Minimal Recursion Semantics and the Language of Acyclic Recursion

Roussanka Loukanova

Uppsala, Sweden

Abstract. Moschovakis (2003-2006) developed a logical calculus of the formal language $\mathsf{L}^{\lambda}_{\mathsf{ar}}$ of acyclic recursion, which is a type-theoretical work with many potential applications. On the implementation side, large-scale grammars for human languages, e.g. versions of HPSG, have been using semantic representations casted in the feature-value language Minimal Recursion Semantics (MRS). While lacking strict formalization, MRS represents successfully ambiguous quantifier scoping. In this paper, we introduce the basic definitions of MRS by reflecting on possibilities for formalization of MRS with a version of the language $\mathsf{L}^{\lambda}_{\mathsf{ar}}$.

1 Introduction: Why MRS Representations?

Research presented in this paper targets formalization and development of analysis with syntax-semantics interface of spoken and written human language (incl. texts larger than sentences), which continues to be a largely open area, in need of theoretical foundations for reliable coverage.

Versions of constraint-based lexicalist grammar (CBLG), in particular of Head-Driven Phrase Structure Grammar (HPSG), have achieved significant developments and accumulated large resources for English and other languages, incl. for Norwegian, Danish, Arabic, etc. An international consortium, which originates by work in Stanford[1], developed a grammar tool, LKB, for writing grammars of human languages. In the last years, LKB comes with possibility for semantic representation, by using Minimal Recursion Semantics (MRS), see [3]. By its nature, MRS in HPSG is a notational, specialized version of Situation Semantics, with feature-value structures representing information terms.

Among the existing approaches to theory of meaning of natural language (NL), model-theoretic alternatives provide viability of computational semantics for applications to the study of language faculty, knowledge representation in general, and in particular, for representation of linguistic knowledge, and development of intelligent computerized systems. Typically, computational semantics of NL involves rendering of NL expressions into some formal logic language. First-order languages and logic, while well-studied and understood, have repeatedly exposed their unsuitability for semantics of NL, from the perspectives of computability and linguistic adequacy. On the other hand, higher order languages and typed λ-calculi have pleasant computational properties, but are still problematic from theoretic and application points as theories of meaning, representation of knowledge and information flow, for which they are under active

[1] see <http://www.delph-in.net/lkb/>

Loukanova R..
Minimal Recursion Semantics and the Language of Acyclic Recursion.
DOI: 10.5220/0003309800880097
In *Proceedings of the 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology* (BILC-2011), pages 88-97
ISBN: 978-989-8425-42-3
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

developments, for applications to semantics of artificial languages, e.g., for semantics of programming languages, and for NL.

Semantic representation, including rendering of NL expressions into formal logic languages, such as first or higher order languages, have been problematic in systems for NLP. The variety of such applications is large and growing: semantic transfer approaches to machine translation (MT), obtaining semantic representations by parsing NL sentences, generation of NL sentences for given semantic representations, and other more advanced applications to automatic understanding of NL, for example, in question-answer systems, information transfer, information extraction, knowledge representation systems including knowledge inference, update, etc.

For example, a simplified semantic transfer schemata for MT typically consists of the following stages, some of which, at least the first two, may be carried on in a compositional way:

Parsing an expression of a source NL, which produces:
Semantic representation of the input NL expression in some formal language. The semantic representation is called *source LF*.
A transfer component, which converts the source LF into a semantic representation, called the *target* LF.
A generator converts the target LFs into expression(s) of the target NL.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

In such systems, ideally, a semantic analyzer of the source NL sentences produces semantic representations, called logic forms (LFs) in some formal language, to be used for generating logically equivalent sentences in a target NL. The basic problems that emerge are related to mismatch between LFs and NL expressions. The LF produced by a parser typically carries on the syntactic structure of the input NL expression. For example, the order of the atomic formulas in a LF, e.g., such as a conjunction, may correspond to the syntactic structure of the NL expression, while it is irrelevant for the semantic interpretation. In a simplified approach, a generator can be build to try all logically equivalent LFs until finds the appropriate ones. Such approaches meet serious problems, for example, involving spurious ambiguity or unacceptability; e.g., analyses may produce various logically equivalent LFs some of which correspond to unacceptable NL sentences (see Copestake et al. for examples and discussion). Depending on the formal language chosen, such approaches may inherit some serious drawbacks with respect to computability: computational inefficiency and/or undecidability of the problem of logical form equivalence. Some of these problems get pleasantly resolved for a semantic core of NL, which has a syntactic expression in NL, by a recent development of a grammatical framework (GF), see [12] and [13].

Some of the classic semantic theories used in NLP may carry on more fundamental problems, among which, a serious one is the quantifier scope ambiguity. This is demonstrated by any of the notorious examples, with at least two quantifier NPs, like (1a), for which there is only one classic context-free parse tree, while having more than one possible logic forms, representing alternative scoping:

(1) a. $[[\text{Every man}]_{NP}$ loves $[\text{a woman}]_{NP}]_S$.
b. *de dicto* reading:

1. [every man]$_i$ [[a woman]$_j$
   [e$_i$ loves e$_j$]$_S$]$_S$]$_S$
2. $\forall x(man(x) \rightarrow \exists y(woman(y) \wedge love(x,y)))$

c. *de re* reading:
1. [[a woman]$_j$ [[every man]$_i$
   [e$_i$ loves e$_j$]$_S$]$_S$]$_S$.
2. $\exists y(woman(y) \wedge \forall x(man(x) \rightarrow love(x,y)))$

Among the quantifiers and quantifier scope ambiguity problems for NLP the following ones are ongoing:

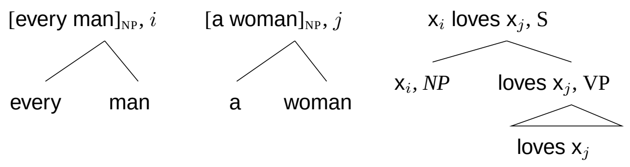
1. A typical (context-free) parser gives only one parse tree structure, which corresponds to multiple LFs, without any direct compositional way to derive them.
2. A classic style treatment of quantifiers (as in the lines of classic Montague's PTQ, see [8]) results in computational inefficiency, in particular, if all readings of a NL expression with several quantifiers have to be derived at each level of processing the sentence and its components.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

Linguistic studies of underspecification in human languages (NL) have been broadly reviewed in [2]. Some approaches, closely related to the topic of this paper, have been tried in logic type theories to represent multiple scoping by techniques for underspecified representation: for an example in the line of a Montagovian approach, see [10]; for representation of quantifier ambiguities and, in general, of partiality of information in Situation Semantics, by using Situation Theory, see [4]. A simplified version of a quantifier storage technique was implemented in HPSG, e.g., see [11], which in recent years evolved in elaborated MRS representation, see [3]. More recently, a new approach has been initiated in [5], [6], and [7].

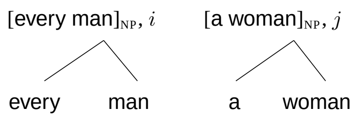
A demonstrative example for underspecified scoping is depicted by the following unconnected graph ("underspecified tree"), which carries information about the "bare"-predicate structure of the sentence, where the "disconnected" quantifiers carry indexing information about the corresponding subject–complement argument roles they fill up.

(2) Underspecified tree (actually a graph) structure:

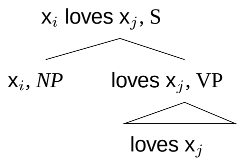


The set of the two indexed NP sub-graphs represents syntactically the 'logic' storage needed for computing the resolved logic form of the sentence, e.g., see [4]. The S subtree represents the underspecified semantic basis of the sentence. By this partially connected graph we have a syntactical representation of the underspecified logic form:

(3) a. Quantifier storage:



b. Underspecified semantic basis:



2 MRS Representations and Notations

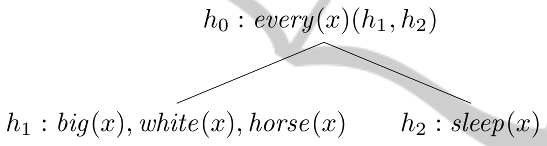
MRS uses a language with n-ary conjunction. Since binary conjunction $\wedge$ causes spurious ambiguity in parsing NL expressions, for efficiency of processing, $\wedge$ is taken as an n-ary operator, which is represented implicitly: any list of atomic formulas is interpreted as a conjunction.

Elementary predication (EP) is any atomic formula or a conjunction of atomic formulas. EPs are tagged with labels. Thus the MRS formula (4a) is represented by the tagged tree (4b):

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

(4) a. $every(x)\Big(\wedge\big(big(x), white(x), horse(x)\big),\ sleep(x)\Big)$

b.


$h_0 : every(x)(h_1, h_2)$
$h_1 : big(x), white(x), horse(x)$
$h_2 : sleep(x)$

The above MRS representations (4a) and (4b) can be written in the following notation, by using assignments to resemble terms in Moschovakis language L^{λ}_{ar}:

(5) h_0 where $\{\ h_0 := every(x, h_1, h_2),$
$h_1 := \wedge(big(x), white(x), horse(x)),\ h_2 := sleep(x)\ \}$

Note that, in Moschovakis' calculus of L^{λ}_{ar}, coordination terms containing conjunction (the value of h_1 above) undergo further reduction to canonical forms. In a resolved term, the *head* subterm does not need to be assigned to any location like h_0 above. In a realistic NL grammar, top locations (labels) simplify the compositional derivations, but add "computational" steps.

MRS uses three kinds of variables:

1. *Variables*, called also *parameters*, for quantification over and reference to individuals denoted by NPs and for filling up argument slots of relations, when the arguments designate individuals;
2. *Labels*, for tagging EPs and filling up argument slots of relations, when the arguments are for EPs;
3. *Free labels* for labels to which no EPs are assigned.

Respectively, the language $\mathsf{L}^{\lambda}_{\mathsf{ar}}$ has two sorts of variables:

1. *Pure* variables, which are to be quantified, i.e. corresponding to the MRS variables for individuals;
2. *Recursion* (*location*) variables to which EPs are assigned, i.e. these correspond to the MRS labels. For example: $h := \mathit{sleep}(x)$, where h is a location variable, i.e. a label, and x is a pure variable.

Different MRS Representations of the NL Sentence.

(6) *Every dog chases some white cat.*

(7) a. ***De Re* reading, in predicate language**:

$$\mathit{some}(y)[(\mathit{white}(y) \wedge \mathit{cat}(y)) \wedge \\ \mathit{every}(x)\,(\mathit{dog}(x) \rightarrow \mathit{chase}(x,y))]$$

b. ***De Re* reading, by a MRS tree:**

$$\begin{tikzcd}
 & h_5 : \mathit{some}(y) \arrow[dl, dash] \arrow[dr, dash] & & \\
h_7 : \mathit{white}(y), \mathit{cat}(y) & & h_1 : \mathit{every}(x) \arrow[dl, dash] \arrow[dr, dash] & \\
 & h_3 : \mathit{dog}(x) & & h_4 : \mathit{chase}(x,y)
\end{tikzcd}$$

c. ***De Re* reading, in MRS term:**

$$h_5 : \mathit{some}(y, h_7, h_1),\ h_7 : \mathit{white}(y),\ \mathit{cat}(y), \\ h_1 : \mathit{every}(x, h_3, h_4),\ h_3 : \mathit{dog}(x),\ h_4 : \mathit{chase}(x,y)$$

d. ***De Re* reading, in the language $\mathsf{L}^{\lambda}_{\mathsf{ar}}$:**

$$h_5 \text{ where}\{\ h_5 := \mathit{some}(h_7, h_1) \\ h_7 := \lambda y\,(p(y) \& q(y)), \\ p := \lambda y\, \mathit{white}(y),\ q := \lambda y\, \mathit{cat}(y), \\ h_1 := \lambda y\, \mathit{every}(h_3(y), h_4(y)), \\ h_3 := \lambda y\, \lambda x\, \mathit{dog}(x),\ h_4 := \lambda y\, \lambda x\, \mathit{chase}(x,y)\}$$

(8) a. ***De Dicto* reading, in predicate language:**

$$\mathit{every}(x)[\mathit{dog}(x), \mathit{some}(y)(\wedge(\mathit{white}(y), \mathit{cat}(y)), \mathit{chase}(x,y))]$$

b. ***De Dicto* reading, in MRS:**

$$\begin{tikzcd}
 & h_1 : \mathit{every}(x) \arrow[dl, dash] \arrow[dr, dash] & & \\
h_3 : \mathit{dog}(x) & & h_5 : \mathit{some}(y) \arrow[dl, dash] \arrow[dr, dash] & \\
 & h_7 : \mathit{white}(y), \mathit{cat}(y) & & h_4 : \mathit{chase}(x,y)
\end{tikzcd}$$

c. ***De Dicto* reading, by a MRS term:**

$$h_1 : \mathit{every}(x, h_3, h_5), \quad h_3 : \mathit{dog}(x), \\ h_5 : \mathit{some}(y, h_7, h_4), \quad h_7 : \mathit{white}(y),\ \mathit{cat}(y), \quad h_4 : \mathit{chase}(x,y)$$

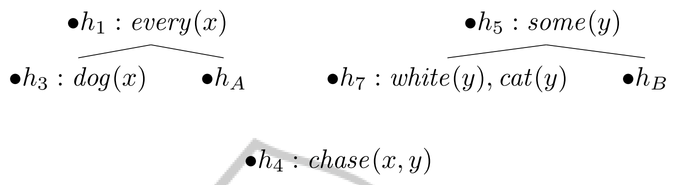
d. ***De Dicto* reading, in the language $\mathsf{L}^{\lambda}_{\mathsf{ar}}$:**

$$h_1 \text{ where } \{\ h_1 := \mathit{every}(h_3, h_5),\ h_3 := \lambda x\, \mathit{dog}(x), \\ h_5 := \lambda x\, \mathit{some}(h_7, h_4(x)),\ h_7 := \lambda y\,(p(y) \& q(y)), \\ p := \lambda y\, \mathit{white}(y),\ q := \lambda y\, \mathit{cat}(y), \\ h_4 := \lambda x \lambda y\, \mathit{chase}(x,y)\}$$

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

Underspecified Representation. The underspecified quantification can be depicted by the labeled graph with nodes that are unconnected:

(9) a. **Underspecified MRS graph:**



b. **Underspecified MRS term:**

$h_1 : every(x, h_3, h_A)$, $h_3 : dog(x)$,
$h_5 : some(y, h_7, h_B)$, $h_7 : white(y),\ cat(y)$,
$h_4 : chase(x, y)$
There are exactly two ways of assigning EPs to the variables h_A and h_B to form well formed MRS terms and corresponding tree representations I.e., the system of equations has exactly two solutions corresponding to:

1. $h_A : h_5, \quad h_B : h_4$;
2. $h_A : h_4, \quad h_B : h_1$.

Note that, in MRS, the variables h_A and h_B are called *handles*.

c. **Underspecified $\mathsf{L}^{\lambda}_{\mathsf{ar}}$-term:** (for example, by using the rules of the reduction calculus)

$h_0(u)$ where $\{\ h_1 := \lambda y\, every(h_3, \lambda x\, h_A(x)(y))$,
$h_5 := \lambda x\, some(h_7, \lambda y\, h_B(x)(y))$, $h_3 := dog$,
$h_7 := \lambda y\, (p(y) \& q(y))$, $p := \lambda y\, white(y)$, $q := \lambda y\, cat(y)$,
$h_4 := \lambda x \lambda y\, chase(x, y)\}$

This system of equations has exactly two solutions, in case it is extended, correspondingly, by adding the following assignments inside the scope of the recursion operator where:

1. $h_B := h_4, \quad h_A := \lambda x \lambda z\, h_5(x), \quad h_0 := h_1$;
2. $h_A := h_4, \quad h_B := \lambda z\, h_1, \quad h_0 := h_5$.

3 Basic Definitions of MRS

I will introduce MRS as currently given in Copestake et al., but in Moschovakis terms. A language of MRS includes, along other symbols and types, a set Rel of relation symbols. MRS, by using Moschovakis terminology, has two sorts of variables:

Definition 1 (Variables)**.**

– a set of *pure* variables:
$V_{pure} = x, y, z, \ldots$
Pure variables are to be quantified and for reference to individuals.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

– a set of *recursion* or variables:
  $V_{rec} = h_0, h_1, \ldots$
  Recursion variables are called also *location* in $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, or *labels* and *handles* in MRS.

In the above informal introduction of MRS representations, it is clear that the variables called handles and labels are of the same formal sort, similar to location variables in the language $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ of recursion. Notationally, in the MRS syntactic constructs, the distinction between handles and labels is with respect to the positions taken by these variables. Note that in MRS, 'location' variables are used for labeling EPs and for filling up scopal argument slots of relation symbols. In $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, location variables are used in the construction of recursive terms.

Definition 2 (Elementary predication (EP)). *This definition corresponds closely to the one given in Copestake et al. (p.12):*

$$\mathsf{label} : \mathsf{relation}(\mathsf{arg}_1, \ldots, \mathsf{arg}_n, \mathsf{sc\_arg}_1, \ldots, \mathsf{sc\_arg}_m), \text{ where}$$

1. $\mathsf{label} \in V_{rec}$ and is called the label of the EP;
2. $\mathsf{relation} \in Rel$ is an $(n + m)$-argument relation symbol;
3. $\mathsf{arg}_1, \ldots, \mathsf{arg}_n \in V_{pure}$ are the ordinary variable (i.e. non-scopal) arguments of relation;
4. $\mathsf{sc\_arg}_1, \ldots, \mathsf{sc\_arg}_m \in V_{rec}$ are the scopal arguments of relation.

Examples:

(10) a. $h : every(y, h_1, h_2)$
b. $h : sleep(x)$
c. $h : probably(h)$

Now, by considering the examples given in the paper, the above definition implies a wrong interpretation of the quantifier symbols like $some$ and $every$ as 3-place argument relations.

MRS has no λ-abstraction terms and no types corresponding to those of typed λ-calculus. Versions of CBLG similar to HPSG, use a SEM feature INDEX which, up to some extend, corresponds to λ-abstraction.

Revised[2] Definition 2 (Elementary predication)

$$\mathsf{h} : \mathsf{relation}(\mathsf{a}_1, \ldots, \mathsf{a}_n, \mathsf{h}_1, \ldots, \mathsf{h}_m), \text{ where}$$

1. $\mathsf{h} \in V_{rec}$ and is called the label of the EP;
2. $\mathsf{relation} \in Rel$ is a relation symbol;
3. $\mathsf{a}_1, \ldots, \mathsf{a}_n \in V_{pure}$ are variables which either fill up argument slots of the relation symbol relation or, in case of a quantifier relation, are the variables bound by it;
4. $\mathsf{h}_1, \ldots, \mathsf{h}_m \in V_{rec}$ are variables filling up the scopal arguments slots of relation.

Some Abbreviations and Notations: In MRS, If a variable $h \in V_{rec}$ labels an EP, it is called a *label*; if $h \in V_{rec}$ fills up an argument position of a relation, it is called a *handle*. A bag of EPs that have the same label is interpreted as a conjunction. A bag of co-labeled EPs $\mathsf{h} : \mathsf{E1}, \ldots, \mathsf{h} : \mathsf{E}_n$ is denoted by $\mathsf{h} : \mathsf{E1}, \ldots, \mathsf{E}_n$.

[2] I am giving a minimal revision in order to keep this introduction to MRS close to Copestake et al.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Definition 3 (*Immediate Outscoping* Relation between EPs)**.** Given a bag of EPs M, and two EPs $E, E' \in M$, E *immediately outscopes* E' iff one of the scopal arguments of E is identical to the label of E'. I.e., for some $p, p' \in Rel$ and $l, h \in V_{rec}$, $E = l : p(\ldots, h, \ldots)$ immediately outscopes $E' = h : p'(\ldots)$ and it is said that l immediately outscopes h and E'.

Definition 4. Given two conjunctions of EPs M and M', M *immediately outscopes* M' iff there are $E \in M$ and $E' \in M'$ such that E immediately outscopes E'.

Definition 5 (*Outscoping* Relation)**.** *Outscoping* relation over a set of EPs is the transitive closure of the immediate outscoping relation between EPs in that set.

Definition 6 (MRS Structure)**.** A MRS structure is a tuple $\langle GT, LT, R, C \rangle$, where

- R is a bag of EPs.
- $GT \in V_{rec}$ is a label (recursion variable) such that there is no label $h \in V_{rec}$ which occurs in R and outscopes GT. GT is called the *global top* of M.
- $LT \in V_{rec}$ is the topmost label in R, with respect to the outscoping relation over the labels in R, and which is not the label of a floating (see later) EP. LT is called the *local top* of M.
- C is a set of constraints (introduced later on) satisfied by the outscoping order in R.

SCIENCE AND TECHNOLOGY PUBLICATIONS

Definition 7 (Scope-resolved MRS Structure)**.** A scope-resolved MRS structure is an MRS structure such that:

1. The MRS structure forms a tree of EP conjunctions, where dominance is determined by the outscope ordering on EP conjunctions.
2. The global and local top labels and all handle arguments (i.e. recursion arguments) are identified with an EP label (in Moschovakis terminology: there are no free recursion variables).

4 Conclusions: Advancing New Developments

Arguments for "Flat" Semantic Representation that Pends Further Development. The original arguments for introducing MRS representation have been efficiency without loss of information. Further theoretic and development work is needed for the following initiations:

Underspecification. MRS permits underspecification in representations of quantifier scopes so that a single MRS construct represents multiple scopes without loss of grammatical information available in the structure of NL expressions.

Flat Semantic Representation. "Flat" MRS representations, which consist of the most basic facts, without loss of information, improve the efficiency of NLP. For example, flat representations have been favored in systems such as Generation from semantic representations and machine translation with semantic transfer.

Currently, MRS is in development stage for use in HPSG. For example, MRS has been extensively implemented in grammars for English, Norwegian and Danish by using LKB.

Further Development of MRS and L_{ar}^{λ} Representation.

Syntax-Semantics Interface. MRS representations can be written in a feature-value language for formal and computational grammar of NL, such asCBLG (e.g., HPSG). CBLG incorporates several linguistic components: vocabulary, lexicon, syntax, and semantic representations, in a unified way, which is a basis for compositional syntax-semantics interface.

Logic Foundation. MRS offers semantic representations which are close to the canonical forms of terms in the formal language of Acyclic Recursion. This gives opportunities to formalize MRS and develop new implementations by re-using existing CBLG resources.

Rendering. Render relation is the translation from NL into the language L_{ar}^{λ} of acyclic recursion. Feature-value lexicalist approach to grammar theory, such as CBLG (HPSG, LFG, etc.), is a good computational approach to syntax, which provides procedures for *rendering* into logic forms and is linguistically and semantically adequate.

Indexing. Indexing procedure from NL into L_{ar}^{λ} can be provided, e.g., by appropriately development of Binding Theory in CBLG (HPSG)

With appropriate adjustments of MRS, Moschovakis Acyclic Recursion can provide appropriate formalization of MRS. Using a version of Acyclic Recursion can contribute to developing MRS itself, for example, to:

1. formal representation of quantifiers;
2. representation of abstraction in MRS (resembling λ-abstraction);
3. finding a better incorporation of utterance and described situations into the MRS expressions (and MRS feature structures used in CBLG (HPSG)).
4. representing higher order relations for modifiers that are not conjunctively interpreted: *alleged, former, . . .* ;
5. representing higher order relations denoted by lexemes creating oblique contexts, such as *know, believe, ...*

Relation to other Type-theoretic Developments for NLP. In recent years, a powerful type-theoretical grammar formalism for natural, i.e. human, language processing (NLP), Grammatical Framework (GF), see [12] and [13], has been under active development. Future work, which is tightly related to the subject of this paper, is research on the placement of GF in the family of CBLG, with respect to syntax, semantics, and syntax-semantics inter-dependencies, its theoretic foundations, and applications.

The work on computational semantics presented in this paper is in a direction of theoretical developments, for providing foundations for, and extending, current applications in the lines of CBLG (e.g., HPSG and GF), and for new ones.

SCIENCE AND TECHNOLOGY PUBLICATIONS

References

1. Barwise, J., Perry, J.: Situations and Attitudes. Cambridge, MA: MIT press. Republished in 1999 by The David Hume Series of Philosophy and Cognitive Science Reissues. (1983)
2. Bunt, H.: Semantic Underspecification: Which Technique for What Purpose? In: Bunt, H., Muskens, R. (eds.): Computing Meaning, Vol. 3. Studies in Linguistics and Philosophy 83. Springer, Dordrecht (2007) 55–85
3. Copestake, A., Flickinger, D., Pollard, C., Sag, I. A.: Minimal Recursion Semantics: an Introduction. Research on Language and Computation 3.4 (2006) 281–332
4. Loukanova, R.: Generalized Quantification in Situation Semantics. In: Gelbukh, A. (ed.): Computational Linguistics and Intelligent Text Processing. Lecture Notes in Computer Science, Vol. 2276. Springer Berlin Heidelberg (2002) 173–210
5. Loukanova, R.: Typed Lambda Language of Acyclic Recursion and Scope Underspecification. In: Muskens R. (Ed.) Workshop on New Directions in Type-theoretic Grammars. (2007) 73–89
6. Loukanova, R.: Constraint Based Grammar and Semantic Representation with The Language of Acyclic Recursion. In: Bel-Enguix, G., Jiménez-López M.D. (Eds.): Proceedings of the International Workshop on Non-classical Formal Languages in Linguistics. (2008) 57–70
7. Loukanova, R.: Semantics with the Language of Acyclic Recursion in Constraint-Based Grammar. In: Bel-Enguix, G., Jiménez-López M. D. (Eds.): Bio-Inspired Models for Natural and Formal Languages. Cambridge Scholars Publishing. (to appear)
8. Montague, R.: Formal Philosophy. Thomason, R. H. (ed.): Selected papers of Richard Montague. Yale University Press. New Haven and London. (1974)
9. Moschovakis, Y. N.: A logical calculus of meaning and synonymy. Linguistics and Philosophy, Vol. 29 (2006) 27–89
10. Muskens, R.: Underspecified semantics. In: Egli, U, von Heusinger, K. (ed.): Reference and Anaphoric Relations. Studies in Linguistics and Philosophy, Vol. 72. Kluwer (1999) 311–338
11. Pollard, C., Sag, I. A.: Head-Driven Phrase Structure Grammar. Chicago: University of Chicago Press (1994)
12. Ranta, A.: Grammatical framework: A type-theoretical grammar formalism. Journal of Functional Programming, 44 (2004) 12–36
13. Ranta, A.: The GF Resource Grammar Library. Linguistic Issues in Language Technology, 2(2). (2009)
14. Sag, I. A., Wasow, Th., Bender, E. M.: Syntactic Theory — A Formal Introduction. 2nd edn. Stanford: CSLI Publications (2003)

SCIENCE AND TECHNOLOGY PUBLICATIONS