relations, didactic relations (e.g. explanations and annotations), key signs, the list of segments that comprise the subject domain. After those elements comes the text of the segments themselves. The order of the segments is specified by the author of the subject domain specification, which eliminates the need to determine their order automatically.

Having such a specification will improve the quality of the resulting natural language text. However, in order to allow for translation of an arbitrarily defined fragment of the knowledge base, as well as to define the order of elements within the aforementioned substructures, an algorithmic way of deriving the order is needed.

However, the elements listed above (substructures of the original fragment of the knowledge base (in this case, the sc-text of a subject domain)) contain elements of their own, which also need to be linearized. Therefore, at the second stage, we derive the order of elements within such substructures based on the concepts contained in them. This process includes:

- Derivation of the order of concepts in the structure to be translated, i. e., the order in which their semantic neighborhoods should be translated;
- Derivation of the order of elements within such semantic neighborhood.

At the first step we propose to build a tree (graph) of dependencies between concepts according to the relations between them, and then to use this tree to derive the order of elements. For example, if a fragment has multiple classes of objects, then the first to be translated should be the supersets, followed by the subsets; sets should be translated before their elements, and so on.

At the second step (derivation of the order of elements within the semantic neighborhood of each concept) we propose to use a predefined order of relations and parameters. This will allow us to specify, for example, that when translating the semantic neighborhood of a concept the first elements to be translated should be the concept's identifiers, then its definition, then its membership in different sets followed by all the subsets of the given concept. There can be multiple potential variants of such a specification, depending on the class of the fragment.

We should note that this agent, and the structure specifications used by it, can be utilized not only for translating fragments of the knowledge base into a natural language, but also for translating them into other variants of linear representation of sc-code, for example, SCn.

C. Abstract sc-agent of generating an equivalent natural language text

As mentioned above, this agent in turn includes the following agents:

- Abstract sc-agent of generating a rough version of a natural language text
- Abstract sc-agent of converting the rough version into a correct natural language text

The input of the first agent is a structure to be translated, while the output is an ostis-system file with the resulting text. The text obtained as a result of this agent's execution may not fully correspond to the grammar of a particular natural language (in our case, English).

This approach explicitly sidesteps a much more complex task of microplanning (i.e. mapping of certain information in the semantic representation to the verbalization of this information) [12]. Instead of solving the problem of generating referring expressions, lexicalization, and so on, at this stage we propose to use a straightforward approach of using a finite set of specific rules of translating sc-code expressions into a natural language.

Currently, the agent has a simplified variant of implementation that is reduced to implementing a number of translators, each of which is dedicated for processing certain sc-code constructions (e. g., parameters of elements, their relations, etc.) Every construction has a corresponding natural language verbalization that is used during the translation. An example of a construction to be translated can be seen in figure 3. 1

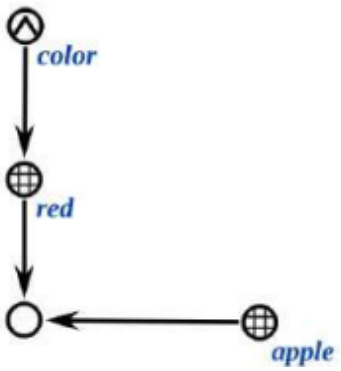

Figure 1: An example of a construction to be translated.

The membership arc corresponds to the English is, while the parameter in this case is translated as a pair (parameter, value). Therefore, the construction above will be translated by this agent into the following rough natural language text: *is apple, color red.*

This implementation has been chosen because it is relatively uncomplicated. In the future, we plan to elaborate it in way that the agent would arrive at a rough natural language verbalization algorithmically using formalization of natural language syntax proposed in our earlier work [15].

Complete rule-based algorithmic translation of knowledge base fragments into an adequate and coherent natural language text appears to be an infeasible task due the complexity of decision-making at each stage of the process. This is exemplified by the fact that the practice of designing fully rule-based intelligent systems has been largely supplanted by application of neural network-based

solutions, which, in the case of large language models, outperform all currently available methods of automatic natural language text generation.

It is for this reason that we propose to introduce the *Abstract sc-agent of converting the rough version into a correct natural language text.* This agent is implemented using a large language model. Its input and output arguments are ostis-system files. The input file contains the original rough natural language text that needs to be transformed, and the output file contains the text generated by a large language model.

Using a large language model is a convenient alternative to rule-based generation because it allows to sidestep certain sub-tasks of generating a coherent natural language text, such as choosing particular means of cohesion and coherence, which allows us to reduce the task to forming an ordered set of substructures that sets the order of segments of a coherent text, while individual verbalization choices are made by the large language model, which they in general excel at [16].

Using an intermediate representation for now also increases the likelihood of obtaining acceptable results without language model hallucinations [1], since the model in this case is not utilized in a zero-shot scenario and is provided with an extensive context that has a formal nature.

Thus, to generate the resulting natural language text we propose to use and intermediate representation (the output of the agent of generating a rough version of a natural language text), which is necessary at this stage because existing neural network solutions cannot be directly integrated with knowledge bases of ostis-systems. In the future, the OSTIS Technology will have support for "native" representation of neural networks as well as the means of preprocessing the input for traditional neural networks in such a way as to enable them to handle sc-code constructions [17]. This will eliminate the need for translating fragments of the knowledge base into intermediate variants of representation, and will enable us to use the actual sc-text of a knowledge base fragment as input for a large language model.

1 IV. Potential applications

Finally, we would like to discuss potential applications of the natural language generation module described above. These are three main ways in which it can be used:

- Exporting an arbitrary fragment of the knowledge base in a natural language;
- Navigating the knowledge base in a natural language;
- Dialog with an ostis-system using a natural language.

A. Exporting an arbitrary fragment of the knowledge base in a natural language

In this scenario, the fragment to be exported is specified by the user manually. For this application, the corresponding ostis-system can support various existing natural language text formatting styles.

One potential benefit of translating arbitrary fragments of the knowledge base into a natural language is that it makes it possible to use knowledge bases appropriately as the primary means of storing knowledge. Whereas, traditionally, knowledge has been stored mostly in natural language texts of various kinds, having a system that allows to translate formalized representations of knowledge into natural language texts on demand will significantly help with complex automation of various types of human activity [18].

B. Navigating the knowledge base in a natural language

The main way to navigate he current OSTIS Metasystem interface [19] is by navigating semantic neighborhoods of elements and/or other constructions. The external languages of sc-code representation used for outputting the content of the Metasystem's knowledge base are SCn and SCg [3].

It is possible to introduce a new way of navigating knowledge bases of ostis-systems whereby the fragments are translated into a natural language, which makes it possible to interact with ostis-systems effectively for users who are unfamiliar with the languages of external representation of sc-code.

This application would require additional work on the translation module in order to allow for hyperlinks within the natural language text markup, which will make the navigation easier.

C. Dialog with an ostis-system using a natural language

We plan to provide for the ability to communicate with an ostis-system using a natural language by implementing a question-answering support subsystem for users of the OSTIS Metasystem [19]. This subsystem should allow the user to ask questions about any knowledge stored in the Metasystem's knowledge base and get a response in a natural language.

The pipeline of this subsystem can be decomposed into the following stages:

- Message classification and question argument identification;
- Response generation;
- Translation of the response into a natural language using the means described above.

During the ongoing implementation of the prototype of this subsystem we have decided to use one of the existing neural network-based classifiers for the task of message classification and question argument identification: Rasa [20], Wit.AI [21], and others. We consider

Rasa to be the preferable option due to the possibility of local deployment and its open-source nature.

This approach has been chosen in order to obtain quickly a working prototype of the system. In the future, neural network-based classifiers can be replaced with an sc-agent of natural language understanding based on the approach discussed in [15].

The input of the response generation agent is a message that has been classified, while the output is a structure from the knowledge base that is an appropriate response to the message. An example of message classification received by the agent is available in figure 4.

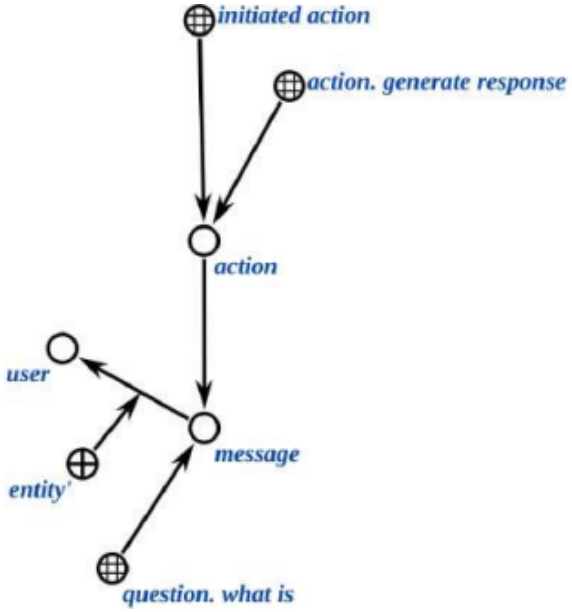

Figure 2: An example of message classification.

The agent operates in two steps:

- firstly, it tries to formulate a response using search templates;
- secondly, it tries to formulate a response by executing an appropriate action, in case the first step was unsuccessful.

The first step is introduced because responding to certain user questions can be reduced to searching for a relatively simple construction in the knowledge base, which can be implemented by mapping the corresponding classes of questions to certain search templates, as well as mapping message arguments to variables contained in such search templates. The response is a structure that contains the result of search by a template that corresponds to a certain class of questions after variables have been replaced with the corresponding question arguments.

Extending the set of questions that can be answered using search templates is an uncomplicated task that does not require modifying the problem solver. This task is reduced to introducing a new search template and specifying its connection with a class of questions and its arguments.

However, such constructions may be difficult to describe using one search template, or the answer may not be reducible to simple search and may require detailed transformations. For this reason, the second step is necessary.

If formulating a response using search templates is impossible, then the system searches for classes of actions connected to the corresponding class of questions by the relation *response action**. An instance of such action is then created with the corresponding argument received in the question.

Let us illustrate this using the question *What is X?* as an example.The response to such questions is a description of a certain element in the knowledge base, i. e. its semantic neighborhood. An example of the connection between a class of actions and a class of questions described above can be seen in figure 5.

In order to handle questions with two or more arguments that have different roles, the roles in the message can be mapped to respective arguments of actions in the knowledge base.

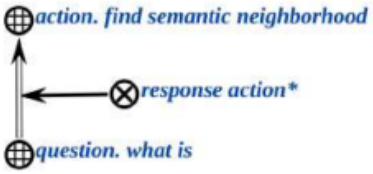

Figure 3: An example of the mapping between a class of actions and a class of questions.

Thus, when the agent receives the message illustrated in figure 4, an instance of the action of finding a semantic neighborhood is created. Then the problem solver waits until the agent is executed, and the agent's response is then connected with the message. An example of a construction obtained in this way can be seen in figure 6.

V. Conclusion

We have provided a sketch of the architecture of a module for translating fragments of ostis-system knowledge bases into coherent natural language texts. Our proposed approach subdivides the task of generating a natural language text into three sub-tasks: structure filtering, knowledge base fragment decomposition, and generating an equivalent natural language text.

The most important sub-task is knowledge base fragment decomposition because it ensures cohesion and coherence of the resulting natural language text. We have proposed two preliminary ways of solving this task: specification of element ordering within a fragment of the knowledge base as a sort of schema of the overall structure of the resulting natural language text, and