Linguistic Processor Based on Object-Attribute Grammar

Sergey Salibekyan and Petr Panfilov

National Research University Higher School of Economics, Myasnitskaya Ulitsa 20, Moscow 101000, Russia,
salibek@yandex.ru, ppanfilov@hse.ru

Abstract. In this paper, we present an object-attribute grammar (OAG) – an original formalism for describing the natural language semantic analysis algorithm for the linguistic processor (LP). Special focus is made on formatting input and output data for LP. The LP uses the graph containing word interpretations of source text and transforms it to the graph representing meaning of the text. We present definition of graph transformation algorithm of LP by means of the developed notation, denoted a template of a subgraph which is searched in a graph and an operation of subgraph transformation. The software implementation of the LP based on OAG is described.

Keywords: natural language processing, linguistic processor, semantic network, graph-transformation system, formal grammar, text-meaning representation

Introduction

The paper is devoted to the field of mathematical linguistics and a formalism describing the algorithm of the linguistic processor (LP) operation is considered. The LP goal is to transform a text in a natural language (NL) into an information structure (as a rule, a semantic network) representing the text meaning.

Much effort has been directed to solve this problem, and as a result, several mathematical models were constructed. The most famous of them are the following ones: Generative and transformational (TG) or transformational-generative (TGG) grammar developed by Noam Chomsky and Head-driven phrase structure grammar (HPSG) developed by Carl Pollard and Ivan Sag. But each of these formalisms has significant drawbacks which are manifested when a NL is analyzed. These drawbacks will be discussed below. Therefore, we propose a new mathematical model of LP operation, which is based on the object-attribute (OA) principle of data structure organization. This formalism (OA-grammar) was constructed in the process of developing a system of NL semantic analysis and is an integral part of this analysis.

Although the goal in this paper is to describe the developed LP formalism, we cannot neglect the format of the initial data for LP (i.e., the form of the text representation with is processed by LP) and the formal of the data obtained after their processing by LP. This problem is considered in the first section. The input and output data are graph structures, and LP itself is a graph-transformation system which transforms the initial graph into a semantic network representing the text meaning. The OA-LP (i.e., LP based on the OA-principle) transforms graphs according to elementary transformation rules (productions).The second section directly deals with the OA-grammar. In the third section, we consider the example of OA-grammar applications in analysis of the English language. This allows the reader to understand the principle of the proposed formalism more precisely.

1 Object-attribute principle of data structure organization for LP

The data structure and format used in text representation for the natural language processing play important role in design of a language processor (LP). The initial text representation must be convenient for its analysis by the LP, and the final data structure must adequately reflect the text meaning and must be convenient for the further analysis (e.g., in the information search or data mining applications). We propose our own format of initial and final data which is based on the OA-approach to the data structure organization. We can say that all formats listed above are different methods for representing the frame network [1]. The frame is a set of named slots storing the properties of some entities (as a rule, one frame is used to describe the properties of an entity). A slot contains either a constant or a reference to another frame. The frames united by references form a frame network. In linguistics, the frame network is used to represent the text meaning and to describe words in the text under study. In the first case, the frames are associated with entities described in the text, and the references to other frames are associated with semantic relations between the entities. In the second case, the frames store the description of the word semantic and morphology and are used in the process of semantic-syntactic analysis of NL. The notion of frame was introduced by M. Minskii at the end of the 1960s [2]. Formally, a frame is represented as (1):

$$F = \big(FN, (SN1, SV1), \ldots, (SNn, SVn)\big), \tag{1}$$

where FN is the frame name, SN is the slot name, SV is the slot value $SV \in \{SN \cup \text{Const}\}$, where Const denotes a constant (number, symbol string, etc.), i.e., the slot value can be the name of a frame, which is equivalent to a reference to a frame (the references allow one to unite frames into a network).

The concept is used in the frame linguistics introduced by Ch. Fillmore [3]. In the case of such a data structure organization, the relations between frames can have any arbitrary topology, which theoretically allows one to describe any entities and systems. But there is no acceptable formalism for the frame network which can be used to describe the synthesis of the information structure reflecting the text meaning. For example, the generative grammar introduced by

N. Chomsky can synthesize information structures only if they have a “tree”-type topology. The frames can also be used to describe the properties of parts of speech so that these descriptions allow one to analyze the text. For example, the parts of speech correspond to the frames whose slots may be semantic relations to other words. Such slots are called *semantic valences*, because they, by analogy with chemical valences forming molecules from atoms, “attach” the values of other words (actants) from the text and thus form a semantic network. The actants are rather often attached not to all valences of the word, and some of them turn out to be inactive.

The attribute value matrices (AVM) [4] format describes a frame by a matrix (feature structure) consisting of two columns (the number of rows is unbounded). Each row (feature structure) is used to represent one property (feature) of the entity (one slot of a frame) and contains two elements: [attribute and value]. The component value is atomic (a string) or another feature structure. Thus, the feature structure is an embedded structure which allows one to construct a frame-like network of a tree topology. Such a restriction on the network topology is a significant drawback of AVM, because, in the real world, one can rather often observe semantic feedbacks between objects and phenomena. An advantage of AVM is that a formalism, which describes the synthesis of the frame structure reflecting the text meaning (HPSG), has been developed for AVM.

There are methodologies for NLP based on graph grammars. According to this methodologies, text is converted into the form of a graph (semantic network). Then, the graph is converted into a final graph describing the meaning of the text through a sequence of transformations. Each transformation named production consists of two parts: left and right. The left part describes the subgraph that must be searched in the graph. The right part specifies the conversion rule of the found subgraph. As such a system can result in [5]. This article provides the syntax for describing the initial graph and syntaxes of graph transformation rules. This approach is very similar to the approach presented in the article.

The object-attribute (OA) approach to the data organization was developed for the purpose of obtaining more convenient descriptions of complex structured dynamically transformable data. In fact, the approach is similar to the frame network. The main notion in the OA-approach is an information pair (IP). Similarly to the slot, IP consists of the following two fields: a load containing both a constant or a reference and an attribute characterizing the load (an analog of the frame slot name). But in contrast to the frame, the IP attribute is not a text string but a number (identifier). The second notion is an information capsule (IC) (2). Similarly to the frame, IC is a set of IP. Each IC has its own unique identifier (or address/reference). The IC identifier can be located in the IP load to organize references between IC (similarly to the frame slot with a reference to another frame). A set of IC united by references is called a OA-*graph* or a OA-network (the OA-network can have any arbitrary topology).

The IC is formally determined as (2):

$$IC = \big((A1, L1), \ldots, (An, Ln)\big), \tag{2}$$

where A is an IP attribute ($A \subset N$, where N is the set of positive integers) and L is the IP load ($L \in \{\text{Const}\}$, where Ω is the set of IC identifiers ($\Omega \subseteq N$), Const denotes a constant (a number or a symbol string)).

The IP can be divided into two classes. The first class contains fields. The load of such an IP contains a constant, and the attribute of such an IP identifies a constant ($L \in \text{Const}$). The load of IP in the second class (relation) contains a pointer (index/identifier) to another IC ($L \in \Omega$). Such a reference reflects the semantic relation between objects, and the attribute of such an IP identifies the type of the relation.

The OA-graph topology can be arbitrary, which is an advantage of this data representation format. An advantage over the frame network is that the OA-graph has a more flexible structure, i.e., some IP can be copied and transferred into another IC during analysis of the text. It should be noted that we developed a formalism which permits describing the process of synthesis of a semantic network, which reflects the text meaning, starting from the initial text. The formalism has a rather significant capacity. This formalism is described and compared with the already existing formalisms in the next section.

Another important problem of this study is to distinguish the basic types of semantic relations between objects. So Ch. Fillmore distinguished nine types of relations (roles) in the theory of syntactic roles [6]: agent, contractor, object, place, addressee, patient, result, tool, and source. But we succeeded in distinguishing a much greater number of such relations. To organize a OA-graph in the process of studying, we distinguished approximately a hundred of attributes (types) of IP-fields and IP semantic relations. These relations can be divided into groups.

The group for describing a set is used to group objects, properties, and states of object. All sets can be divided into two classes. The first class comprises the sets, where all elements are indicated. The second class unites the sets, where a typical representative of the set and the number of elements are indicated (for example, a "crowd of 100 people"). We note that the representatives of the theory of semantic roles (Ch. Fillmore, R. Schank, P. Winston) did not distinguish such group of semantic relations. There are groups for describing an object, object propertias and states, spatio-temporal relations between physical objects and cause-effect relations between phenomena. We introduse the term **Confine** whitch is a spatiotemporal delimiter which restricts the existence of certain properties of an object in space and time. **Confine** can be of several types: topology (indicates the relative position of objects with respect to each other), direction (indicates the direction of motion of an object or its orientation in space), dynamics (prescribes a sequence of points in space and time on the trajectory of motion of an object), shape (determines the shape and dimensions of an object), and cause-effect (denotes two events one of which is a cause of other).

The meaning extracted from a text is represented as an OA-graph whose vertices can be divided into three levels, namely, description of objects, description of their properties and states, and description of spatiotemporal relations between the objects and cause-effect relations between events. As an example, we consider the description of spatiotemporal and cause-effect relations. Namely, as an example, we consider the OA-graph containing the meaning of the sentence “A boy pushed the door, and it opened” (Fig. 1). The OA-graph contains descriptions of the following two objects: “Boy” and “Door”. The boy, who is a subject (Subj), was in the state of opening the door (Push), the object (Obj) of his action is the door. The door was in the following two states: before the opening and after the opening. We do not know the type of the state before the opening, and therefore, the IP load with attribute Stage contains nil (an unknown constant or a pointer). The second state of the door is that the door is open. More precisely, for an object of action, the reference indicates the object properties but not its description. This is necessary because it may happen that an object experiences an action in a certain state and does not experience this action in another state. So, in our case, the door is under the action at the moment when it is in the first state. As a result of this action, the door is transferred into another state (the state of being open). This fact is given by Confine of cause-effect type, i.e., the first element of the ordered set is the cause (the state of pushing the door by the boy) and the second element is the state of the door, which is a consequence of this action (i.e., the state of being open).

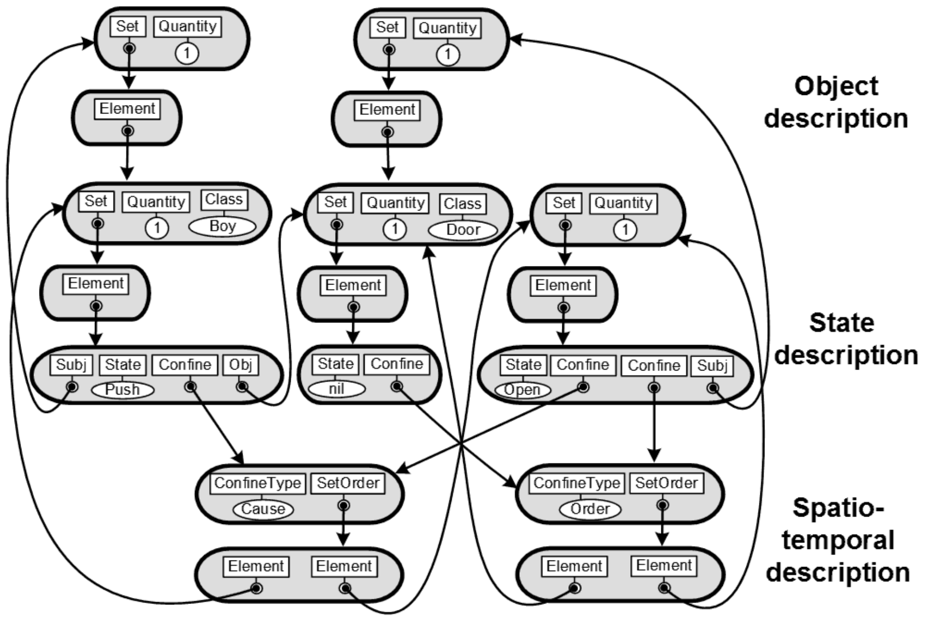

Fig. 1. OA-graph describing the sentence “A boy pushed the door, and it opened”

The OA-graph is used not only to represent the meaning of the text but also to represent the initial data for LP (Fig. 2). So the initial data are represented as an OA-graph of a special format, i.e., as a list of interpretations of words in the initial text (OA-grammar transforms this list into a network reflecting the text meaning). The list of word interpretations can be divided into the following three levels: the root list, the list of interpretation branches, and the sequence of word interpretations. At the beginning, each element of the root list is associated with one word. But rather often, the word has a set of interpretations, and therefore it is necessary to associate each word with the list of its interpretations (the second level of the list of word interpretations). But in the synthesis of the semantic structure representing the meaning, it may happen that the interpretations are already different for rather large fragments of the text (i.e., sequences of words). This sequence of word interpretations forms the third level of the list of word interpretations. Each word interpretation is given by at least two IC. The first IC contains the description of morphological properties of the word and the properties required for the semantic agreement with the nearest words (Fig. 2). This IC contains IP with the attribute SemProp (semantic properties) whose load contains the pointer to IC contained the description of the semantic properties of the word. The first IC is used to analyze the text and is deleted after analyzing. As the analysis of the text is finished, then the second IC is included to the semantic network (OA-graph) and is stored there to describe the semantic properties of the object, system, or the phenomenon denoted by the word. The descriptions of the morphological and semantic properties of words are stored in the dictionary of the NL analyzing system, and, if necessary, they are transferred into the list of interpretations of words of the initial text. But in the present paper, we do not pay attention to the dictionary organization, because of this is no importance for describing the OA-grammar formalism. Further, the OA-grammar is implemented by transforming the OA-graph storing the list of word interpretations into another graph describing the text meaning.

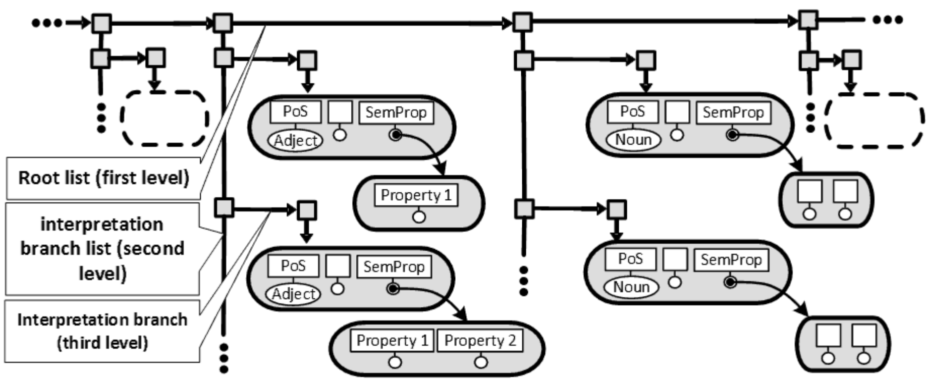

Fig. 2. List of interpretations of words in the initial text

2 Object-attribute grammar

The field of natural language processing (NLP) heavily relies on grammar(s) used in language analysis process. There are a number of research projects addressing the challenge of processing the syntax, morphology, phonology and semantics of a language using diverse formal representations of the natural language analysis process. One of the first formalism to describe the natural language analysis process was introduced by Noam Chomsky in mid-1950s. He consecutively introduced the Generative Grammar [7], the Transformational Grammar, and the Transformational-Generative Grammar (TG, TGG) [8] as a formal frameworks for language analysis. His early works have laid the foundation for further R&Ds in the NLP field, including such developments as the Head-Driven Phrase Structure Grammar (HPSG) [9] developed in mid-1980s as an alternative to the TG. The HPSG operates with the data represented as AVM that ensures a higher level of abstraction and permits detailed description of words and language structures with different characteristics and embedded substructures. Since the HPSG was aimed more at the synthesis of language structures, the problem of a natural language semantic analysis was not addressed properly with this grammar.

For the NLP domain, several grammars based on graph transformation formalism were proposed, such as the Node Label Controlled Grammar (NLC) [10] and the hypergraph grammars [11] including the Hyperedge Replacement Grammar (HRG), the Synchronous Hyperedge Replacement Grammar (SHFG), and the Adaptive Synchronous Hyperedge Replacement Grammar (ASHFG). The major disadvantage of graph grammars was considered a low abstraction level due to the fact that only one mark is associated with the node whereas one language entity may have several attributes.

To address challenges of the NLP and overcome drawbacks of existing solutions we propose a new formalism of an object-attribute grammar (the OA-Grammar) that allows for describing not only the algorithm for synthesizing phrases of natural language but the process of language analysis, as well. In this paper we present in details the analysis of a single sentence using the proposed object-attribute language processor (OA-LP), while leaving the description of technology for merging meanings of several sentences beyond the scope of current work.

The OA-grammar transforms the initial data (list of word interpretations description) into a semantic network (semantic OA-graph) by a set of rules. Each rule consists of two parts (the left and right sides). The left side presents a pattern, which is sought in the word interpretations list (as a rule, the pattern indicates subgraphs containing descriptions of the morphological properties of one, two, or three neighboring words). If the subgraph (or several subgraphs) is found, then the obtained format of the list of word interpretations is transformed according to the notation given in the right side of the transformation. If several subgraphs is founded then rule with least index is implemented (every rule has index). The transformation rules most often “glue” the word interpretations together, when the ICs with the semantic properties of the dependent word are attached to the description of the principal word in the phrase, and the morphological properties of the dependent word are then deleted from the list of word interpretations. The text is recognized in several stages from dependent to principal parts of speech, and the recognition process is complete, when the list contains only the principal word of the sentence (in the English language, this is the verb (i.e., the predicate)). Then the description of the morphological properties of the last word is deleted as useless, and a fragment of the semantic OA-graph describing the meaning of the sentence under recognition is obtained. Figure 4 illustrates an example of the operation of gluing a noun and an adjective. The following notation is used in Fig. 4: PoS – Part of Speech, SemProp – Semantic Properties, Atr – attribute. After the gluing operation is complete, the IC with the semantic properties is removed from the description of the dependent word in the phrase (adjective) and is placed as an attribute (Atr) into the the semantic properties description of the principal word (noun). The IC storing the morphological properties of the dependent word is then deleted as useless.

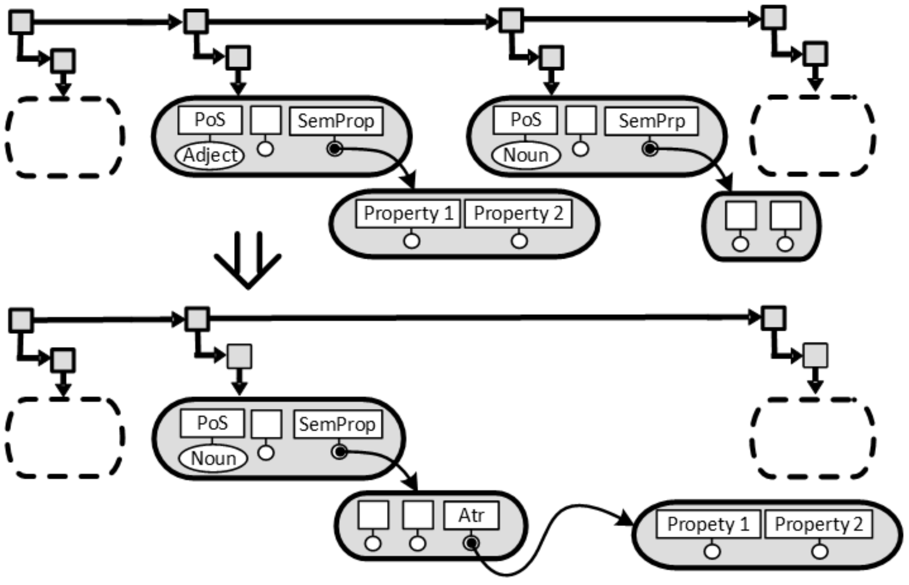

Fig. 3. Gluing the noun and adjective interpretations

The OA-grammar can operate with words and language structures in the case of their multiple meaning. This is ensured by the forking operation applied to the list of word interpretations. Assume that two word interpretations are glued together, and each of the processed words has two interpretations. In this case, the forking operation is applied to the fragment of the word interpretations list containing the description of interpretations of these two words, and as a result, one list of four interpretations of the text fragment is obtained (Fig. 4).

Each interpretation branch contains two word interpretations. Further, the two word interpretations which coincide with the pattern of the transformation rule on an appropriate interpretation branch are glued together. The interpretation branches can not only be generated by the forking operation but can also be deleted; for this, there is a special symbol in the notation of transformation rules (a branch is deleted if the word interpretation contained in it is inconsistent with the neighboring words). Thus, the process of transformation of the list of word interpretations into a semantic network is the process of generation and deletion of interpretation branches. Ideally, only one interpretation branch must remain after the all transformations, but if, as a result, there are several interpretation branches, then this means that the analyzed sentence is multi-valued. These interpretations are put into the semantic OA-graph as a set of alternative interpretations.

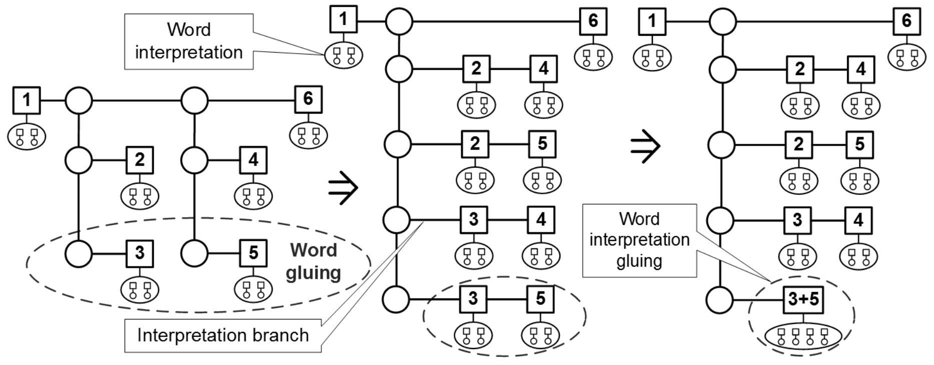

Fig. 4. Forking operation

3 The example of OA-grammar application

Now we study the description of the formalism describing the transformation of the list of interpretations of the words in the initial text. Formally, the OA-grammar is the quadruple (4):

$$\{A, L, G, P\}, \tag{3}$$

where A is the set of IP attributes, L is the set of IP loads ($L \in \{\Omega \cup \text{Const}\}$, where Ω is the set of IP indices ($\Omega \subseteq N$), and Const is a constant – symbol string, digit etc.); G is an OA-graph which is transformed by the OA-grammar, ($G \in T(A, L, \Omega)$, where $T(A, L, \Omega)$ is the set of all OA-graphs composed from the symbols $\text{IP}(a, l)$, where $a \in A$, $l \in L$; P is the set of rules of the OA-graph transformation in the form $Gt(A, L, \Omega 1_i) \to Gr(A, L, \Omega 2_i)$, where Gt is the algebra of the graph pattern descriptions, Gr is the algebra of descriptions of the graph fragment which replaces the obtained fragment of the list of word interpretations in G, and $\Omega 1_i$, $\Omega 2_i$ is the set of IC indices for the right and left sides of the rule (i is the index of the replacement rule).

The OA-grammar operates as follows. At the beginning, there is an initial OA-graph G. Then the rules of transformation of the OA-graph G are applied. One can apply a rule (such a rule is said to be enabled; otherwise, it is said to be disabled) only if G contains a subgraph (or several subgraphs) identical to that described in the left side of the rule.

The following notation has been developed for the transformation rules:

$\rightarrow$ is the delimiter between the right and left sides of the transformation rule (the left side contains the notation is for the OA-graph pattern describing, and the right side contains the notation for the OA-graph fragment used to replace (transform) the localized fragment of the OA-graph (word interpretation list)).

$a = l$ is the IP notation ($a \in A$, $l \in L$, where A is the set of attributes and L is the set of IP loads);

Name$\{\ldots\}$ is the IC notation; the IP contained in IC are written between the symbols "{","}"); Name is the IC name (or the mnemonics of the IC index); for convenience, the references (indices) are denoted by the symbol string (name).

$\{\ldots a1 = \mathrm{ICName}\{\mathrm{Ip1}, \mathrm{Ip2}\ldots\}\ldots\}$ is the IC containing the IP with the pointer to another IC in the load; the IC name (ICName) need not be pointed out;

$*$ denotes union of IC (the notation IC1$*$IC2 means that IC1 must be combined with IC2, i.e., the new IC must contain all IP from IC1 and IC2);

All rules of the OA-grammar are numbered and if, in the process of transformation of an OA-graph, it turns out that several rules are simultaneously enabled, then the rule with a lesser number is executed. If the strict succession of the rule execution is violated, then the semantic analysis of the text may be incorrect.

To illustrate the principles of OA-grammar operation, we consider the example.

Example 1. In the OA-grammar rules, the operation of gluing an adjective and a noun together (Fig. 4) is written as follows:

$$\mathrm{ADJECT}\{\mathrm{SemProp} = \mathrm{tmp}\}\mathrm{NOUN} \rightarrow \mathrm{NOUN}^*\{\mathrm{SemProp} =^* \{\mathrm{Atr} = \mathrm{tmp}\}\}, \tag{4}$$

where

ADJECT, NOUN are references to IC containing the description of the properties of the adjective and the noun;

SemProp is the attribute of the reference to IC containing the description of the semantic properties of the word;

tmp is a temporary variable containing the index of IC which is stored in the load of IP with the attribute SemProp;

Atr is the attribute of the reference to the description of the object properties.

Let us explain the notation of transformation rule (5). The rule is enabled if, in the list of word interpretations, the interpretations of the adjective and the noun follow each other (ADJECT, NOUN denote the pointers to IC containing the respective IP LangConstr=ADJECT and LangConstr=NOUN (LangConstr is the attribute of the language construction)). The pointer tmp in the left side of the rule denotes the index of IC stored in IP with attribute SemProp in the right side of the rule. In the right side, the following modification of the OA-graph is prescribed: the IP with attribute Atr, whose loads contain the pointer to IC with the description of the adjective semantic properties, are glued to IC containing the description of the semantic properties of the noun. Now the semantic properties of the adjective become the description of the properties of the object determined by the noun. And IC with the description of the morphological properties of the adjective is deleted, because it is not used in the further analysis.

4 Conclusion

It is convenient to analyze NL by the OA-grammar due to several reasons. First, the OA-grammar, can process languages which are context-sensitive according to the Chomsky hierarchy [12] (NL belong to this class), because the OA-grammar rules analyze language structures rather than separate words. Second, the OA-grammar can deal with multi-valued words and multi-valued language structures. Third, the OA-grammar can generate semantic net of any topology for describing the text meaning. Fourth, the OA-grammars can be used both to synthesize a semantic network reflecting the text meaning and to synthesize the text from a semantic network. The contemporary grammars can only be used to synthesize the language structures, while the inverse problem (i.e., to use the grammar to construct systems for analyzing the languages) is already a rather difficult task. In the text synthesis from an already available semantic network, the left side of the OA-grammar rule indicates the pattern of the semantic network subgraph, while the generated language structures are indicated in the right side, and all this permits constructing an automatic translator. Such a translator will operate as follows: first, LP synthesizes an OA-graph reflecting the text meaning from one language, and then PL synthesizes the text in the other language from the semantic OA-graph. The problem of text synthesis is extremely complicated, because it is necessary to make the text concise and understandable and to preserve the style of the text. This field of research has not yet been investigated because of its complexity, but it deserves attention. Fifth, the OA-grammar ensures a flexible and simple notation, i.e., the notation is sufficiently limited and the form of the rule representation is rather compact. The separation of the OA-grammar rules into stages makes the description of the text analysis more understandable for the LP designer.

The program realization of OA-LP is currently being developed. The program is based on the program module (linguistic processor) which transforms the list of word interpretations into a semantic OA-graph. Starting from the OA-grammar rules, the programmer constructs a program of the module operation. The program takes into account all specific features of the language recognition, which cannot be performed by using the OA-grammar rules or which would be irrational, namely, the meaning coupling of sentences, the automatic coordination of language structures (for example, in the Russian language, it is automatically agreement of the parts of speech according to number, gender, and case), the adverb coupling with the denotatum, etc. Nowadays, we realized an experimental base of knowledge for the semantic analysis of the English and Russian languages. The knowledge base contains a restricted set of word descriptions and a program for synthesis of the semantic OA-graph, which realizes a rather restricted set of OA-grammar transformation rules.

References

1. Unknown: Minsky's frame system theory. In: Proceedings of the 1975 workshop on Theoretical issues in natural language processing – TINLAP '75. pp. 104–116 (1975)
2. Minsky M.: A framework for representing knowledge. MIT AI Laboratory Memo 306 (1974)
3. Fillmore C.J.: Frame semantics and the nature of language. In: Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech. Volume 280: 20–32 (1976) http://onlinelibrary.wiley.com/doi/10.1111/j.1749-6632.1976.tb25467.x/abstract
4. Sag, Ivan, Thomas Wasow, Emily Bender: Syntactic Theory: A Formal Introduction (2nd ed.). Stanford: CSLI Publications (2003)
5. Bernd Bohnet, Leo Wanner: Open Source Graph Transducer Interpreter and Grammar Development Environment In: Proc. LREC (2010)
6. Fillmore C.: The Case for Case. In: Bach and Harms (Ed.): Universals in Linguistic Theory. New York: Holt, Rinehart, and Winston, pp. 1–88 (1968)
7. Chomsky, N.: Three models for the description of language. IRE Transactions on Information Theory (2), pp. 113–124 (1956)
8. Chomsky, N.: Syntactic Structures. The Hague: Mouton (1957)
9. Müller: Unifying everything: Some remarks on simpler syntax, construction grammar, minimalism and HPSG. Language 89(4), pp.920–950 (2013)
10. Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformations, Vol. 1: Foundations. World Scientific, Singapore (1997)
11. Jones B., Andreas J., Bauer D., Hermann K, Knight K.: "Semantics-based machine translation with hyperedge replacement grammars." In M. Kay and C. Boitet, editors, Proc. 24th Intl. Conf. on Computational Linguistics (COLING 2012): Technical Papers, pp. 1359-1376 (2012)
12. Jüger G., Rogers J.: Formal language theory: refining the Chomsky hierarchy. In: Philosophical Transactions of the Royal Society B. 367 (1598): pp. 1956-1970 (July 2012)