Computer Science Journal of Moldova, vol.**15**, no.3(45), 2007

An approach for NL text interpretation

Anatol Popescu, Sergiu Creţu

Abstract

For modeling the interpretation process of NL sentences we use the mechanisms implying semantic networks that assure syntactic – semantic text interpretation (SSI), including an understanding axiomatic model, interpretation model and denotation model to represent the result of SSI. These models estimate the correctness and the consistency of texts too. Also it implements an information extraction from texts in NL. Our approach based, mainly, upon semantic networks grammars has an extraordinary interpretation potential implying a system of completely new concepts and processing methods.

1 An axiomatic model for representation of NL text information

The problem of modeling the linguistic information has been widely treated in artificial intelligence (AI). It is the case to remark in this context the fundamental results obtained by Chomsky N., Hewitt C., Hunt E., Newell A., Schank R., Shaw A.,C., Winograd T., Woods W., A. etc.

The designing of systems with some elements of processing the given text in the NL has been focused on different domains of applicability: robotics, translation from one natural language to another, documentary searching systems (Information Retrieval – IR systems), extracting systems (Information Extraction – IE systems), question answering systems. In accordance with those works, it became clear that in order to elaborate a good informational technology some rules have to be assured:

- Elaboration of big knowledge bases, in order to stock the needed information for the mechanism of processing the NL text;
- Development of some techniques of interpretation and deduction, that would assure management of the implicit and the explicit information, contained in the text in NL, using the given information from our knowledge bases.

The implementation of both directions of designing needs conceptual and physical efforts.

Our work represents only a description of the principles and methods of designing a system of semantic – syntactical interpretation of the texts in NL.

By SSI of the texts in NL we understand the process of analysis and extraction of linguistic information, contained in the texts, used in a well determined context, assured by an adequate knowledge base.

The main difficulties that may appear in the process of designing are:

- The absence of a good linguistic corpus (because of the huge dimension);
- The complexity of the reconstruction of the elaborated grammars, using linguistic corpus;
- The difficulty of modifying the used grammars.

The system which we propose in our work, the ISS – GLOS, unlike other similar systems (LUNAR, ORACLE, MURAX, PROTOSYNTEX, etc.), interprets the text given in NL using a knowledge base, which might be accessed by a formal language – language of the knowledge base (LBC), designed *ad-hoc.*

The novelty of our proposal consists in the fact that the same formal language (LBC) is used both for designing the rules of the interpretation the text in NL and for the elaboration and the administration of our knowledge base. This uniformity of our work permits the development of some extensible techniques, extremely efficient in the regrouping of our system for new domains of applicability.

In order to realize the system of SSI we have made the following:

- The elaboration of the principles of modeling of understanding of given texts in NL, with the purpose of extracting and analyzing the information;
- Design of the process of storage the information with semantic significance;
- The elaboration of the principles of extraction and identification of the information stocked in our memorizing base;
- The implementation of some techniques of extension and adaptability of the elements responsible for the storage of information.

Syntactic – semantic text interpretation of the texts in NL, for our proposed system, is based on a model of understanding the sense of the analyzed text. The reader (the speaker) realizes this fact by the linguistic competence. There are three possible types of linguistic competence.

Definition 1.1. The linguistic competence of the speaker to produce sequences of words, syntactically correct, without any implications of the sense we will name as syntactical competence in general or just syntactical competence, where there are no ambiguities.

The second definition deals with semantics and has to be treated as the process of giving meaning to the linguistic unities. In this domain there exists sufficient confusion and different opinions concerning its elimination. We will adopt the following strategy.

Definition 1.2. We will name as semantic competence the linguistic competence of the speaker to establish semantic relations between different parts of speech: relations of inclusion – that contain the sense of the speech – anaphoric relations, etc.

It remains the last aspect of our linguistic competence – the denotative one.

Definition 1.3. We will name as pragmatic competence the linguistic competence of the speaker to assign the denotative sense to syntagmas, propositions and linguistic phrases.

We can make the following conclusion:

Assertion: the process of interpretation a text in NL supposes the ability of the speaker to manifest three linguistic competences, which have been mentioned above: syntactical in general, semantic and pragmatic.

This manifestation has an integrative character. It's hard to formulate in an explicit way the rules of the interaction of the syntax, semantics and pragmatics, in the process of understanding the text. We may suppose that the syntactical competence is a primary one. On its base it manifests the semantic competence of the speaker (the speaker may indicate the synonyms, the antonyms etc. and he has the conscience of the supposed anaphoric relations). And finally, the pragmatic competence is the result of the synthesis of our syntactic competence, our semantic competence, but also of the pragmatic competence which is anterior to the act of understanding, based on a pragmatic experience (stocked in a knowledge base, for example like some syntagmas with a more or less linguistic form). Their integration is effectuated by a formal, specialized language LBC, proposed during our work and which contains the essence of our proposals. The axiomatic model, which underlies the LBC is only a development of the other models used in order to describe the semantics of programming languages [1, 2, 5]. It stipulates the existence of the following entities – a finite set or an infinite one:

1) O – the totality of the objects going to be used;

2) M – the totality of mental acts – judgments, thoughts about the analyzed objects;

3) © – the composition of the mental acts.

We won't specify the physiological nature of our mental acts. It is important that when applied to the object from the set O, the mental act produces an object that belongs to the same set of objects O. It postulates the inefficiency of the application of the mental act, by admitting the existence of an object ω – the null object of the set O. It's admissible the application of a long chain of mental acts, concerning the object. In this case, the chains of mental acts are produced using the operation of composition ©. There exists a mental act I, which identifies the object itself. The model, which integrates the sets O, M may be defined as:

$$\mathrm{MA} = (\mathrm{O}, \mathrm{M}, ©),$$

where O, M and © are the entities from above.

In order to create some axioms for the MA model we have to make the following definition:

Definition 1.4. There exist the following sets:

1) T=$\{$ t $|$ $(\forall$ m) m © t $= \omega\}$ – the set of terminal objects – the objects with the sense fixed in a vocabulary or register. Every mental act, in this situation, is considered null (the ω object). We have to accept the interpretation proposed in the vocabulary, register;

2) E= $\{$ e $|$ e $\neq \omega \wedge (\forall$ m) m © e $= \omega\}$ – the set of elementary terminal objects. So, the null is eliminated;

3) C= O $\setminus$ E – the set of composed objects, non-elementary, non-terminal. Towards those objects we can formulate and apply the chains bound together by the operation ©.

The stipulated axioms for the MA model are:
A1. m © A $\in$ O
A2. (α © m) A=α (m(A))
A3. I © A=A
A4. $(\exists\,\omega)\,(\forall$ m) m©$\omega = \omega$
A5. $(\forall\,\omega\,[(\forall$ m) m©ω=A $\longrightarrow$ $(\forall$ A $(\exists\,\alpha)$ α ©A=ω)]
A6. $(\forall\,\alpha$, e) [α (A) = e $\longleftrightarrow$ α (B) = e] $\longrightarrow$ A = B

Axiom **A1** assures closing of the set O, concerning the mental acts. It's an important condition that simplifies the situation.

Axiom **A2** defines the technique of combination of the mental acts and the axiom **A3** stipulates the existence of a mental act, which identifies the object. The axiom **A4** postulates the existence of the null object and the axiom **A5** denotes the existence of the bound chains of mental acts, which may generate null objects. The equality of the objects (its interpretation) is dictated by the axiom **A6.** In order to define our axioms we have used the following notations:

- m is the mental act that belongs to M;
- α is the complex mental act got by the operation © and belongs to the model;
- A, B ... are names of the objects that belong to the set O.

Definition 1.5. Two mental acts α_1 and α_2 are dependent one on another, if there exist a third mental act β so that the following equalities are true:

$$\alpha_1 = \beta \copyright \alpha_2 \quad \text{or} \quad \alpha_2 = \beta \copyright \alpha_1$$

We will respect the tradition and we will denote this dependence – $dep(\alpha_1, \alpha_2)$, which is a predicate. The defined definition has the following properties:

1) The relation of semantic dependence in M* is reflexive, symmetrical and non-transitive.

2) $\text{dep}\,(\alpha, \text{I}) = \text{dep}\,(\text{I}, \alpha)$.

3) $\text{m}_1 \neq \text{m}_2 \longrightarrow \text{not}\,(\text{dep}\,(\text{m}_1,\ \text{m}_2))$

4) $\beta_1 \copyright \alpha_1\ =\ \beta_2 \copyright \alpha_2 \longrightarrow\ \text{dep}\,(\alpha_1, \alpha_2)$

5) $\text{dep}\,(\alpha_1, \alpha_2) \longrightarrow\ \text{dep}\,(\alpha_1 \copyright \beta, \alpha_2 \copyright \beta)$

6) $\text{dep}\,(\beta \copyright \alpha_1, \alpha_2) \longrightarrow\ \text{dep}\,(\alpha_1, \alpha_2)$.

We will name the expression $(\forall\beta)$ $[\text{not}\,(\text{dep}\,(\beta,\alpha)) \longrightarrow \beta(A) = \beta(B)] \longrightarrow$ A=B as the equality of two objects A and B, belonging to the set O and $\alpha \in$ M* and α (A)=α (B) $\neq \omega$.

This axiomatic model has served as a conceptual base, when we tried to define the formal language LBC.

The language LBC permits the defining and the usage of the needed knowledge bases in the process of interpretation the texts in NL. The usage of the knowledge bases for the process of generating the glosses, in order to interpret the texts, is a logical solution to the process of giving sense to the analyzed text fragments.

Conclusion: In the present paragraph we have formulated the following principles of interpretation the given texts in NL:

1. The process of memorizing the information with a high semantic charge, as the linguistic information is, will be effectuated with a semantic-syntactic interpretation, with the add of a knowledge base, which forms the semantic context of the interpretation, built in accordance with the axiomatic model of understanding the texts, presented above;

2. The semantic field used at the interpretation and the mechanism of interpretation itself have to be compatible with the sense of the descriptive formalism. In this work, LBC assured this demand;

3. The knowledge base, used in the interpretation, needs huge dimensions, in order to assure the adequate semantic context. This demand is very hard to be realized, that's why a specific approach would be needed. This specific approach would assure the gradual accumulation of information (of the knowledge). That's why we need some extensible techniques.

2 A formal mechanism for syntactic – semantic text interpretation of NL

The formal mechanisms actually used for syntactic – semantic text interpretation (SSI) of NL may be divided in three categories: a) the mechanisms assure data extraction from texts in NL; b) the mechanisms fulfill statistical processing of texts in NL; c) the mechanisms implying semantic networks estimate the correctness and the consistency of texts. Our approach is based, mainly, upon semantic network grammars having an extraordinary interpretation potential. On the other hand for estimative needs we have applied a model on the modal logics results.

For modeling the interpretation process of NL sentences we have to adopt a system of completely new concepts and processing methods.

The process of text interpretation consists in the act of giving sense to strings of characters which form NL. Psychologists stress 2 levels of this hierarchical process: the morphological one and the syntactic one (the profound structure and the structure on the surface). The structure on the surface formulates the judgment in accordance with morphology. The analytical phase of the syntactic structure and the phase which consists in the act of giving sense interpose and can't be done separately, as in the case of different formal languages: logic, algorithmic, programming.

Affirmation: A specific formalism for syntactic-semantic interpretation must include some mechanisms assuring:

1) The representation of the profound structure of the sentences;

2) The representation of the structure on the surface of the sentences;

3) The process of assuring the existence and the functioning of a system of processes interpreting the dynamic aspects of understanding the texts in NL.

Syntactic-semantic interpretation of texts in NL comes to the act of giving sense to syntactic constructions in NL. So, we used simple semantic modified networks (SSMN). These networks realize the syntactic analysis of the sentence and attribute semantics as a system of cooperating algorithmic processes. This solution differs from the traditional way of constructing compilators and interpreters for different programming languages, where processing phases are consecutive. In order to perform the act of giving sense to the syntactic constructions of the formal languages the formalism known as transformers was widely used.

The most popular transformers are based on CF grammars, used to build the transitional networks. These networks, however, are not appropriate for the process of interpreting NL, because NLs presume agreements, on a morphological level, between different parts of the sentence, parts without a stable position which can't be surprised by CF grammars.

So, we can't apply this technology in the syntactic-semantic interpretation of texts in NL. Transforming grammars of Chomsky, proposed as a solution to this problem, proved to be too complex for NLs interpretation. Woods proposed to modify the transitional networks admitting some restrictions (the transition is possible only if it's in accordance with the given restriction) saved on the transitional arc.

So, the transitional networks obtain a semantic charge, similar to Turing's mechanisms.

That's why we will reinterpret and adapt Woods's proposals to the act of processing texts in NL.

SSMN is an oriented graph with marker along the edges and is defined in the following way:

Definition 2.1. SSMN is defined as a graph g formed from a limited set of N nodes and the set of edges H $\subset$ N $\times$ N.

Graph's nodes are indexed (natural numbers are used). The E $\subset$ N subset forms the subset of final nodes. The edges' marker function F is defined in the following way:

If $\sum$ and Δ are 2 limited sets of characters called input alphabet and output alphabet and IdN – non-terminals alphabet, then:

- F : H $\rightarrow$ (L$_t$ ®IdN), where ® is exclusive reunion operation,
- L$_t$ – the set of pairs $\{(\sum \cup \{e\}\ ) \times (\Delta \cup \{e\})\}$ and
- IdN – the set $\{(Id \mid Id \in IdN)\}$.

Each SSMN must have a name. The name of the network is assured by the one-to-one correspondence **R** function, defined on the set of SSMN networks:

- R: SSMN $\rightarrow$ IdN, where IdN is the non-terminals alphabet. The nature of these names is irrelevant. Symbol **e** represents the null element ({**e**} – the null string).

Definition 2.2. Semantic modified network (SMN) is an object formed from the following compounds:

- SMN=$\{\Sigma, \Delta$, SSMN, R(SSMN), Id$_0\}$, where $\sum$ – input alphabet, Δ – output alphabet, SSMN– the limited set of simple semantic transitional networks, which forms SMN, R(SSMN) – the set of names of simple networks, because each SSMN is accessed through its name and Id$_0$ – the name of the initial network.

For RSM we will define the inference relation in the following way:

Definition 2.3. As the result of inference we will accept the following pairs:

- Rez$_{SMN}$ $=\{(\xi,\ \mu)\ |(\mathrm{Id}_0,\ \mathrm{Id}_0)\ \longrightarrow\ *\ (\xi,\ \mu),\ \xi\ \in\ \Sigma^*,\ \mu\ \in\ \Delta^*\}$, where "*" represents the end of the relation $\longrightarrow$.

Operational semantics of RSM can be interpreted as a semantic translator as well:

- T=$\{\Sigma,\ \Delta$, SMN, Id$_0\}$

Constructively, the semantic translator includes the following elements:

- Input band, which represents a string of completed cells with symbols from $\sum$ alphabet (the cell can be also null);
- Output band is identical in structure with the input band, but symbols are taken from Δ;
- A device for the process of management of the limited memory;

- Auxiliary memory;
- 2 heads each one for each band, which have the capacity to read or/and write on the respective bands.

In order to simplify the process of understanding the definition we kept the notes for SMN, but they were interpreted in accordance with the list stated above (for example, for the input band – Σ, etc.).

Functioning of such a device is well known. Next we will reinterpret and adapt it for SMN (the third compound of the translator).

The state of the translator T is defined by the pair (Id, i), where Id is the name of the simple network and i – the index (integer) of the node of the network. The set of the states of the translator T is formed form the set of all the states of the simple networks – S_T.

The auxiliary memory contains symbols which present the relation between nodes. These symbols can be described as (Id, i, f), where i and f are the indexes of the initial (i) and final (f) nodes of the edge from the network with the name Id. We will consider A_T the set of these objects. The auxiliary memory represents a typical stack. The third static object is the state of the translator T, represented by the structure ((Id, i), α, γ, β), where (Id, i) is the current state of the translator, α – the string of the input symbols, β – the string of the output symbols, γ – the state of the stack.

The process of functioning of the translator is expressed through its states. There are possible the following typical situations of the process of functioning of the T:

a) ((Id, i), aα, γ, β)| ((Id, i) , α , γ, bβ), where the terminal a symbol is written on the input band and the edge of the simple network with the nodes i and j is marked with the pair (a, b) (b belongs to the output alphabet and a , b can be also null);

b) ((Id, i), aα, γ, β)| ((Id$_i$, 1), aα, (Id, i, j)γ, β), where the terminal symbol a is visualized on the input band and the edge of the simple network with nodes i and j is marked with non-terminal Id$_i$. (Id, i, j) symbol is written in the memory of the stack and the current state of the translator becomes the first state of the simple network Id$_i$ – the node with the index 1;

c) ((Id$_i$, f), aα, (Id, i, j)γ, η)| ((Id, j), aα, γ, η), where f is the final node of the simple network Id$_i$ and, in consequence, the translator passes in the state given by the stack.

There are no other situations.

The result of the functioning of the translator is defined in the following way:

Approach 1. The result of functioning is the pairs of strings:

- Rez$_\Lambda$ = {(x, z)|((Id$_0$,1),x, λ,λ) |— $*$ T ((Id$_0$,f), λ, λ, z))}, where Id$_0$ – the name of the simple initial network and f – the final state in this network. Asterisk „$*$" represents the end of the relation |—, and λ is the null string.

Approach 2. The result of functioning is the pairs of strings:

- Rez$_f$ = {(x, y)|((Id$_0$,1), x, λ, λ) |— $*$T((Id$_0$, f), λ, χ, y))}, where χ represents a string of symbols from the stack alphabet (the stack alphabet are the elements with the structure (Id, i, j)).

These two ways of defining the result of functioning of the translator are equivalent. So, we will use both of them.

Syntactic-semantic interpretations, proposed above, assume the existence of some precise suppositions concerning the input and output languages:

a) The input language is a text in NL (an annotated text in SGML or LBC);

b) The output language is a text in the language of the semantic interpretation of the initial text presented as a semantic network. Creating this text in LBC we produce, in fact, the syntactic-semantic interpretation itself;

c) It's essential that the order in which the text is interpreted and the order in which the results of interpretation are presented to should be the same (it's about the existence of, at least, one homomorphism).

Establishing a link between the concept of translator T and SM networks permits to determine the descriptive power of SSMN, proposed here to make possible syntactic-semantic interpretation of texts in NL and to substitute the inference with a dynamic transitional process from one state to another of the translator. This link will be discussed as a theorem. Before doing that we will operate some modifications in the definition of SSMN. The purpose of those modifications comes from the existence of a more semantic charged SMN. Firstly, definition 2.1 will be modified in the following way:

Definition 2.4. SSMN′ is a graph g formed from the limited set of nodes N and the set of edges H $\subset$ N×N. The nodes are indexed. The subset E $\subset$ N forms the subset of final nodes. Marker function F is defined:

- If Σ and Δ are two limited sets of symbols known as the input and the output alphabet and IdN – non-terminals alphabet then:
  - F: H $\rightarrow$ (L'_t ®IdN) $\cup$ Ex $\cup$ Proc, where ®is the operation of exclusive reunion,
  - L'_t – the set of pairs$\{(\sum' \cup \{e\}) \times (\Delta' \cup \{e\})$ and
  - $\sum' = \{(t, A(t)) \mid t \in \sum, A(t) \in \mathrm{Val}_t\}$,
  - $\Delta' = \{(r, A(r)) \mid r \in \Delta, A(r) \in \mathrm{Val}_r\}$,
  - $\mathrm{IdN}' = \{(\mathrm{Id}, A(\mathrm{Id})) \mid \mathrm{Id} \in \mathrm{IdN}, A(\mathrm{Id}) \in \mathrm{Val}_{Id}\}$

Each SSMN′ should have a name. The name of the network is assured by the one-to-one correspondence function R, defined on SSMN′ set:

- R: SSMN′ $\rightarrow$ IdN, where IdN is the non – terminals alphabet. The nature of those names of networks is irrelevant. Symbol **e** represents the null element ({**e**} – the null string).

Function A puts in correspondence a set of attributes to each terminal input-output symbol and to each non-terminal symbol (numerical, textual variables etc.). So, A(t) = $A_1 x A_2 x ... x A_n$, where A_i are corresponding attributes. Dimension n of this set is variable and depends exclusively from the value of the argument t. Each A_i has a variation domain with well-established values. More precisely, A(t, n) = $A_1xA_2x...xA_n$. For simplicity we will take the notation A(t, n) = $Val_{t,n}$, because the matter concerns values. When n is irrelevant we will write Val_t.

The set Ex is the set of the logic expressions situated on the edge of the network. The edge is read only if the expression on the edge is 'true'. The set Proc is the set of procedures attached to the edges which might be appealed if the transition on the edge of the network occurs.

Definition 2.5. SMN′ is an object formed from the following compounds:

- SMN′ = {Σ′, Δ′, SSMN, R(SSMN′), Id'_0}, where $\sum'$ – is the set of input symbols redefined above, Δ′ – the set of the output symbols redefined above, R(SSMN′) – the set o names SSMN′, Id'_0 – the name of the initial network.

Translator T′ includes the same elements defined above for the translator T. ρ has a specific meaning. This function is necessary because for NL the displacement of constituent fragments of text, with an undetermined place is specific. This function can be defined in the following way:

$$\rho : \mathrm{IdN}' \rightarrow \mathrm{REGI}.$$

The set REGI is an auxiliary memory. Each element of the set is in fact a stack. In REGI the parts of the analyzed sentence are memorized: subject, predicate, direct object etc, which follow to be interpreted. However, the stack from the definition of the translator T remains. It becomes the main stack, assuring the functioning of the translator T. In the linguistic model non-terminal symbols represent the noun group, adverbial clauses, etc. These non-terminals appear on RSSM′ edges. Each non-terminal of this type has a stack which denotes the part of the sentence expressed in this non-terminal. In order to be more cogent we will insist on the specifics of the function A, which puts in correspondence to each terminal or non-terminal a set of variables called attributes (in the case of the linguistic model attributes denote, mainly, grammar categories: number, case, gender etc.). Some attributes are static (for example, we can obtain the value for gender from the vocabulary), other are dynamic and need to be synthesized. Usage of attributes charges our semantic networks. Ex and proc elements from the edges of SSMN are expressions with attributes.

The relation between the symbols of the network and the sets of attributes are assured by the following rules:

Rule 1. Each terminal input-output symbol and each non-terminal symbol has a set o attributes and each attribute has a well defined set of values.

Rule 2. Attributes are divided in two subsets with a null intersection. One set includes the attributes called inherited attributes, the other one contains the synthesized attributes – with the value obtained and calculated on the edge.

Rule 3. For each inherited attribute on the edges of SMN$'$, the proc element calculates its value in accordance with the values of the attributes SMN$'$ for the current nodes, already read. For the initial set SSMN$'$ the Id_0' defines the initial values of the inherited attributes.

Rule 4. For each synthesized attribute of the SSMN$'$ the proc element of its edge calculates its value in accordance with the values of the other attributes of the network.

Rule 5. Synthesized attributes of the input symbols are calculated only in accordance with the values inherited from these symbols.

These rules assure L – the **attributivity** of the input-output grammars of the transitional semantic networks.

We will establish the relation between SMN$'$ and the translator T$'$.

Theorem 2.1. Translator T$'$ is functionally equivalent to a SMN$'$ (Definition 2.5), for which the stated restrictions of L-**attributivity** are satisfied (rules 1-5 stated above).

The relation of equivalence (from Theorem 2.1) implies a coincidence of the pairs of the translator's T$'$ interpretative strings and of the SMN$'$ (Definition 2.5, Approach 1, Approach 2).

But before proving this theorem we will precise the following requirements: a) the state of the translator; b) content of the auxiliary memory; c) typical transitional situations.

A. The state of the translator will be the object:

(Id$'$, Val$^{i*}_{id}$, i, ex$_{i*}$, proc$_{i*}$), where Id$'$ is the name of the RSM$'$ network, i is the node of the network and ex$_{i*}$ is the logical expression associated with the edge (i, *) and proc$_{i*}$ are the executable actions, and if the expression ex$_{i*}$ is "true", they are associated with the same edge. Additionally, the notation Val$^{i^*}_{id}$ contains the specification of the node i as well.

Commentary: The sign "*" is a **quantificator**, which specifies a certain edge situated between the edges belonging to the node i.

B. The auxiliary memory (the main stack) will contain (Id$'$, Val$^{ij}_{id}$, i, ex$_{ij}$, proc$_{ij}$, j) as a symbol. It's a simple extension of the auxiliary memory specified in the definition of the translator T.

C. The typical transitional situations from one state to another will be the following:

a) ((Id$'$, Val$^{i^*}_{id}$, i, ex$_{i^*}$, proc$_{i^*}$), aα, γ, η) |— ((Id$'$, Val$^{j^*}_{id}$, j, ex$_{j^*}$, proc$_{j^*}$), α, γ, bη).

The semantics of this transition is the following: if ex$_{ij}$ is 'true' then the terminal symbol 'a' is read; proc$_j$ is executed;

b) ((Id$'$, Val$^{i^*}_{id}$, i, ex$_{i^*}$, proc$_{i^*}$), aα, γ , η) |— ((Id$'_1$, Val$^{1^*}_{id1}$, 1, ex$_{1^*}$, proc$_{1^*}$), aα, (Id$'$, Val$^{ij}_{id}$, i, ex$_{ij}$, proc$_{ij}$, j) γ, η).

This situation is identical with the execution of the operation Push for the main stack. On that edge the non-terminal Id$'_1$ is read. The current state is changed and the procedures are executed for the new state.

c) ((Id$'_1$, Val$^{*f}_{Id1}$, f, ex$_{*f}$, proc$_{*f}$), aα, ((Id$'$, Val$^{ij}_{Id}$, i, ex$_{ij}$, proc$_{ij}$, j) γ, η) |— ((Id$'$, Val$^{j^*}_{Id}$, j, ex$_{j^*}$, proc$_{j*}$), aα, γ, η).

   In this case the situation stimulates the operation POP of the main stack. This popping process takes place when f is the final node. The demonstration of this theorem can be found in [3].

Conclusion: In accordance with the stated facts the following result can be formulated:

1. It was elaborated an interpretative model of attributing sense to texts in NL.
2. It was defined the semantic modified network as a tool of syntactic-semantic interpretation of the text in NL.
3. It was introduced, in order to present an algorithm for SSI of the texts, the notion of semantic translator with stack.
4. It was proven the functional equivalence of the semantic translative transitional network with the semantic translator with stack (Theorem 2.1.).

3 Denotative semantics of languages (general notions)

The theoretical basis used in the process of elaboration of an efficient model for SSI of the texts in NL is formed by the denotative interpretative scheme.

In accordance with this scheme the semantics of the texts in NL can be done defining some semantic functions, with the domain of definition representing the semantic networks of the interpreted texts and the domain of variation – the sets of semantic values (glosses). In order to realize these applications of the remembered domains, stated above, it's necessary the elaboration of an auxiliary semantic language S, which will save this attribution of sense to the texts in NL. This language conceives the SMN$'$ and is, in fact, a subset of the LBC language stated above. This scheme is largely applied in linguistics: a certain language is explained with the help of another language, with a clear semantics. Alternative solutions can be consulted in [4].

We will present such a language. But we need to make firstly an adjustment

Adjustment: The process of giving sense to the syntactic constructions of the NL can be compared with the process of consulting a certain vocabulary: the main concept is searched in the left column of the vocabulary. Then the textual fragment is consulted from the right column. If the right part represents an intelligible text, then this process can be repeated for the next examined concept from the interpreted text. Contrarily, the textual adjacent fragment serves as starting point for the next adjustment. Repeated processing of the concepts has a recursive character.

The adjustment above is in concordance with SMN$'$. We will resume the stated facts in the following thesis:

Thesis: All the actions which can be initiated while consulting the knowledge base trying to establish the sense of the interpreted textual fragment, can be described with the help of a multi-sort algebra.

We will discuss briefly the definitions in relation with multi-sort algebras.

Definition 3.1. A sort is an element s belonging to a limited set S, which serves as index for a family of sets $<\mathbf{P}_s>$, called **portant** sets.

Observation: The sort corresponds, generally, to the names of the RSMP$'$ networks. The family of sets $<\mathbf{P}_s>$ is a set of graphs representing the abstract syntax of the textual fragment. The abstract syntax, according to Bacus-Naur, is a limited collection of productions, with the right part beginning with a special symbol. It identifies the production (the name of the operational class) and is followed by one or more sorts. For example:

<Predicate>::= **Verb** <verb><name>

Definition 3.2. **S**-sorted signature $\sum$ represents a family of sets $< \sum_{\mathbf{w,s}} >$, where **w**∈**S*** (limited strings of the elements of the set **S**), and **s** ∈ **S**. Elements are considered operational symbols having **w** orientation and **s** sort.

Definition 3.3. If $\sum$ is an **S**- sorted signature, then $\sum$**-** algebra is a set $\mathbf{P}_s$ of portant sets with function f for each element $\sigma \in \sum_{w,s}$ and **w = s1...sn** as:

$$\mathbf{f}\sigma\mathbf{:}\ \mathbf{P}_{s1}\times\ \mathbf{...}\ \times\mathbf{P}_{sn}\rightarrow \mathbf{P}_{s}\mathbf{.}$$

A great importance in the process of giving sense to a text in NL has the notion of homomorphism between $\sum$**-algebras**, because the vocabulary practically represents an **S-sorted** $\sum$**-algebra.**

Definition 3.4. **A** and **B** are 2 **S-sorted** $\sum$**-algebras**. A homomorphic application of the **A-algebra** on **B-algebra** is a family of functions:

$$\mathbf{g}_s\ \mathbf{:}\ \mathbf{P}_S^A \rightarrow \mathbf{P}_S^B,$$

where $\mathbf{P}_S^A$ and $\mathbf{P}_S^B$ are the **portant** sets of **A** and **B**, which satisfy the condition:

$$\mathbf{g}_s(\ \mathbf{f}_{\sigma A}\ (\ \mathbf{p}_1^A,\ \ldots,\ \mathbf{p}_n^A\ )) = \mathbf{f}_{\sigma B}\ (\ \mathbf{g}_s(\mathbf{p}_1^A),\ \ldots,\ \mathbf{g}_s(\mathbf{p}_n^A)).$$

This result presents us the fact that in an interpretative process the semantic structure of the text is kept.

So, we can make the following affirmation:

Affirmation: The semantic language, assuring the interpretation of the text in NL in accordance with the thesis stated above should assure the **homomorphic** application from the Defintion 3.4. Without this each interpretation is inconsistent.

4 The structure of the semantic language

We have already determined (in the paragraph above) the sense and the properties of the semantic language in SSI of the texts in NL in the context of the **denotational** approach of semantics. This language is necessary to be represented by syntax and a semantic is well-determined, so it has to be formal. The semantics of the semantic language S has to be consistent and simple in order to be expressed formally. At the same time S has to be able to express the semantics of the texts in NL. As a starting point for the process of elaboration of such a language there were used typical actions determining the significance of a part of the speech consulting the knowledge base.

The formal languages are defined specifying the types and the structures of data assured by the language, using all the types of operations with data as operators, which represent procedures used to manipulate data. Semantic language S hasn't fixed types of data. It only possesses the techniques used to identify data. An important notion from the semantic language S is the notion of syntagma. Syntagmas in linguistics are in general groups of words, which can't be reduced to the semantic values of the compounds. Syntagmas denote groups of values. In the semantic language S they have a clear sense, but the sense of syntagmas in the S language can be synthesized basing on the sense of their compounds. The semantic language S contains the following types of symtagmas:

a) Executable syntagmas, giving a material result while executing certain strings of dynamic entities, separated by the **delimitator** ";";

b) Syntagmas of the semantic networks, having elements with a typical structure

Each type presented above will be specified as:

a) < executable syntagma> ::= <name of syntagma>:(<string of executable syntagmas>)

< string of syntagmas >::= <identificator of data> | <string of sintagma > ; <identificator of data >

<identificator of data > ::= <identificator of data without parameters > | <identificator of data with parameters>

<identificator of data without parameters> ::= <name of syntagma >.{*| < value > | <identificator of data>}

<identificator of data with parameters> ::= <identificator>(<string of parameters>)

< string of parameters> ::= <parameter> | < string of parameters>, <parameter>

<parameter> ::= <identificator of data >

b) <sintagma semantic network> ::= <name of sintagma>:(<name of executable sintagma > **IN**<string of sintagma>)

<string of sintagmas> ::=<executable syntagma > | <syntagma semantic network> | < syntagma>; <executable syntagma> | <string of sintagma>; <sintagma of semantic network>

<name of executable syntagma> ::= <name of syntagma>

<name of syntagma> ::=<identificator>

Examples:

(1) Identificator of data: **VOCABULARY.VERB.***

It should be interpreted as: all the verbs are identified (syntagma **VERB)** from sintagma **VOCABULARY.** The sign "*" plays the role of the **cuantificator** $\forall$ from mathematical logic.

(2) executable syntagma: **EXECUTE:(VOCABULARY.VERB.*)**

EXECUTE is the name of the executable syntagma. The result of the execution of syntagma consists of the process of finding all the verbs (specified by the identificator of data **VOCABULARY.VERB*** written in parenthesis)

(3) Syntagma semantic network:

```
CONTEXT :( NOUN IN
TRANS:VOCABULARY.VERB.TRANSITIVE.*;NOUN;
NOUN: VOCABULARY.NOUN.man;ADJ;
.
.
.
ADJ:VOCABULARY.ADJECTIVE.nice;TRANS
)
```

Semantic execution of sintagma CONTEXT take place in the following way:

Step 1. The string of syntagmas and identificators of data situated after the key-word IN are executed. In order to have access to the interior syntagmas it has to exist a name of interior syntagma identical to the name of syntagma situated before IN. If this restriction is not satisfied, then the execution of syntagma is ended.

Step 2. Contrarily, that syntagma is executed which has the name of IN specified before.

Step 3. The execution continues until there are names of interior syntagmas which can be accessed

It has to be noticed that syntagma CONTEXT might contain in interior other syntagmas CONTEXT. In this case it's valuable the rule of the nested blocks from programming languages.

Example:

The following sentences are given:

(A) The student studies.
(B) He turns on the computer.

The fragment above contains 2 different types of information: explicit and implicit.

Explicit information can be found using syntactic-lexical analysis. For example, 'student' is a noun, plays the role of the subject and does the action, 'he' is a personal pronoun, replacing 'student' and is used as a subject. So, the predicate of the sentence denotes the action executed by the ‘student’. The last part of the analysis represents the implicit information. Semantic analysis includes both types of information. Pragmatics of the concept ‘student’ for Romanian will be represented so:

Site student: (morf: morph, fam: name, place: adres, studies: location, gloss1:txt;

> **rel** pragmatica1 (**intra** studies, **intra** place, **intra** fam, **rezul**t glosa1);
>
> **rel** pragmatica2 (**intra** fam, place, **rezult** studies);
>
> **rel** pragmatica3 (**intra** studies, **rezult** fam, place)
>
>) morf.rg=’student’, morf.gender= ’m’, case=(’n’,’a’,’d’,’g’), clg=clg(gender,number,case),
>
> /morf.number=’s’, gloss = ’ person studying in a location used for studying’/,
>
> /morf.number=’p’, glossa =’ all persons studying in a location used for studying – all the students’/,
>
> gloss =’students studying in a location used for studying and their addresses’;
>
> morph:(rg, gender, number, case: txt, clg: num, glossary: txt;
>
> **rel** intra rg, clg result glossary)

Commentary: The presented site describes the concept of student with its compounds. Compound ‘morph’ contains the morphological characteristics of the concept. It presumes that the syntagma ‘morph’ is already present in the knowledge base or will be defined in the same site, as it is stipulated in the example above. Syntagma ”morph” contains the following compounds: rg (the generalized **roo**t of the **morpheme**), number, case and gender.

The compound ,,clg” represents a numeric code built upon morphological characteristics of the Romanian words..

Relations pragmatica1, pragmatica2, pragmatica3 specify semantic network edges of the concept ,,student”.

Conclusion. This work offers a conceptual basis for implementation of a syntactic – semantic NL text interpretation system using a knowledge base. The following result can be formulated concerning this study:

1. An implementation scheme of NL text SSI was elaborated being based upon denotational techniques using the semantic language S;
2. Semantic language S is a subset of LBC language which assures knowledge base interface.

References

[1] S. Creţu. *Elaborarea unui mecanism formal de interpretare sintactico – semantica a textelor în limbaj natural*, in Proceedings of the 30th Annual Congress of the American – Romanian Academy of the Arts and Sciences(ARA), Chişinău, 2005, pp. 131 – 133.

[2] S.Creţu. *A system for natural language text syntactic – semantic interpretation (SSI)*, in the 2^{nd} supplement of the review Informatica Economică, International Conference Knowledge Management: Projects, Systems and Technologies, Bucharest, vol. 1, November, 2006, pp. 171 – 174.

[3] S.Creţu. *Interpretarea semantică a textelor în limbaj natural*, Edu-Soft, Bacău, 2007, 208p.

[4] A.Popescu, V.Cijacovschi, V.Procopovici. *Machine translation without post-editing*, Probleme de lingvistică inginerească, ULIM, Chişinău, 2003, pp. 161 – 187.

[5] A.Ollongren. *Programming languages determination by interpretive automata*, Mir, Moscow, 1977, pp. 99 – 127.

Anatol Popescu, Sergiu Creţu

Received October 17, 2007

Anatol Popescu
Technical University of Moldova

Sergiu Creţu
E.S. Academy of Moldova
E–mail: *srgcretu@yahoo.com*