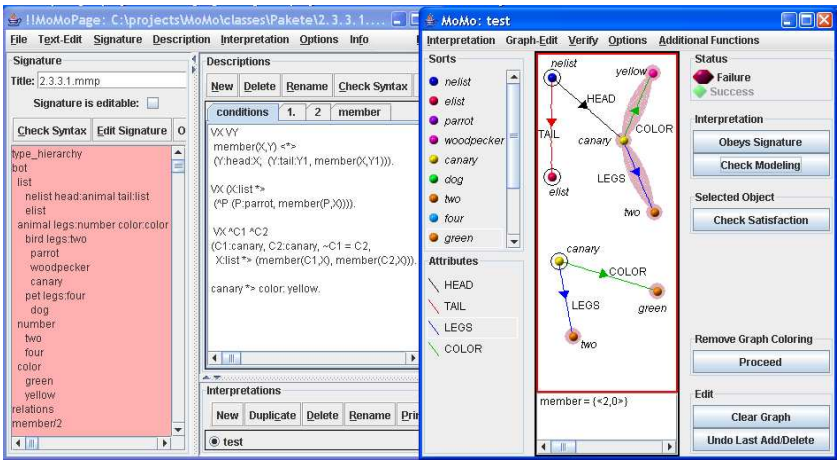
MoMo's most important functions visualize the relationships between sets of logical formulae and their model-theoretic interpretations, and support reasoning about interpretations of logical statements.[1] In MoMo, users can (1) write down sets of logical formulae; (2) declare arbitrary sets of basic non-logical syntactic

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

symbols and organize them into a subsumption hierarchy; (3) construct interpretations of a given set of formulae as labeled (possibly cyclic) directed graphs.

MoMo has two main windows, shown in Fig. 1. The note pad window is divided into three areas: a *Descriptions* area containing a set of logical formulae; a *Signature* area with a hierarchy of symbols which can be 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).

Figure 1: Screen shots of MoMo: Note pad (left) and graph window (right)



The graph window contains a toolbar for drawing interpretations. Objects in interpretations are depicted as labeled nodes. The node labels come from the elements in the hierarchy and are called *sorts* in RSRL and *concept names* in DL. Relationships between the objects are depicted as labeled arrows. The arrow labels are taken from the non-logical symbols assigned in the hierarchy to each object type. These are called *attributes* in RSRL and *relations* in DL. The relation field at the bottom of the graph window is available only in RSRL mode.

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.[2] If all the structures in the graph window model the set of formulae, the green light *Success* lights up in the upper right corner of the graph window, and the interpretations are outlined in red. If not, the red light *Failure* lights up and only those nodes in the interpretations that satisfy all the formulae in the note pad are outlined in red. All other nodes receive black circles, as illustrated in Fig.1. MoMo is also equipped with a message window which informs the user about the results of the tests and sometimes provides direct links to teaching materials on the Web.

3 Relational Speciate Re-entrant Language

RSRL is a very expressive feature logic designed to capture, as precisely as possible, the mathematical foundations of HPSG. Its syntax uses feature paths, path equations, the standard boolean connectives including disjunction and negation,

[2]The notion of satisfaction is explained below in terms of the concrete formalisms.

relational expressions (such as `member` and `append`) and a special restricted kind of quantification. The syntax of RSRL is implemented in MoMo as a syntactic parallel to the descriptions of the HPSG grammar implementation environment TRALE[3] in order to establish a direct link to grammar implementations.

RSRL descriptions depend on a given signature consisting of a set of sorts, a partial order on the sorts (the sort hierarchy) and a set of attributes. The sorts and attributes provide the non-logical symbols of the language. Attribute appropriateness conditions in the signature, which declare certain attributes appropriate for certain sorts, assigning a second sort to each of these sort-attribute pairs as value, impose restrictions on well-formed interpretations by requiring the presence of certain attribute arcs and of certain values of these arcs.

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. *Check Satisfaction* implements the RSRL notion of constraint satisfaction: Informally, a particular node in an interpretation satisfies a description iff its substructure is well-formed and it is in the denotation of the description. The relation field at the bottom contains user-defined tuples of nodes which are in the relations given by the signature. Each node on the canvas is assigned an integer representing it in the relation tuple.

Another important function in RSRL mode is the automatic construction of complete well-formed feature structures from partial feature structures. For a complete description of this and other features, the reader is referred to the User's Manual and to [4].

4 Description Logics

Description logics are a widely used class of model-theoretic logics, designed for the representation of ontologies and for reasoning based on the knowledge they contain (see [1] for an overview). Logical formulae in DL can be translated to first order logic or a slight extension thereof. A subset of description logics called $\mathcal{ALCN}$ can be translated to RSRL. These observations inspired the addition of description logics to MoMo in the context of the project *Adaptive Ontologies on Extreme Markup Structures*.[4]

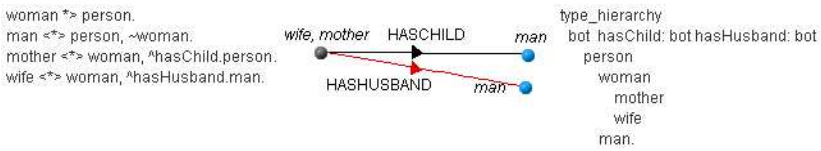
A DL ontology is represented by a *TBox* containing *terminological axioms* of the form $C \sqsubseteq D$ or $C \equiv D$, where C and D are concepts. Concepts are constructed from *concept names* (denoting a set of individuals) and *relation names* (denoting binary relationships between individuals) 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. At the moment MoMo implements $\mathcal{ALCN}$-DL. 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 $C^{\mathcal{I}} = D^{\mathcal{I}}$.

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 (see Fig. 2). Unlike in RSRL, individuals may be instances of several atomic concepts in DL.

One important DL inference task of MoMo is subsumption hierarchy extraction. At present only simple structural subsumption has been implemented. We

[3] `www.sfs.uni-tuebingen.de/hpsg/archive/projects/trale/`
[4] `tcl.sfs.uni-tuebingen.de/tt/english.html`

Figure 2: TBox, interpretation and extracted subsumption hierarchy



plan to integrate the KAON2 plug-in that reasons over OWL-DL and SWRL ontologies.[5] Fig. 2 shows the extracted subsumption hierarchy for an example TBox. Unlike in RSRL, there are no necessary restrictions in DL on the arguments of relations. In the hierarchy they are declared to be of the root type *bot*.

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 according to the subsumption hierarchy. It follows that the interpretation in Fig. 2 (center) models the TBox (on the left) under the subsumption hierarchy (on the right) although the node labeled *mother, wife* does not explicitly belong to the interpretation of *person, woman*, and the nodes labeled *man* are not labeled *person*.

5 Conclusions

MoMo has been used successfully both in teaching courses in HPSG and in grammar implementation in HPSG. It is very popular with the students and substantially improves their understanding of the relationship between a constraint-based grammar and its meaning in terms of a set of configurations of objects. We plan to use MoMo in seminars on DL ontologies soon.

The next inference to be implemented in MoMo is automatic construction of a model 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 an interesting 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] Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds. *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

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

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

[4] Richter, F., E. Ovchinnikova, B. Trawiński, D. Meurers. Interactive Graphical Software for Teaching the Formal Foundations of Head-Driven Phrase Structure Grammar. In *Proc. of Formal Grammar 2002*, pp. 137–148, 2002.

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