Acta Universitatis Apulensis,
ISSN 1582-5329, 2011,
Special Issue on Understanding Complex Systems,
Eds. Barna Iantovics, Ladislav Hluchý and Roumen Kountchev

SEMANTIC REPRESENTATION FORMS OF NATURAL LANGUAGE SENTENCES

Laszlo Kovacs

Abstract. The paper proposes a framework for sentence transformation module used in natural language processing. The transformation module performs mapping between the different information representation forms. The work focuses on two intermediate models, the role of predicate calculus formalism and the role of conceptual graph called ECG are investigated in details. Both models provide a semantic-based, language independent description of the environment. To demonstrate the functionality of the transformation module, the paper presents a pilot project that translates the sentences in Hungarian language into API function calls within a specific application domain.

Keywords: *natural language processing, conceptual graph, word grammar, semantic annotation*

2000 *Mathematics Subject Classification*: 68T50.

1. Introduction

The application of statistical learning methods is a very intensively investigated area of soft computing. The key question of this problem domain is how far the human intelligence can be modeled with computational and statistical methods. The goal of the paper is to investigate a characteristic phenomena of human beings namely the usage of a natural language. Most of the research activities on this field relate to natural language processing (NLP). In NLP systems, the key component is the transformation of sentences of NL into function calls of the execution engine. In order to provide a flexible and efficient sentence processing, the transformation is usually performed via some intermediate representation forms. These intermediate forms are used to describe the semantics of the sentences. The role of semantic description is very important on other fields of text processing too. For example, as the experiences in grammar induction show [12], the free text alone is not enough to learn the grammar at acceptable high level. On the other hand, the grammatical annotation requires a large amount of preprocessing and restricts the training pool. Our proposed approach is based on a model where the free text is annotated with a semantic, ontology description. The main benefit of this kind of semantic annotation over grammatical annotation is a greater efficiency of mapping semantics into syntax and it resembles more the human way of learning. In the literature, there is only few research works in this field, the interest of researchers for this topic raises only in the recent years [9]. The paper describes a framework model of the language interface engine including both the grammatical and the semantic components. The language interface engine can be used for two main activities: for the conversion of the sentences into a semantic graph models and to map semantic model to sentences. Within this frame, an important step is to find an appropriate formalism of the semantic model. According to our analyses, the traditional semantic modeling languages are not sufficient enough for reflect the process of conceptualization. The second base component is the grammar system that can be used to parse the incoming sentences or to generate the correct sentences.

2. Language models

In the everyday common sense, the language is the communication channel between humans to transfer some information from one person to other per-

sons. A key characteristic of the natural language is that it is strongly coupled with the semantics of the context world. The language is used to describe the semantics of our observations or thoughts. The semantics of human languages is a widely investigated area by linguistics and philosophers. We can refer here to the foundation work of Tarski [26].

The human language is a very flexible tool, it contains structures to approximate any kind of information we are aware. To provide this kind of high flexibility, the language has a modular structure: it has a base set of elements and a set of construction rules. The generated sentences carry the encoded information. In order to decode the information on the correct way, the language should use a fixed and shared set of rules. This rule system constitutes the grammatical, syntactical part of the language.

Thus a natural language can be considered as a pair of

$$\{SE, SY\}$$

where

- SE: semantic of the language,
- SY: syntax of the language.

Considering the semantic part, the model should describe a scenario that contains concepts describing entities and concepts describing relationships among the entities. The term relationship means here a broad definition including all type of relationship (like association, specialization, containment,..). The theory of semantics of natural language deals with giving meaning to every meaningful expression of the language. The corresponding grammar should give an exact mapping of the scenarios into sentences. The language is considered as set of sequences of characters forming words and sentences. In that way, the language is equal to the set of valid sentences. The grammar constraints the set of character sequences to valid sequences. A grammar usually encodes the semantics on the following ways:

- the base entity concepts are encoded with word stems
- the base relationships are encoded with
  - ordering of the words, or

  - inflection of the words, or
  - attaching modifiers

- complex concepts and relationships are encoded with sentences or compound words

In the case of spoken languages, there are additional ways to encode the information like accent or tune.

According to [18], an acceptable theory on semantics of the language should determine the meaning of every sentence by analyzing it as composed of elements drawn from a finite stock. He argues that the syntax of the language should be formalizable that every valid expression may be analyzed as formed from elements in vocabulary by application of grammar rules.

Some frequently occurring situations may be composed to special phrases and these phrases may be assigned a special meaning, independently from the usual general grammar. For example, the phrase 'white paper' does not refer to a piece of paper in color white. Beside the handling of phrases, there are many other difficulties in managing the grammar of natural languages. Some key problems:

- ambiguity of the mapping (the same word can be mapped to different concepts)
- missing components (the default elements of the context are not given explicite)
- scoping problem (the scope of some expression is not known)
- the terms have fuzzy meaning

There are several approach developed to solve this kind of problems, for example the work of Zadeh [19] proposes fuzzy term representation and fuzzy logic to describe the meaning of human sentences.

3. Hopl formalism

The semantics of natural language is concerned with the relation between language and the world. Hence, the meaning of a sentence determines the conditions under which it is true. Since, by definition sentences are finite sequences of words (which are the basic semantic units), and as a consequence of

the recursive nature of language, the meaning of a word will determine what contribution it makes to the truth conditions of the sentences in which it occurs [20]. This is called the principle of compositionality. For correct interpretation, however, we also need to have world knowledge. Without context, that is without defining the domain of discourse, many human language sentences could be assigned several meanings. This ambiguity may result from the lexical ambiguity of words, or from the syntactic ambiguity of sentences (word combinations). In other words, NL sentences build up of word constituents bearing a set of possible meanings which are made concrete by the actual context.

Thus, analogously to FOPL (first order predicate logic) syntax and semantics, the syntax of NL defines the set of well-formed grammatical sentences (WGSs), while its semantics determines the truth value of a WGS in a given interpretation. First, we intend to examine the semantic equivalence of NL and FOPL statements with true logical value. According to the definition of logical equivalence, this examination requires an interpretation $I = \langle \Delta, I \rangle$, a variable assignment φ and two formulas: an NL WGS and an FOPL WFF. Let us suppose that the content words (those with lexical meaning) constituting the NL sentence comprise the interpretation domain. The proposition(s) expressed by the NL sentence will be the predicate(s) of the FOPL formula, while the other constituents will be assigned to FOPL variables.

An important question in our discussion is to what extent rendering NL sentences into logical notation should reflect the logical forms of those sentences. That is to say, it is one thing for a sentence to be rendered into a logical formula, and quite another for the sentence itself to have a certain logical form [21]. The difference is evident if we consider the following examples.

- a: There are students.
- b: $(\exists x)S(x)$.
- c: Some students are foreigners.
- d: $(\exists x)(S(x).F(x))$.
- e: $(\exists x)(F(S(x)))$.

The sentence (b) reveals the true structure of the existential proposition (a) expresses. Thus this logical form of the sentence shows inherent properties of the sentence itself, therefore we can refer to it as a level of syntactic structure.

On the other hand, (c) does not express existential proposition and it does not contain any sentential constituent corresponding to the conjunction in (d). On the other hand, (e) is not a valid FOPL statement although its approach mirrors the true structure of the NL sentence.

In our discussion, we restrict our attention to logical forms reflecting syntactic structure. For handling discrepancies, we give the definition of the equivalence of two different notation systems by introducing the definition of a composition preserving transformation. Given two languages $L_1(F_1, O_1)$ and $L_2(F_2, O_2)$, where F denotes the set of formulas and O denotes the set of operations over F, the transformation $\tau : L_1 \rightarrow L_2$ is said to be composition preserving if

$$\tau(o(f_1, f_2, ...)) = \tau(o)(\tau(f_1), \tau(f_2), ...),$$

i.e. $\tau(o(f_1, f_2, ...))$ and $\tau(o)(\tau(f1), \tau(f2), ...)$ are equivalent in all interpretations. Without the criterion of composition preserving, an $ST(w_1, w_2, ...)$ general FOPL predicate could be assigned to any arbitrary $s = w_1, w_2, ...$ NL sentence. In this case however, the semantic interpretation of the FOPL formula is not easier than that of the NL sentence.

Let consider the following examples from the viewpoint of composition preserving:

- a: You know I like sports.
- b: Know(you, Like(I, sports)).
- c: KnowLike(you, I, sports).

Here, the logical counterpart (b) of the NL sentence is not a valid FOPL formula, because predicates are not allowed to be arguments of other predicates. On the other hand (c) is a well-formed FOPL formula, but it is not composition preserving. Barwise and Cooper [22] have shown, that the notation of FOPL is not adequate for symbolizing such quantificational expressions as most, many, several, few (not mentioning numerical quantifiers and more complex quantificational expressions). Thus, the sentences of NL can not be mapped to FOPL expressions unambiguously and without loss of information. Let us take another example to show this problem.

- a: Every student read a book (over the vacation).
- b: $(\forall x)(\exists y(S(x) \wedge B(y) \wedge R(x, y))$.

- c: $(\exists y)(\forall x(S(x) \wedge B(y) \wedge R(x,y))$.

We can render the NL sentence either as every student read a separate book as in (b), or as every student read the same book as in (c). This phenomena is known as scope ambiguity, and results from the fact that NL, in opposition to FOPL, is structurally ambiguous [6]. From this analysis we can conclude that the semantic content of FOPL expressions may be narrower than that of NL.

Some papers propose special techniques within FOPL to overcome of these difficulties. For example, reification [10] is a technique used for representing all concepts that one wants to make statements about as objects in FOPL, instead of using higher-order predicates. In this case, however, new relations need to be introduced which in fact do not solve, but only shift the problem to another level. Moreover, the resulting valid FOPL formulas will not be in accordance with the precondition of composition preserving. The decomposition theory [2] states that FOPL is sufficient, since HOPL formulas can be converted into FOPL formulas. In the proposed formalism, an arbitrary $P_1(P_2(x))$ second-order statement can be transformed into a $P_1(y_1) \wedge P_2(y_2) \wedge PR(y_1, y_2, x)$ FOPL formula; while $\forall p.P(x)$ is rendered into $\forall y.P(y) \rightarrow PR(y,x)$. Although, this solution formally results in valid FOPL formulas, but the criterion of composition preserving is violated again.

As a consequence, in order to describe the semantics of NL sentences, we have introduced the higher-order calculus and numerical primitives as representation formalism.

The most obvious differences between HOPL and FOPL are that 1) HOPL uses variables that range over sets instead of discrete variables; and 2) in HOPL predicates can be used as arguments of other predicates or values of variables. In other words, higher-order logics allow for quantification not only of elements, but also of subsets, or of sets of such subsets, and of other objects of higher type (such as relations between relations, functions from relations to relations between relations, etc.). Although higher-order logics are more expressive, allowing complete axiomatizations of structures, they do not satisfy analogues of the completeness and compactness theorems from first-order logic, and are thus less amenable to proof-theoretic analysis [23]. In general, a higher-order predicate of order n takes one or more $(n-1)^th$-order predicates as arguments, where $n > 1$. For further information about second-order logic and its comparison with firstorder logic we refer the reader to [25].

For the composition preserving logical representation of the examined linguistic phenomena we propose the following HOPL extensions.

1.) Arbitrary predicates (relations) are allowed, denoted by capitalized words. Domain types are assigned to the arguments of predicates, which specify the semantic roles these arguments play. The fixed set of roles (analogously to thematic roles [20] in linguistics) are associated with and determined by the predicate.

- a: Peter loves Mary.
- b: Love(Subject: Peter, Object: Mary).

2.) Concepts are regarded as sets. Constants referring to specific objects (concepts) are single-element sets denoted by capitalized words, while constants referring to abstract concepts are multiple-element sets denoted by lower-case words. An element of a set a is denoted by the isa(a) function (functions are denoted by lower-case words). We can refer to an object by the : operator.

- a: Peter reads a book.
- b: Read(Subject: Peter, Object: isa(book)).

From the set-based treatment of concepts follows that plural forms, when used for referencing objects in general, are represented as abstract concepts, i.e. multiple-element sets.

3.) By the representation of adverbs we should make a distinction between those that describe the circumstances of the action or state expressed by the predicate, and those that add extra conditions connected with the basic assertion. The latter is represented by the use of the Happens relation.

4.) Adjectives can be added to the assertion by the use of the Property relation.

- a: Peter reads a scientific book.

- b: Read(Subject: Peter, Object: isa(book):x | Property(Subject: x, Object: Scientific)).

5.) For the treatment of numerical expressions we need to introduce numerical relations and numerical primitives, as well as the some function for creating a group of objects.

- a: Peter reads two books.
- b: Read(Subject: Peter, Object: some(isa(book)):x | Property(Subject: x, Object: Two)).

6.) Existential and universal quantifiers are defined similarly by means of the some and all functions, respectively.

7.) Logical operators can be applied to predicates or to arguments of predicates.

9.) The examination of causes and results leads us back again to the Happens relation.

- a: Peter can not sleep because Tom is dancing.
- b: Happens(Cause: Dance(Subject: Tom), Result: NotSleep(Subject: Peter)).

From this analysis we can see, that in view of the criterion of composition preserving, the extended HOPL approximates NL better than the one without these extensions.

4. Extended conceptual graph

The goal of the investigation is to develop a conceptual modeling language that can be used to describe the semantics of an agent's internal conceptual model. The model language should support a formal specification that enables the mapping of semantics into a symbolic representation of the entire conceptualization process. The analysis of existing conceptual models shows that they all have some shortcomings from the aspects of our requirements. In

order to provide a model language with rich set of specific features, a specific knowledge representation model was developed. It contains beside the usual modeling elements, like the specialization relationship, additional elements to enable a more efficient and powerful description of the conceptualization process.

The proposed extended conceptual graph (ECG) model contains three primitive-types: concept, relationship and container. Based on the behaviors, the following concept subtypes are defined:

1. According to their grade of identification
    - (N) Noname concept: is a primary concept that has no context-unique identification name.
    - (R) Permanent-named concept: is a concept having a context-unique name. A permanent concept is associated with an implicit definition that enables the identification of its instances in the environment.
    - (T) Temporary-named concept: is a concept occurring in some previous snapshot(s) of the history as a noname concept
2. Categories on a logical basis
    - (P) Predicate concept: is a concept that is used to denote predicates that are usually given by verbs in sentences. Predicate concepts are the kernels of the model fragments.
    - (C) Category concept: is the term covering all non-predicate concepts. Category concepts can denote various attributes for example. Each category concept defines a subset of instances that match this category concept.
3. According to the model level
    - (F) Primary concept: is a concept at the instance level. Primary concepts correspond to instances of the agent's environment.
    - (A) Abstract (derived) concept: a higher level concept in the agent's extended knowledge model. The derivation rule is defined with a sequence of snapshots.

4. Categories on cardinality
    - (I) Single instance concept: only one object is identified by this concept
    - (M) Group concept: several instances can belong to the concept. There are different types of group defined, like AND or OR groups.

Due to the semantic integrity constraints, only the following concept types are allowed: FICN, FICT, FICR, FMCR, FMPR, AMCR and AMPR. Regarding the relationship type, it has the following categorization:

1. According to the model level:
    - (F) Primary relationship: is a relationship that can be recognized by the agent. It is detected usually in the environment.
    - (A) Abstract (derived) relationship: is a relationship that is based on primary relationships. The derivation rule is defined with a sequence of snapshots.
2. According to the logical level
    - (I) Specialization relationship: is equivalent to the usual ISA relationship. It provides inheritance. A concept may have multiple parents.
    - (R) Role relationship: arbitrary attribute of a predicate concept
3. categories on cardinality
    - (S) Single instance relationship: only one object cluster is identified by this relationship
    - (M) Class relationship: several instances can belong to the relationship.

The allowed relationships types are FMI, FSR, FMR, AMR. The group of container elements includes structure modules, like model fragment, model history.

Beside the mentioned semantic elements that refer to the state the environment to be observed, some other such elements can be included into the model

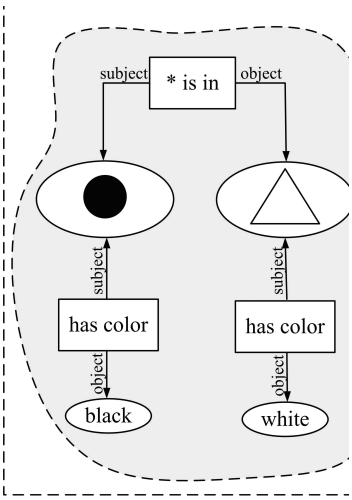

Figure 1: The ECG model

that are related to the agent itself. These semantic elements describe internal state of the agent. The state of these internal attributes is also converted into the output sentence. The following parameters are used to describe the internal state:

- Mode (or indent): statement, open question, closed question, imperative,
- Timestamp: it denotes the time of the snapshot.

The Figure 1 shows a sample semantic graph that describes a simple test world.

During the processing of the ECG, the base units of the graph are the ECG atoms. An ECG atom corresponds to a primitive statements related to one predicate. It has a structure of one-level deep tree, where the root of the tree is the predicate and the concepts linked to it are the leaves. The child concept of the root predicate may be not only a single concept but it can be another ECG atom. Thus, the ECG atoms can be linked into a hierarchy of ECG atoms. Our model is based on the assumption that at words at a sublevel do not contain words from the upper level. The whole sublevel is considered as an atom at the parent level. This assumption improves significantly the modularity and the efficiency of the grammar system.

5. Word graph and dependency grammar

Grammars, and theories of grammar, can be classified according to whether the basic unit of sentence structure is the phrase, or the dependency between two words. A dependency-based linguistic approach to the description and analysis of natural language syntax is constituted by distinguishing a head-dependent asymmetry, and describing the relations between a head and its dependents in terms of semantically motivated dependency relations. Dependency grammar (DG) is a class of syntactic theories developed by the French linguist Lucien Tesnire [1]. His model is based on the stemma, a graphical representation of the grammatical dependencies between the words in a syntactic construction. In the sentence, the verb is seen as the highest level word, governing a set of complements, which govern their own complements themselves. Tesnire has had a major influence on linguistic theories that place more emphasis on semantics than on syntax. Klein and Simmons [2] adopted dependency grammar for a machine translation system. Valency theory [3] and Meaning-Text Theory [4] are two ongoing developments of the dependency approach. Schank adopted the dependency approach in his Conceptual Dependency Graph, but he shifted the emphasis to concepts rather than words [5]. For a recent survey on dependency models see [6]. Dependency theories have also been strongly influenced by Case Grammar [7] which provides a convenient set of labels for the arcs of the graphs. Extensible Dependency Grammar [8] and Word Grammar [9] are two general frameworks for dependency grammar which aim at modeling not only the syntactic but also the semantic and phonological levels of linguistic representations.

The main reason why we turn our attention towards dependency structures is that in a phrase structure tree, discontinuous constructions can not be represented. This restriction poses problems for the analysis of word order variation, even for rigid word order languages. On the other hand, dependency grammars are not defined by a specific word order, and are thus well suited to languages with freer word order, such as Hungarian, as well as Scandinavian and Slavic languages. The modeling of other European languages where discontinuous constructions are frequent, such as German, French and Dutch can also benefit from a dependency-based approach.

Dependencies are widely accepted in theories of semantic structure. In fact, one of the main attractions of traditional DG is its close correspondence to meaning:

- syntactic dependencies are meaningful, i.e. they usually carry semantic relations; and
- syntactic dependencies are more abstract than surface order.

Nevertheless syntactic dependencies are distinct from semantic dependencies. The usual problem is how to map syntactic dependency structure to a semantic one. A distinguishing feature of our project is the aim of finding a mapping from semantic (conceptual) dependency structure to a syntactic one, where consequently the elements among which dependencies are examined are not words but concepts and syntactic units, respectively. Hence, our model got the name Conceptual Dependency Grammar (CDG)

The word graph is used to describe the words and the dependency between the words. A word w_1 depends on w_2, if w_1 can occur in the sentence only if $w2$ also occurs. The dependency is denoted with

$$w_2 \Rightarrow w_1.$$

Taking a sample sentence 'The cat is sitting in a chair.' the dependency graph has the following elements:

- is, sitting ⇒the, cat, in, a. chair
- cat ⇒ the
- chair ⇒ in, a

For the detection of dependency relationships, the sentence reduction method is applied. This means, that some word or words are omitted from the sentence. The meaning of the altered sentence is evaluated by a teacher. The new sentence may be judged as

- C: correct, but it has a reduced meaning compared with the original sentence, or as
- U: grammatically not correct, but understandable with a modified, reduced meaning and or as
- N: the sentence is not correct and not understandable.

Some test sentences with their evaluations are given here for demonstrations purposes:

- The cat is sitting in a : N
- The is sitting in a chair :N
- The cat is sitting : C
- Cat is sitting chair : U

Considering the representation tools of the language, the grammar system should contain elements to describe the following information items:

- set of the stem words;
- inflection of the words (the suffixes and prefixes alter the meaning of the stem);
- order of the words (there may be a fixed order between the words belonging to different semantic or grammatical units);
- decomposition rules of grammatical units (the chain of derivation rules assigns a sequence of words to a semantic unit).

One of the main difficulties of the grammar system is that very different semantic elements may affect the same syntactical unit. For example, the inflection rule of a verb in the Hungarian language may depend on the subject, on the object, on the timestamp and on the mode. Another difficulty is the fact, that a semantic unit may be mapped not only to a single word but into a word set. For example, some verb have prefix that are in some situation separated from the stem and in other situations it is merged with the stem.

The proposed grammar system should contain elements to decode the required conversion rules. The main components of the grammar system are:

- a set of stem words that are assigned to abstract concepts with a specialization graph (like: Peter, read, book, evening,..). The concepts may be either semantic concepts or grammatical concepts (car or subject).
- decomposition rules that describe the components of an abstract concepts. The components may be abstract or atomic, word level concepts. (sentence: subject, predicate, object,..).

- set of inflection rules where different grammatical units may have different transformation rules (accusative: Rule1, dative: Rule2). A rule is represented here as a string transformation function, where the basic string operations are the followings:
  - concatenation with a given suffix
  - substitution with a given infix where both arguments may be an empty string
  - concatenation with a given prefix.
  - a set of ordering rules where the nodes refer to grammatical units and the edges are directed and labeled. The label denotes the type of precedence rule.

In this investigation, the grammar rules are known and defined. The grammar is embedded in a module, so a semantic graph can be pressed with different grammar modules.

6. Structure of the pilot system

The goal of the pilot system is to translate incoming natural language sentences into API function calls within a special application domain. The pilot system is developed for the Hungarian language that has a more complex grammar system than the most widely investigated English language has. The Hungarian language is an agglumerative one, a stem word may have several hundred derivations.

The main steps of the mapping function can be summarized as follows:

- Parsing the input sentence into words
- Performing a morpheme analysis on the words
- Determining the stems and the inflection classes for each words
- Matching the annotated words to the nodes of the word graph
- Checking the ordering constraints given in the graph
- Calculating the similarity value between the sentence and the word graph

- Selection of the winner graph
- Returning the corresponding conceptual graph as meaning descriptor of the sentence
- Mapping the winner conceptual graph to an API function
- Mapping the graph nodes to arguments of the API function
- Sending the API function to the execution engine

In this processing a new key element is the morpheme analyzer that contains a grammar specific engine to determine the stems and inflection parts. In the frame of the project a Morpheme Analyzer Module for the Hungarian Language was developed. For a given sentence, the analyzer returns the list of possible morpheme spectrums of the words. With coupling of two directions a translator module can be developed. The grammars of the input and output channels may be different. Thus, a sentence in the first grammar is translated into a sentence of the second grammar.

In the project, the goal was to develop a natural language interface for a service provider. The server module has an application programming interface, a set of functions that should be invoked to start the required service. The front end of the pilot system uses the Hungarian language and the output contains the predicates symbolizing the function calls.

The engine selects the best fitting conceptual graph for the incoming sentence and returns the predicate representation of the selected conceptual graph. The Figure 2 shows the input form with the input sentence (Mit olvas Peter?, What is reading Peter?) and the output form with the generated formula (READ(Peter,?)). The predicate formula can be converted into a function call on a straightforward way.

7. Conclusions

A key element of NLP systems is the transformation of incoming sentences into low level API calls. The paper proposes a framework for sentence transformation that includes formalism based on high order predicate calculus. The HOPL formalism is implemented with an ECG conceptual graph. For the grammar module, the system proposes a grammar based on word-dependency approach. The functionality of the framework is demonstrated with a pilot NLP module for the Hungarian language.

Figure 2: The intput and output forms of the transformation engine

References

[1] L. Tesnire, ElEments de syntaxe structurale, Paris: Klincksieck, 1959.

[2] S. Klein and R. F. Simmons, 'Syntactic dependence and the computer generation of coherent discourse,' Mechanical Translation 7, 1963.

[3] D. J. Allerton, Valency and the English Verb, New York: Academic Press, 1982.

[4] J. Steele Ed., Meaning-Text Theory, Ottawa: University of Ottawa Press, 1990.

[5] R. Schank Ed., Conceptual Information Processing, Amsterdam: North-Holland Publishing Co., 1975.

[6] R. Hudson, 'Recent developments in dependency theory,' in Syntax. Ein internationales Handbuch zeitgenssischer Forschung, J. Jacobs, A. v. Stechow, W. Sternefeld and T. Vennemann Eds., pp. 329-338, Berlin: Walter de Gruyter, 1993.

[7] C. J. Fillmore, 'The case for case,' in Universals in Linguistic Theory, E. Bach and R. T. Harms Eds., New York: Holt, Rinehart and Winston, pp. 1-88, 1968.

[8] R. Debusmann, 'Extensible Dependency Grammar: A modular grammar formalism based on multigraph description,' PhD thesis, 2006.

[9] R. Hudson, Language Networks: The new Word Grammar, Oxford University Press, 2007.

[10] D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to NLP, Computational Linguistics, and Speech Recognition, Prentice Hall, 2nd ed., 2007.

[11] M. Ross Quillian, 'Word concepts: a theory and simulation of some basic semantic capabilities", in Behavioral Science, vol. 12, pp. 410-430, 1967.

[12] G. B. Varile, A. Zampolli, R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen and V. Zue, Survey of the State of the Art in Human Language Technology, Cambridge University Press, 1997.

[13] E. Charniak, Statistical Language Learning, MIT Press, Cambridge, MA, 1996.

[14] C.D. Manning and H. Schtze, Foundations of Statistical Natural Language Processing, MIT Press, Cambridge, MA, 1999.

[15] A. Clark, Unsupervised Language Acquisition: Theory and Practice, PhD Dissertation, COGS, University of Sussex, 2001.

[16] A. Roberts and E. Atwell, Unsupervised Grammar Inference Systems for Natural Language, Research Report 2002.20, University of Leeds, School of Computing, 2002.

[17] A. McEnery, R. Xiao and Y. Tono, Corpus-Based Language Studies: An Advanced Resource Book, in ser. Routledge Applied Linguistics, Routledge, 2005.

[18] D. Davidson: Semantics for Natural Languages, in Truth, Meaning and the Indeterminancy of Translation, 1970, pp 57-65

[19] L. Zadeh: Fuzzy Logic and Approximate Reasoning, in Synthese, 30(3-4), pp 407-428, 1975

[20] P. Blackburn, J. Bos: Computational semantics. Theoria, 18, pp 2745., 2003

[21] K. Bach: A Companion to Philosophical Logic, chap. Language, logic, and form. Blackwell Publishers, 2002.

[22] J. Barwise, R. Cooper: Generalized quantifiers and natural language. Linguistics and Philosophy, 4, 159219., 1981

[23] S. Shapiro: The Blackwell Guide to Philosophical Logic, chap. Classical logic II: Higher order logic. Blackwell Publishers, 2001.

[24] J. Higginbotham, R. May: Questions, quantifiers and crossing. Linguistic Review, 1, pp. 4179., 1981

[25] M. Rossberg: First-order logic, second-order logic, and completeness. In H. et al (ed.), First-Order Logic Revisited, Logos Verlag Berlin, pp. 303 321., 2004

[26] A. Tarski: Logic, Semantics, Metamathematics, Hackett Publishing, 1983

Laszlo Kovacs
Department of Information Technology
University of Miskolc, Hungary
H-3515 Miskolc-Egyetemvaros
email:*kovacs@iit.uni-miskolc.hu*