
Production Systems and Information Engineering
Volume 5 (2009), pp. 19-39.

SEMANTIC REPRESENTATION OF NATURAL LANGUAGE WITH EXTENDED CONCEPTUAL GRAPH

Erika Baksa-Varga
University of Miskolc, Hungary
Department of Information Technology
vargae@iit.uni-miskolc.hu

László Kovács
University of Miskolc, Hungary
Department of Information Technology
kovacs@iit.uni-miskolc.hu

[Received February 2009 and accepted May 2009]

Abstract. We intend to create an intelligent agent that is able to detect the objects and relations in its environment, and based on the received information is able to build up an internal knowledge base on the basis of which linguistic expressions can be formulated. The present investigations focus on modeling the agent's ability to assign semantic representations to its observations. For this purpose we have developed a graphical conceptual modeling language. In this article we examine the model's expressive power, that is to what extent it is able to model natural language semantics. The analysis is performed in comparison with predicate logic. In our view, first-order logic does not provide sufficient means for the translation of natural language expressions and we will argue that also higher-order logic needs some extensions.

Keywords: natural language semantics, semantic representation of languages, predicate logic, Extended Conceptual Graph

1. Introduction

The final goal of our research is to simulate a human agent who perceives signals from the environment and after processing the received information, is able to express the observations with linguistic symbols. We intend to create an intelligent agent – having the cognitive abilities of pattern recognition, association and generalization – that is able to detect the objects and relations in its environment, and based on the received information is able to build up an internal knowledge base on the basis of which linguistic expressions can be formulated. The schematic design of the system can be found in Figure 1.

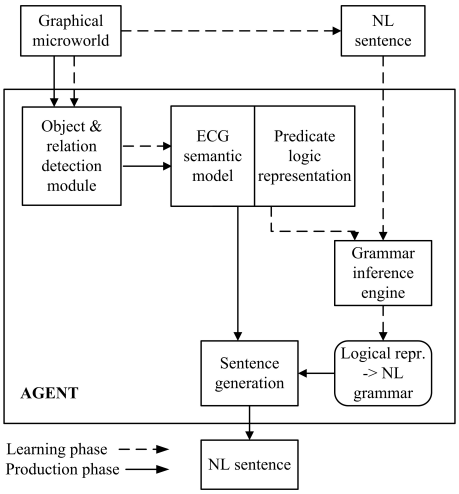

Figure 1. Operational system design of the agent

The project is strongly related to natural language processing (NLP). In NLP, the inference and adequate representation of natural language (NL) semantics is a crucial issue. A relatively new discipline, computational semantics has the aim to find techniques for automatically constructing semantic representations for expressions of human language. For this, the most basic issue is to agree on a semantic representation language. Practically, this should be a formalism with high expressive power and computational effectiveness. "In philosophy and linguistics the predicate calculus is used for analyzing the semantics and logic of natural language... The way expressions and structures contribute to the meaning of a natural language sentence is supposed to be determined and shown by means of its translation into the calculus [1]." The advantages of predicate logic (PL) are the use of a simple and exact notation and interpretation system, the standard formalism and general applicability, the ability for reasoning and rule validation, and its convertability to other symbolisms.

Despite the extensive research on representing NL semantics with PL, the relationship of the two systems is still not fully revealed. It is primarily a consequence of the NL system's complexity and ambiguity. Opinions in the related literature are quite diverse in terms of the suitability of first-order versus higher-order logic for NL representation. For example, [2] argues that first-order predicate logic (FOPL) offers an attractive compromise between the conflicting demands of expressivity and inferential effectiveness, and shows techniques for translating away modalities from intentional and temporal linguistic phenomena, as well as for handling plurals in FOPL. Also, [3] states

that FOPL is sufficient, since higher-order logical statements can be converted to FOPL formulas. On the other hand [4] and [5] prove the necessity of higher-order predicate logic (HOPL) in the field of quantification, while [1] makes a point on the unsuitability of FOPL in terms of representing plural referring expressions. In our view, FOPL does not provide sufficient means for the translation of NL expressions (see the analysis in Section 3) and in this article we will argue that also HOPL needs some extensions.

The article is organized as follows. In Section 2 we give the formal definition of the syntax and semantics of first-order predicate logic, then that of natural language. In Section 3 we examine the semantic equivalence of the NL and PL notation systems. For this purpose, we define first the semantic equivalence of two statements introducing the notion of composition preserving transformation. After that we analyze the properties of a semantic equivalence assignment between the two systems. In Section 4 we argue that first-order logic does not provide sufficient means for the representation of NL semantics, therefore we need to go beyond it. Higher-order logic is introduced, and we show that it also needs some extensions. In Section 5 we define an extended HOPL model and study the semantic equivalence assignment between this model and the NL system. In Section 6 we introduce the Extended Conceptual Graph (ECG) model as the graphical representation of the given extended HOPL model. Finally, we sum up the results of the present investigations concerning the expressiveness of the ECG model.

2. Formalizing Semantic Interpretation

2.1. Formal Definition of First-Order Predicate Logic

FOPL is a flexible, well-understood and computationally tractable approach to knowledge representation [6], which uses a wholly unambiguous formal language interpreted by mathematical structures. It is a system of deduction that extends propositional logic by allowing quantification over individuals of a given domain of discourse. The syntax of FOPL is built up of a vocabulary consisting of non-logical and logical symbols over a Σ alphabet. The set of non-logical symbols includes function symbols with a fixed arity ≥ 0, and the collection of variable and constant symbols. The set of logical symbols comprises predicate symbols with a fixed arity ≥ 0, the boolean connectives ($\wedge$, $\vee$, $\neg$, $\rightarrow$), and the quantifiers ($\forall$ and $\exists$). As its name implies, FOPL is organized around the notion of predicate. Predicates are symbols that refer to the relations that hold among some fixed number of objects in a given domain. Objects are represented by terms, which can be defined as constants, functions

or variables. FOPL constants refer to exactly one object, and are conventionally depicted as single capitalized letters. Functions also refer to unique objects, while variables, which are normally denoted by single lower-case letters, allow us to make statements about unnamed objects (free variables) and also to make statements about some/all objects in some arbitrary world being modelled (bound variables in the scope of a quantifier). Formally,

- all variable symbols are terms;
- if $t_1, \ldots, t_n$ are terms and f is a function symbol with arity n, then $f(t_1, \ldots, t_n)$ is also a term.

A statement is expressed in the form of formulas, which are defined as follows:

- If p is a predicate symbol with arity n, and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is an atomic formula.
- If t_1 and t_2 are terms, then $t_1 = t_2$ is an atomic formula.
- If α and β are formulas, then so are $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \rightarrow \beta)$.
- If α is a formula, and x is a variable, then both $\forall x.\alpha$ and $\exists x.\alpha$ are formulas.
- A sentence is a formula without free variables.

The syntax of FOPL defines the set of well-formed formulas (WFFs), while the semantics of FOPL determines the truth value of an arbitrary formula in a given model or interpretation (an abstract realization of a situation). Formally, an interpretation $\mathfrak{I} = < \Delta, I >$ consists of a domain Δ and an assignment function I which assigns

- an f^I function with arity n to every function symbol f with arity n, where: $f^I : \Delta \times \ldots \times \Delta \mapsto \Delta$, and
- a p^I relation with arity n to every predicate symbol p with arity n, where: $p^I \subseteq \Delta \times \ldots \times \Delta$.

With the aid of interpretation an element of Δ can be assigned to every variable-free expression. Similarly, a truth value can be assigned to every sentence. For the interpretation of expressions with variables, and formulas with free variables a variable assignment function is required. This φ function assigns an element of Δ to each variable symbol x, so that $\varphi(x) \in \Delta$. Given an interpretation $\mathfrak{I} = < \Delta, I >$ and a variable assignment φ, the $t^{\varphi,I}$ meaning of an arbitrary term t is defined as:

- If x is a variable, then $x^{\varphi,I} = \varphi(x)$.
- If $t_1, \ldots, t_n$ are terms and f is a function symbol with arity n, then $f(t_1, \ldots, t_n)^{\varphi,I} = f^I(t_1^{\varphi,I}, \ldots, t_n^{\varphi,I})$.

Given an interpretation $\mathfrak{I} = < \Delta, I >$ and a variable assignment φ, the truth value of an arbitrary α formula is defined as $\mathfrak{I} \models_\varphi \alpha$, that is the interpretation satisfies the formula. Regarding the different types of formulas this definition is the following:

- $\mathfrak{I} \models_\varphi p(t_1, \ldots, t_n)$ iff $\langle d_1, \ldots, d_n \rangle \in p^I$ and $d_i = t_i^{\varphi, I}$.
- $\mathfrak{I} \models_\varphi t_1 = t_2$ iff $d_1, d_2 \in \Delta$ and for both $d_i = t_i^{\varphi, I}$ where $d_1 = d_2$.
- $\mathfrak{I} \models_\varphi \neg\alpha$ iff not $\mathfrak{I} \models_\varphi \alpha$.
- $\mathfrak{I} \models_\varphi \alpha \wedge \beta$ iff $\mathfrak{I} \models_\varphi \alpha$ and $\mathfrak{I} \models_\varphi \beta$.
- $\mathfrak{I} \models_\varphi \alpha \vee \beta$ iff $\mathfrak{I} \models_\varphi \alpha$ or $\mathfrak{I} \models_\varphi \beta$.
- $\mathfrak{I} \models_\varphi \alpha \rightarrow \beta$ iff not $\mathfrak{I} \models_\varphi \alpha$ or $\mathfrak{I} \models_\varphi \beta$.
- $\mathfrak{I} \models_\varphi \forall x.\alpha$ iff for all $d \in \Delta$ $\mathfrak{I} \models_{\varphi[x \mapsto d]} \alpha$.
- $\mathfrak{I} \models_\varphi \exists x.\alpha$ iff for some $d \in \Delta$ $\mathfrak{I} \models_{\varphi[x \mapsto d]} \alpha$.

Where $\varphi[x \mapsto d]$ is the variable assignment which assigns $d \in \Delta$ to x, while assigning the same value to every other variable as φ [7].

2.2. Formal Definition of Natural Language

The term 'natural language' (NL) refers to human languages which are not consciously invented, but naturally acquired by humans. All written natural languages build up of sequences of words which are finite sequences of symbols over a given alphabet. Syntax is the term which defines the set of rules telling us how words may be combined to form sentences. Formally,

- the main building blocks of NLs are characters (symbols) $c \in \Sigma$, where Σ denotes the finite character set (alphabet) of the language;
- words are finite sequences over Σ, that is every $w \in W \subseteq \Sigma^*$, where W denotes the set of words;
- sentences are finite sequences over W, that is every $s \in S \subseteq W^*$, where S denotes the set of sentences.

NLs are infinite recursive systems, hence on the basis of understanding a finite number of words we can understand and construct an infinity of sentences recursively applying the rules of syntax [8].

NL semantics 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 [2]. 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 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. An interpretation $\mathfrak{I} = < D_O, I >$ consists of a domain of objects D_O and an assignment function I which assigns

- an f^I function with arity n to every function symbol f with arity n, where: $f^I : D_{O_1} \times \ldots \times D_{O_m} \mapsto D_O$, and
- a p^I relation with arity n to every predicate symbol p with arity n, where: $p^I \subseteq D_{O_1} \times \ldots \times D_{O_m}$.

With the aid of interpretation an element of D_O can be assigned to every WGS constituent.

3. Examining the Semantic Equivalence of NL and PL Systems

3.1. Formal Definition of Semantic Equivalence

By definition, two statements in the same system are logically equivalent if, for all possible values of the variables involved, both statements are true or both are false. If α and β are equivalent, we write $\alpha \equiv \beta$. Formally, given an interpretation $\mathfrak{I} = < \Delta, I >$ and a variable assignment φ, formula α and formula β are equivalent if

- $\mathfrak{I} \models_\varphi \alpha$ and $\mathfrak{I} \models_\varphi \beta$, or
- not $\mathfrak{I} \models_\varphi \alpha$ and not $\mathfrak{I} \models_\varphi \beta$.

3.2. Semantic Equivalence of NL and FOPL Statements

In this section we intend to examine the semantic equivalence of NL and FOPL statements with 'true' logical value. Following from the definition of logical equivalence, this examination requires an interpretation $\mathfrak{I} = < \Delta, I >$, 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. It is easy to see that without interpretation and variable assignment the equivalence of the next two statements is undecidable.

1/a Jeg elsker deg.
1/b P(x, y).

Assuming you do not speak Norwegian (i.e. without interpretation), you are not able to understand 1/a, and hence you are not able to find out its truth value. Also, without interpretation and variable assignment 1/b can be rendered into any two-place predicate. Now, if we give the interpretation of the predicate and assign values to the variables, we get

2/a I love you.
2/b Love(I, you).

However, we are still not able to determine the statements' truth value, and hence their equivalence. This is because NL sentences are produced and perceived in concrete communicative contexts. Thus, "it is not just what a sentence means, but the fact that someone utters it plays a role in determining what its utterance conveys" [9]. In this case we need to specify who is referred to as 'I' and who is referred to as 'you'. Owning this knowledge then allows us to decide whether the two statements are semantically equivalent or not.

Another 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" [9]. The difference is evident if we consider the following examples.

3/a There are students.
3/b $(\exists x)S(x)$.

4/a Some students are foreigners.
4/b $(\exists x)(S(x) \wedge F(x))$.
4/c $(\exists x)(F(S(x)))$.

3/b reveals the true structure of the existential proposition 3/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, 4/a does not express existential proposition and it does not

contain any sentential constituent corresponding to the conjunction in 4/b. On the other hand, 4/c 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 $\tau : L_1 \to L_2$ is said to be composition preserving if

$$\tau(o(f_1, f_2, ...)) \equiv \tau(o)(\tau(f_1), \tau(f_2), ...), \tag{3.1}$$

i.e. $\tau(o(f_1, f_2, ...))$ and $\tau(o)(\tau(f_1), \tau(f_2), ...)$ 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 (see example 5/c). In this case however, the semantic interpretation of the FOPL formula is not easier than that of the NL sentence.

3.3. Semantic Equivalence Assignment

The previous section defines what we mean by the semantic equivalence of two statements. If the two statements are in the same set of statements (A), then semantic equivalence is a binary relation over the given set, denoted by $R \subseteq A \times A$. If R is reflexive, symmetric and transitive, then is said to be an equivalence relation. In this section we consider two sets of statements: let NL denote the set of well-formed grammatical NL sentences and $FOPL$ denote the set of well-formed FOPL formulas. Examine the properties of semantic equivalence over both sets.

1. R is reflexive, if $\forall a \in A$ (aRa) holds.
2. R is symmetric, if $\forall a, b \in A$ $(aRb \Rightarrow bRa)$ holds. Thus, if a is the semantic equivalent of b, then the opposite is also true.
3. R is transitive, if $\forall a, b, c \in A$ $(aRb \wedge bRc \Rightarrow aRc)$ holds. Thus, if a is the semantic equivalent of b and b is the semantic equivalent of c, it entails that a is the semantic equivalent of c.

Taking the sets of NL sentences and FOPL formulas all three properties evidently hold, therefore semantic equivalence can be considered as equivalence relation over both sets. An equivalence relation divides a set into a number of non-empty, pairwise disjoint subsets (equivalence classes). The statement

sets constructed from these semantic equivalence classes are denoted by NL/R and $FOPL/R$, respectively. We define a semantic equivalence assignment n between these two sets as $n : NL/R \rightarrow FOPL/R$. Now we focus on studying the properties of this assignment in view of the criterion of composition preserving.

1. n is a mapping, if $\forall s \in NL/R$, $\exists f \in FOPL/R$ so that $(s, f) \in n$, and $\forall s \in NL/R$, $\forall f_1, f_2 \in FOPL/R$ $((s, f_1), (s, f_2) \in n \Rightarrow f_1 = f_2)$. Thus, every NL sentence should have a corresponding FOPL formula.
2. n is injective, if $\forall s_1, s_2 \in NL/R$, $\forall f \in FOPL/R$ $((s_1, f), (s_2, f) \in n \Rightarrow s_1 = s_2)$. Thus, every FOPL formula can have only one corresponding NL sentence (but it is not necessary to have any).
3. n is surjective, if $\forall f \in FOPL/R$, $\exists s \in NL/R$ so that $(s, f) \in n$. Thus, every FOPL formula should have one corresponding NL sentence.
4. n is bijective, if n is injective and surjective.

First, we need to prove that n is a mapping. Consider the following examples.

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

Here, the logical counterpart 5/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, 5/c is a well-formed FOPL formula, but it is not composition preserving.

6/a Most students like sports.
6/b (Most x: S(x)) Like(x, sports).

Although in this case the logical form respects the structural integrity of the quantified noun phrase, it is not a standard FOPL statement. Barwise and Cooper [5] 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).

7/a Students travel home regularly.
7/b $(\forall x : S(x))$ Happens(regularly, Travel(x, home)).

The logical form here is not a valid statement even in extended versions of FOPL. The reason for this is that in FOPL it is not allowed to quantify over predicates.

Although we have not considered all linguistic phenomena, we could find some that cannot be represented in standard FOPL at all, or not with the precondition that we would like to keep the structure of the NL sentence. Therefore we can state that $n : NL/R \rightarrow FOPL/R$ is not a mapping. If we restrict though the set of NL sentences to those that can be represented in FOPL, we can prove that n' is still not an unambiguous mapping. Consider the next example.

8/a Every student read a book *(over the vacation).*
8/b $(\forall x)(\exists y(S(x) \wedge B(y) \wedge R(x, y))$.
8/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 8/b, or as 'every student read the same book' as in 8/c. This phenomena is known as scope ambiguity, and results from the fact that NL, in opposition to FOPL, is structurally ambiguous [8]. As a consequence of the possibility of these kinds of multiple assignments n' is said to be a multivalued mapping. On the other hand, n" : $FOPL/R \rightarrow$ *restricted* NL/R would be a surjective mapping if we ignore the criterion of composition preserving. From this analysis we can conclude that the semantic content set $FOPL/R$ is able to cover is narrower than that of NL/R.

4. Higher-Order Logic

We can go beyond FOPL in two directions. On the one hand, we can introduce calculuses of higher order, in which propositions or propositional functions (and therefore sets) can appear as arguments to other functions. On the other hand, we can use higher (constructive and nonconstructive) methods like recursive numerical functions, symbolic structures, and semantic methods. In some ways intermediate between these are systems in which numbers are explicitly introduced (as primitives) into the domain of arguments [10]. From the above examples and examinations it is clear that we emphasize the introduction of higher-order calculus and numerical primitives.

4.1. Formal Definition of HOPL

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 arguments of predicates and values of variables (i.e. quantification over predicates is allowed). In other words, higher-order logics allow for quantification not only of elements of the domain of discourse, but subsets of the domain of discourse, sets of such subsets, and other objects of higher type (such as relations between relations, functions from relations to relations between relations, etc.). The semantics are defined so that, rather than having a separate domain for each higher-type quantifier to range over, the quantifiers instead range over all objects of the appropriate type. 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 [11].

According to [12], a common approach to describing the syntax of a higher-order logic is to introduce some kind of typing scheme. One approach types first-order individuals with ι, sets of individuals with $\langle\iota\rangle$, sets of pairs of individuals with $\langle\iota\iota\rangle$, sets of sets of individuals with $\langle\langle\iota\rangle\rangle$, etc. Such a typing scheme, however, does not provide types for function symbols. A more general approach to typing is that used in the Simple Theory of Types [13]. Here again, the type ι is used to denote the set of first-order individuals, and the type o is used to denote the sort of booleans: {true, false}. In addition to these two types, it is possible to construct functional types: if σ and τ are types, then $\sigma \to \tau$ is the type of functions from objects of type σ to objects of type τ. Thus, an expression of type $\iota \to \iota$ represents a function from individuals to individuals. Similarly, an expression of type $\iota \to o$ represents a function from individuals to the booleans. Using characteristic functions to represent predicates, this latter type is used as the type of predicates whose one argument is an individual. Similarly, an expression that is of type o is defined to be a formula. Typed expressions are built by

- application, i.e. if M is of type $\sigma \to \tau$ and N is of type σ, then their application MN is of type τ, and
- abstraction, i.e. if x is a variable of type σ and M is of type τ, then the abstraction λxM is of type $\sigma \to \tau$.

Propositional connectives can be added to these terms by introducing the constants $\wedge$, $\vee$ and $\supset$ of type $o \to o \to o$ and $\neg$ of type $o \to o$. Quantification arises by adding (for each type σ) the constants $\forall_\sigma$ and $\exists_\sigma$ both of type $(\sigma \to o) \to o$. The intended denotation of $\forall_\sigma$ is the set that contains one element, namely the set of all terms of type σ; while the intended meaning of $\exists_\sigma$ is the collection of all non-empty subsets of type σ.

Higher-order logic can be interpreted over a pair $\langle\{D_\sigma\}_\sigma, J\rangle$, where σ ranges over all types. The set D_σ is the collection of all semantic values of type σ and J maps constants to particular objects in their typed domain. There are two major ways to interpret higher-order logic. A *standard model* is one in which the set $D_{\sigma\to\tau}$ is the set of all functions from D_σ to D_τ. Such models are completely determined by supplying only D_ι and J. If D_ι is denumerably infinite, then $D_{\iota\to o}$ is uncountable, thus standard models can be very large. As a consequence of Gödel's incompleteness theorem, the set of true formulas in such a model are not recursively axiomizable; i.e. there is no theorem proving procedure that could (even theoretically) uncover all true formulas [14]. In the *general (or Henkin) model* [15], however, it is possible for $D_{\sigma\to\tau}$ to be a proper subset of the set of all functions from D_σ to D_τ as long as there are enough functions to properly interpret all expressions of the language of type $\sigma\to\tau$. Hence this model is sound and complete.

4.2. Reasons for HOPL

The necessity of HOPL in representing NL semantics is proved in view of the arguments against it. Firstly, reification [6] 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. Moreover, the resulting valid FOPL formulas will not be in accordance with the precondition of composition preserving. Secondly, [3] 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 $\forall y.P(y)\to PR(y,x)$. This solution formally results in valid FOPL formulas, but the criterion of composition preserving is violated, which leads to the following problems. First, consider the example under 5. In 5/c we lose the syntactic structure of the NL sentence. This FOPL formula ignores the subordination relation between the NL constituents: all elements are at the same level, and the original structure is obscured. This problem can be eliminated by the use of higher-order predicates. In general, a higher-order predicate of order n takes one or more $(n-1)^{th}$-order predicates as arguments, where $n>1$. Now, consider another example.

9/a I drink cold milk regularly.
9/b Happens(regularly, Drink(I, cold, milk)).
9/c Drink(I, regularly, cold, milk).

In 9/c not only the criterion of composition preserving is violated but one may draw the false conclusion that 'regularly' refers solely to the predicate 'drink'. In fact, it refers to the whole statement (see also example 7).

Example 6/b is another argument for the necessity of HOPL, since FOPL restricts the use of quantifiers to $\exists$ and $\forall$. A contemporary theory of quantification is the so-called Generalized Quantifier theory [4], [5]. This theory introduces many primitive quantifier expressions, as well as symbols for handling counting quantifiers.

For further information about second-order logic and its comparison with first-order logic we refer the reader to [16], [17] and [18]. For details about modal logic [19], [20], many-valued logic and fuzzy logic see [21].

Since variables are connected to domains in PL, HOPL expressions need to be embedded (see 5/b, 7/b). However, if we consider 9/b we can see that HOPL is not an adequate formalism either, because the structure under the predicate 'drink' is obscured: one may improperly conclude that 'cold' refers to 'drink' rather than 'milk'. Consequently the formalism needs further extensions.

5. Extension of HOPL

As we have declared previously, our project aims at developing an intelligent agent (see Figure 1) that is able to express its observations and states in NL sentences. In the first stage, we restrict the expressions to the observations which are related to definite, unambiguously interpretable situations. Consequently, the sentences describing these situations are factual assertions with true logical value. Therefore the following linguistic phenomena are beyond the scope of our investigations:

- if-then structures and conditionals,
- imperative, optative, exclamatory and interrogative sentences,
- probability and other certainty/uncertainty factors,
- intentional secondary meaning (pragmatics).

On the other hand, linguistic phenomena that need to be studied are as follows:

- domain types,
- referring expressions,
- adverbs,
- adjectives,

– numerical expressions and cardinality,
– quantification,
– logical connectives,
– historical (temporal) sequences, and
– causality.

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 [22] in linguistics) are associated with and determined by the predicate.

1.1/a Peter loves Mary.
1.1/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.

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

2.2/a Peter reads a/the book Tom likes.
2.2/b Read(Subject: Peter, Object: isa(book):x |
Like(Subject: Tom, Object: x)).

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.

2.3/a Peter likes books.
2.3/b Like(Subject: Peter, Object: book).

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. The difference is clearly seen in the second example.

3.1/a Peter travels by train.

3.1/b Travel(Subject: Peter, Instrument: isa(train)).

3.2/a Peter often travels by train.

3.2/b Happens(Subject: Travel(Subject: Peter, Instrument: isa(train)), Time: Often).

3.2/c *Travel(Subject: Peter, Instrument: isa(train), Time: Often).*

The logical form in 3.2/c is incorrect, because from its truth does not follow that *Travel(Subject: Peter, Time: Often)* is true.

The following example is an illustration for the ambivalent nature of NL, where we cannot decide which predicate the adverb is linked to.

3.3/a I see you running today.

3.3/b See(Subject: I, Object: Run(Subject: You, Time: Today)).

3.3/c See(Subject: I, Object: Run(Subject: You), Time: Today).

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

4.1/a Peter reads a scientific book.

4.1/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 $some(a)$ function for creating a group of objects.

5.1/a Peter reads two books.

5.1/b Read(Subject: Peter, Object: some(isa(book)):x | Property(Subject: x, Object: Two)).

5.2/a Peter reads more books than magazines.

5.2/b Read(Subject: Peter, Object: (some(isa(book)):x, some(isa(magazine)):y) | More(Subject: x, Object: y)).

5.3/a Peter reads more books than Tom.

5.3/b (Read(Subject: Peter, Object: some(isa(book)):x), Read(Subject: Tom, Object: some(isa(book)):y) | More(Subject: x, Object: y)).

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

6.1/a There is a book on a/the table.

6.1/b Is(Subject: isa(book), Location: isa(table)).

6.2/a There are some books on a/the table.
6.2/b Is(Subject: some(isa(book)), Location: isa(table)).

6.3/a All books are on a/the table.
6.3/b Is(Subject: all(isa(book)), Location: isa(table)).

7) Logical operators can be applied to predicates or to arguments of predicates. When they refer to predicates we should note, that *and* means the presence of multiple predicates (they can be connected with the , operator), while *or* means the uncertainty of the observation (which is beyond the scope of our investigations).

7.1/a Peter reads and laughs.
7.1/b (Read(Subject: Peter), Laugh(Subject: Peter)).

7.2/a Peter reads not laughs. ≡ Peter reads.
7.2/b Peter does not read.
7.2/c NotRead(Subject: Peter).

In the latter example, case (a) demonstrates that we restrict our examinations to observations and the addition of extra information is not allowed. Case (b) states that Peter is not doing something without stating what he is doing. As a result, case (c) shows that an observation is uninterpretable without a specific predicate, thus *not* can only be allowed if included in the predicate.

The same applies when logical operators are related to arguments of predicates. Here *and* is represented by the grouping of the corresponding arguments, and *or* means uncertainty which is not covered by our investigations. Also, negation either expresses that an argument is not something without saying what it is, which is uninterpretable in our framework; or it states what the argument is, in which case the negation is an extra piece of information (e.g.: Peter reads not a book but a magazine. ≡ Peter reads a magazine.).

8) Temporal aspects can only be studied when several observations are compared on a historical basis. In this case the former observation(s) must have a tense preceding the latter observation(s). The observation at the end of the history demonstrates the actual (present) state of the system.

8.1/a Peter gives Tom a book. Tom reads the book.
8.1/b Give(Subject: Peter, Object: isa(book):x, Recipient: Tom) →
Read(Subject: Tom, Object: x).
8.1/c Tom reads the book that Peter gave him.

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

9.1/a Peter cannot sleep because Tom is dancing.

9.1/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. Therefore, considering the assignment $m' : EHOPL/R \rightarrow NL/R$ (where $EHOPL/R$ denotes the set of extended HOPL statements constructed from semantic equivalence classes) we can state, that m' is a surjective mapping.

6. ECG: Graphical Representation of EHOPL

The present investigations concerning our project (see Figure 1) focus on modeling the agent's ability to assign semantic representations to its observations. For this purpose we have developed the Extended Conceptual Graph (ECG) model [23], a graphical conceptual modeling language that can be used to describe the semantics of an agent's internal knowledge model. In our model, the process of conceptualization occurs at two levels. At the primary level the direct and static mapping of the objects and relations within an observation takes place. At the extended or abstract level temporal and other complex relationship types are also managed. The main building blocks of the model are concepts, relationships, and containers which serve for structuring the model. The graphical representation of model elements is shown in Figure 2. The 'world' is built up of interconnected ECG model fragments representing separate observations, containing exactly one kernel predicate (denoted by *) and having 'true' truth value. The model is characterized by

- a predicate-centered schema language,
- the fine distinction between the different categories of concept and relationship types,
- the fixed set of elements with flexible semantic assignment, and
- the generality and reusability of basic model structures.

In the present article we examine the expressiveness of primary level ECG. Consequently, we will go through the linguistic phenomena identified in the previous section (except for temporal aspects, because they can only be studied at the extended level) and show that all of them can be represented in our model. In Equation 3.1 we defined a composition preserving transformation, and stated that two statements from different notation systems are said to

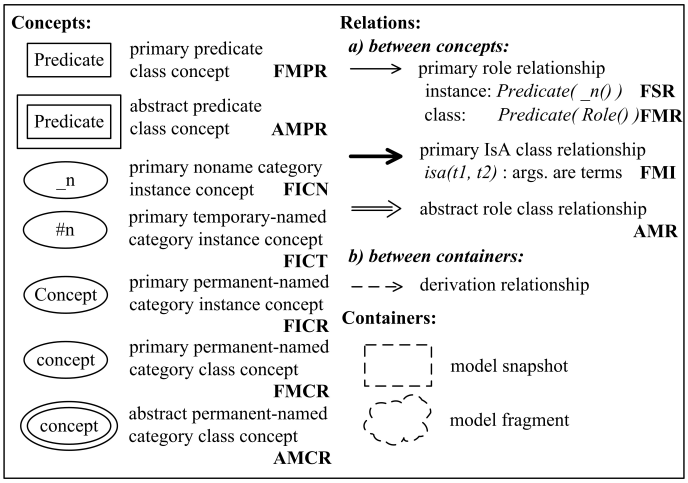

Figure 2. Components of the ECG model

be equivalent if there exists a composition preserving transformation between them by means of which the two formulas are equivalent in all interpretations. The following analysis specifies the composition preserving transformation of EHOPL formulas into graphical ECG structures.

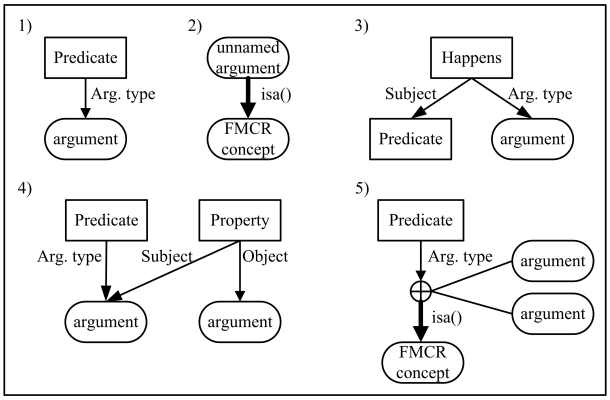

Figure 3. Basic ECG structures

Figure 3 shows the identified basic ECG structures. 1) illustrates a predicate with a typed argument, where types correspond to semantic roles. Arguments can be arbitrary ECG concepts, including predicate concepts as well. Objects are represented by different types of category concepts (see Figure 2).

Accordingly, we make a distinction between concepts referring to concrete objects (FICR), concepts referring to a collection of objects (FMCR), concepts referring to unreferenced unnamed objects (FICN), and concepts referring to referenced unnamed objects (FICT). The two latter serve for making a distinction between the use of indefinite and definite articles, respectively. 2) shows how an unnamed object is associated with a collection of named objects through the $isa()$ relationship. Adverbs connected to the predicate are considered extra arguments of the predicate. On the other hand, 3) demonstrates how adverbs associated not only with the predicate itself but with the whole assertion are handled. Similarly, 4) displays the treatment of adjectives as arguments of the *Property* predicate. 5) shows how groups of objects can be composed. If an adjective indicating the cardinality of the group is also present then a *Property* predicate with an argument needs to be added to the construction. The handling of quantifiers and logical connectives originates in the previously discussed basic structures with the extension that also predicates can comprise a group. Causality can be traced back to 3) where the *Happens* predicate has a *Cause-type* and a *Result-type* predicate argument.

From these basic structures an ECG model, which is actually a semantic network, with arbitrary complexity can be built. For illustration, see Figure 4, which shows the ECG fragment for the observation "A black circle is in the white triangle".

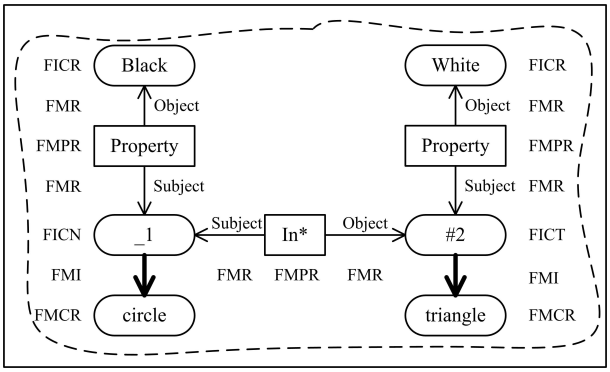

Figure 4. ECG fragment for an observation

This analysis proves that the assignment $e : EHOPL/R \rightarrow ECG$ (excluding temporal aspects) is a bijective function, therefore $e^{-1} : ECG \rightarrow EHOPL/R$ also exists. Thus we can state that the two formalisms are semantically equivalent, that is the same semantic content can be represented by both symbolisms. Therefore the ECG model can be viewed as the graphical counterpart of

the EHOPL language. As a consequence, the assignment $f' : ECG \to NL/R$ is a surjective mapping (just like m'). Note here that the set of ECG statements needs not be restricted to a set of semantic equivalence classes, because the ECG model is a semantic model constructed for representing the semantic content of a given situation.

7. Conclusion

The aim of the present article was the examination of primary level ECG's expressive power. In other words, we have been looking for an answer to the question: to what extent primary level ECG is able to model natural language semantics. Since the analysis was performed on a logical basis, this examination covered not less than the semantic comparison of natural language statements and logical formulas. We were interested in logically significant natural language expressions, and we have considered to what extent their semantics is captured by the logical behavior of their formal counterparts.

We can now draw the conclusion that the ECG model is able to grasp the semantic content of situations, and from the article we can see that every ECG statement can be rendered into an NL sentence. This assignment is unambiguous, that is every ECG statement can have only one corresponding NL formulation (with the assumption that semantically identical NL sentences are considered to be one). On the other hand, we can also state that every NL sentence can be approximated by an ECG model, if the pragmatic level of language is not taken into account. The ECG model is a recursive, compositional system: that is infinitely many statements can be constructed from the small finite set of model elements. Consequently, the more extended an ECG model is, the better it is able to approximate NL.

REFERENCES

[1] Ben-Yami, H.: *Logic & Natural Language*. Ashgate, 2004, ISBN 0-7546-3743-3.

[2] Blackburn, P. and Bos, J.: Computational semantics. *Theoria*, **18**, (2003), 27–45.

[3] Peregrin, J.: What does one need when she needs 'higher-order' logic? In *Filosofia*, LOGICA'96, Praha, 1997.

[4] Higginbotham, J. and May, R.: Questions, quantifiers and crossing. *Linguistic Review*, **1**, (1981), 41–79.

[5] Barwise, J. and Cooper, R.: Generalized quantifiers and natural language. *Linguistics and Philosophy*, **4**, (1981), 159–219.

[6] Jurafsky, D. and Martin, J. H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2nd edn., 2008.

[7] Szeredi, P.: Az ontológiakezelés matematikai alapjai. Ontosz Klub, 2008.

[8] Keenan, E.: How much logic is built into natural language? In *Fifteenth Amsterdam Colloquium*, ILLC, University of Amsterdam, 2005, pp. 39–45.

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

[10] Curry, H. B.: *Foundations of Mathematical Logic*. Dover Publications, Inc., New York, 1977, ISBN 0-486-63462-0.

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

[12] Miller, D.: *Encyclopedia of Artificial Intelligence*, chap. Logic, Higher-order. 1991.

[13] Church, A.: A formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, **5**, (1940), 56–68.

[14] Andrews, P.: *An Introduction to Mathematical Logic and Type Theory*. Academic Press, 1986.

[15] Henkin, L.: Completeness in the Theory of Types. *Journal of Symbolic Logic*, **15**, (1950), 81–91.

[16] Hinman, P. G.: *Fundamentals of Mathematical Logic*. A K Peters, 2005, ISBN 1-56881-262-0.

[17] Väänänen, J.: Second-order logic and foundations of mathematics. *The Bulletin of Symbolic Logic*, **7**(4), (2001), 504–520.

[18] Rossberg, M.: First-order logic, second-order logic, and completeness. In *First-Order Logic Revisited*, Logos Verlag Berlin, 2004, pp. 303–321.

[19] Gamut, L. T. F.: *Logic, Language, and Meaning. Volume 2. Intensional Logic and Logical Grammar*. The University of Chicago Press, 1991.

[20] van Benthem, J.: *The Logic of Time*. Kluwer Academic Publishers, 2nd edn., 1991.

[21] Jacquette, D. (ed.): *A Companion to Philosophical Logic*. Blackwell Publishers, 2002, ISBN 0-631-21671-5.

[22] Fillmore, C.: *Universals in Linguistic Theory*, chap. The Case for Case. New York: Holt, Rinehart and Winston, 1968.

[23] Baksa-Varga, E. and Kovács, L.: Knowledge base representation in a grammar induction system with Extended Conceptual Graph. *Scientific Bulletin of 'Politehnica' University of Timisoara, Romania*, **53**(67), (2008), 107–114.