Intelligent Tutoring System for Discrete Mathematics

Kanstantsin Shurmel and Artur Sharapov
Eugene Samokval and Vitaly Tsishchanka
Belarusian State University of Informatics and Radioelectronics
Minsk, Belarus
Email: shurmel.konstantin@mail.ru, artursharaser@gmail.com
samokvalz@gmail.com, vitalik.tsishchanka@gmail.com

***Abstract*—The article presents a model of intelligent tutoring system for discrete mathematics. The model of such system uses methods and tools designed to build intelligent tutoring systems for any discipline and easy integration of new disciplines into the existing tutoring system.**

***Keywords*—knowledge, knowledge base, intelligent systems, problem solver, interface, discrete mathematics.**

I. Introduction

Discrete mathematics is fundamental these days and finds wide application in various fields. These fields include logistics, geographic information systems, computer science, modeling of physical and mathematical phenomena. This variety of applications makes *discrete mathematics* attractive both to commercial organizations looking for process optimization and solving complex problems, and to non-profit organizations engaged in research and development of new methods and algorithms.

Moreover, there are many other fields in which *discrete mathematics* has potential for application, such as sociology, biology, chemistry, and economics. This emphasizes its importance and relevance in modern society. Hence, understanding *discrete mathematics* plays an important role in the progress of science and technology. Therefore, in order for people to learn this science in a convenient way, it is necessary to develop new teaching methods to make the educational material more effective and accessible.

Modern methods of tuition involve not only internal presence of the learner in a particular discipline, but also the possibility of distance learning. As a rule, many of those who already have higher professional education, wish to deepen their knowledge in the discipline of interest, to expand competence in a related professional field of activity and to obtain new skills and knowledge, giving the opportunity to occupy a more successful position in the professional environment.

The first mention of the concept of *intelligent tutoring systems* was defined in 1970 by J. Carbonell. More than 10 years later, real working *intelligent tutoring systems* appeared. The difference between *intelligent tutoring systems* and automated systems is that *automated system* is a *consolidated knowledge base*, based on the results of work with which the system gives the learner the results of correctly and incorrectly answered questions. In turn, *intelligent tutoring system* is aimed at the process of diagnosing learning, its correction. The essence of the work of such a system is not just in diagnosing the learner's mistakes, but also in issuing advice based on predetermined strategies of distance learning [1].

Intelligent tutoring system
:= [A set of software and hardware that uses *artificial intelligence* techniques to create interactive and adaptive educational tools. Such systems are usually able to adapt to the individual needs and knowledge level of each learner, offering personalized assignments, materials selection and feedback.]

Automated learning system
:= [A program or set of programs that facilitate or fully automate the learning process. They may include various functions such as organizing learning material, creating tests and assignments, and tracking student progress. Such systems are usually designed to optimize the learning process, reduce the time spent on routine teacher tasks, and improve learning efficiency.]

The main advantages of an intelligent tutoring system:

- personalized approach to learning, taking into account the individual needs and knowledge level of each student;
- the possibility of interactive classes and the use of visualization to explain theoretical concepts more clearly;
- automatic identification of students' weaknesses and suggestion of additional materials to reinforce the material;
- providing access to a wide range of educational

resources, including study materials, assignments, and tests;
- the ability to receive feedback from the system and teachers to assess progress and correct the learning process.

The main disadvantages of an intelligent tutoring system:

- limited ability of the system to adapt to the specific needs of students with different backgrounds and learning abilities;
- lack of accuracy in identifying student errors and failures, which may lead to underestimation of their real abilities;
- necessity of constant updating of educational content and algorithms of the system in accordance with changes in the discipline program and teaching methods;
- limited opportunities to interact with students outside the learning environment, which can make it difficult to solve individual questions and problems.

One of the best known systems in problem solving is **WolframAlpha** [2]. This system can solve problems from various fields, including graph theory problems.

This system supports functions such as graph editing, basic graph operations, providing examples with known graphs, and searching for elements in a subgraph.

However, the disadvantages of WolframAlpha are that it cannot provide step-by-step solutions to problems, it has a special query language that not every user can understand, and it does not provide training tasks for users.

ALEKS is an intelligent tutoring system that provides a customizable program for teaching students in different subject areas, including discrete mathematics [3]. The system uses an approach based on passive learning theory, which allows hypothesizing about a student's knowledge and determining the most effective learning path based on the student's learning data.

The ALEKS system uses mathematical algorithms that are the basis of its knowledge base and artificial intelligence technology. The ALEKS knowledge base is a system of concepts and tasks that are interconnected and divided into levels of complexity. Each assignment has its own unique difficulty level, which is determined based on the student's study data. This allows students to learn at their own pace and make the most efficient use of their time.

The disadvantages of this approach are its limited depth of understanding of topics and its inability to integrate with other systems. This can lead to limitations in using the system within broader programs of study and limitations in accessing deeper knowledge of discrete mathematics. In addition, a customizable curriculum may be less effective with students who already have a certain level of knowledge in discrete mathematics and do not require such customization.

Webwork is a system of online teaching materials [4]. This system allows teachers to create individual assignments and exercises that students can solve in real time, and also provides the ability to automatically create different versions of assignments and assessments, and use interactive elements such as graphs and visualizations to facilitate the understanding of the material.

The Webwork information base consists of mathematical formulas, algorithms for solving problems, and a database containing assignments and students' answers. The Webwork approach allows new assignments and answers to be quickly added to the database, which is its advantage. However, the disadvantage is the limited functionality, since Webwork cannot use complex mathematical objects such as graphs or matrices. Also, due to the lack of systematization and semantic structuring of knowledge, it is impossible to automate the process of creating tasks and generating answers, and in the absence of a certain problem in the database its solution will not be found.

Maple T. A. is a program that allows teachers to create and automatically check tests and assignments in discrete mathematics [5]. Maple T.A. uses a combination of several approaches. First, this system uses traditional mathematical methods to develop an information base, including the creation of test assignments and courses, and a semantic approach to create knowledge models and intelligent data processing systems. Second, it utilizes machine learning algorithms for adaptive learning and assessment of student knowledge, and incorporates automatic answer checking, manual evaluation, and proof checking. It is possible to add new mathematical objects such as graphs or matrices, which is an advantage of the system.

The system has limited functionality and no integration with other systems, which may limit its use in some areas. For example, in a business domain where it is important to have access to different systems and tools to solve problems, this limitation can be a significant disadvantage. In addition, the limited functionality may not meet the needs of users, which can lead to loss of interest and unproductive use of the system.

An intelligent tutoring system for discrete mathematics should not have the disadvantages inherent in the above systems, namely it should provide the ability to view step-by-step solutions, ease of integration with other systems, and ease of integrating new skills, knowledge, and themes into the existing system.

Also, the above systems are only able to solve a limited range of tasks that are predetermined by the developers, and the systems cannot generate tasks themselves for the user training.

II. Proposed approach

The proposed approach implies the developing of a system based on the *OSTIS Technology* and its basic principles to overcome these disadvantages [6].

OSTIS Technology is a complex *intelligent system* design technology based on semantic knowledge representation, which includes:

- library of generic, reusable and semantically compatible components and *intelligent systems* (components of *knowledge bases, intelligent problem solvers, intelligent user interfaces*);
- compatible semantic knowledge representation languages of various kinds, providing semantic compatibility not only for reusable components of *intelligent systems*, but also for entire *intelligent systems*;
- compatible semantic models of problem solving.

As a formal basis for knowledge representation within the framework of *OSTIS technology*, a unified *semantic network* with a set theory interpretation is used. Such a representation model is called *SC-code* (Semantic computer code). The elements of such a *semantic network* are called *sc-nodes* and *sc-connectors* (*sc-arcs, sc-edges*) [7]. A model of an entity described by means of *SC-code* is called a *sc-model*.

Intelligent systems developed with the use of the *OSTIS Technology* are called *ostis-systems*. Each *ostis-system* consists of a platform-independent unified logical-semantic model of this system (*sc-model* of a computer system) and a platform for interpretation of such models. In turn, each *sc-model* of a computer system can be decomposed into *sc-model of a knowledge base*, *sc-model of a knowledge processing machine*, *sc-model of an interface* and *an abstract sc-memory* where *SC-code* constructions are stored [7].

The creation of *sc-models of the knowledge base* is based on the ontological approach, which implies the creation of ontologies as systems of concepts describing a particular subject area [8].

The problem solver is the main part of the *sc-model of the knowledge processing machine*, which is built on the basis of a *multi-agent* approach, where the interaction of *agents*, called *sc-agents*, is carried out exclusively by means of *semantic memory*, which stores *SC-code* constructions [9]. This approach allows to ensure modularity and flexibility of the developed machine, and also provides the possibility of parallel execution of different knowledge processing processes.

III. Knowledge base

An important step in the design of *intelligent tutoring systems for discrete mathematics* is the creation of a *knowledge base* that will contain complete and structured information on *discrete mathematics*. However, in order for this *knowledge base* to be as useful and effective as possible, it is necessary to consider a number of criteria that will allow it to be evaluated.

Based on this, the following basic and most important criteria for analyzing a knowledge base were highlighted:

- *Knowledge base* for *discrete mathematics* can be organized as a tree structure, where each node represents a particular topic, or as a network structure, where each node represents a different theorem or algorithm. Evaluating the structure of the *knowledge base* may include analyzing the hierarchy of topics, the relationships between topics and individual elements of the *knowledge base*, and the ease of navigating the *knowledge base*.
- *knowledge base* for *discrete mathematics* should contain sufficient information about key concepts, theorems, algorithms, and applications. Assessment of the quality and completeness of the information may include verifying that all necessary definitions and theorems are present and that the information provided is accurate and reliable.
- *knowledge base* for *discrete mathematics* should be user-friendly and easily accessible for use by students, teachers and researchers. Evaluation of functionality and usability may include an analysis of the accessibility of the *knowledge base* and the ability to search for information.

An important factor affecting the efficiency and quality of the system is the structured nature of the *knowledge base*. For this purpose, it is necessary to have an organized hierarchy of partial *subject domains*, which are strongly connected with each other by many different relations related to some common (base) *subject domain*.

Figure 1 contains an example of the description of a **subject domain of graph theory**.

Determining the relations between concepts is an important step in the development of a *knowledge base*, and requires careful analysis and careful approach. According to *OSTIS Technology*, to achieve the best quality of the *knowledge base*, concepts and relations will be described using their *semantic neighborhoods*. The basic relations and concepts include identifier, definition, statement, inclusion and decomposition. The implementation of each relation and their representation in the *knowledge base* depends on the particular case.

The next step is to define the rules and principles of describing the elements of the *knowledge base*. If there are several concepts in the *knowledge base* that are related to each other, the correct description of these concepts and connections between them allows to avoid duplication of information and ensure the integrity of the *knowledge base*.

Figure 2 shows an example of the description of the concept **disconnected graph**.

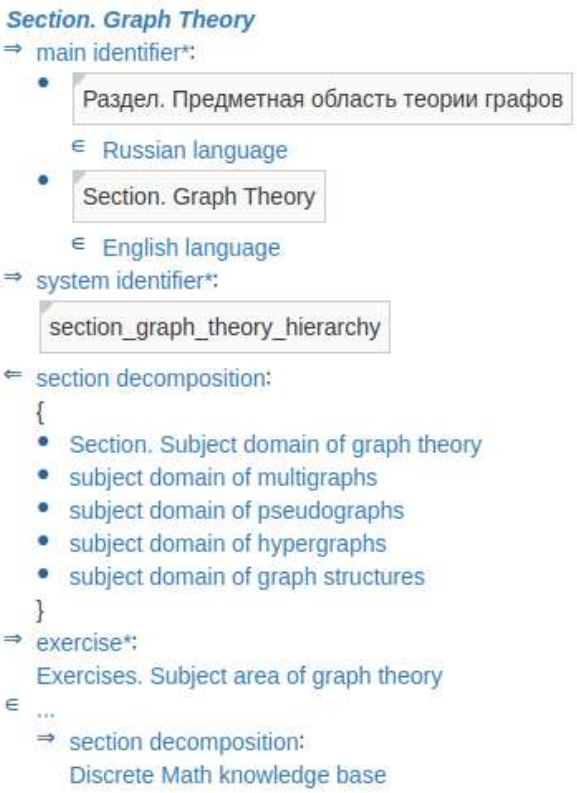

Figure 1. Section. Subject domain of graph theory

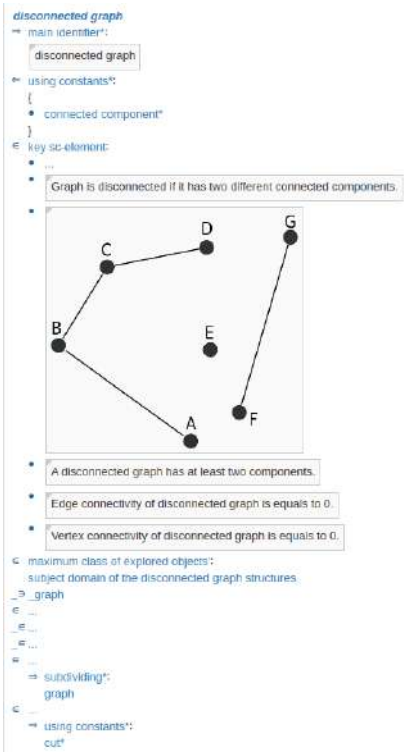

Figure 2. Concept disconnected graph

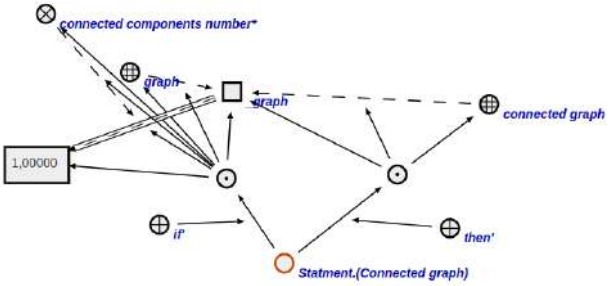

Figure 3. Statement about connected graph

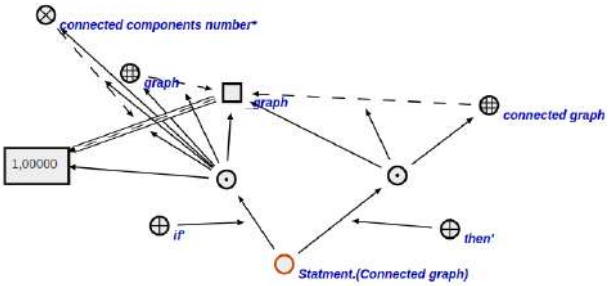

Figure 4. Specification of program for finding the union of two graphs

A unified description of the elements of the *knowledge base* creates an opportunity for its expansion in the future. The predefined rules and principles for describing elements will allow us to quickly and easily add new elements to the *knowledge base* without having to redesign its entire structure. Thus, we can be sure that system is flexible and scalable [10].

An example of a statement about a *connected graph* is shown in Figure 3, it is a logical formula written as an implication. The premise of the implication contains a pattern denoting that the graph has only one connectivity component. In the conclusion of the implication, the graph belongs to the set of connected graphs.

The *knowledge base* also contains specifications of programs that can be used by the *problem solver*. Thus, Figure 4 shows the specification of the *graph union finding program*. The result of this program is a graph that is the union of two input graphs.

An example of an exercise is shown on Figure 5.

IV. Problem solver

An important step in creating a *intelligent tutoring system* for *discrete mathematics* is to develop a *problem solver* that will solve problems in *discrete mathematics* with a full description of the step-by-step solution. But in order for the *problem solver* to fully fulfill its function, it is necessary that the *problem solver* meets the following criteria:

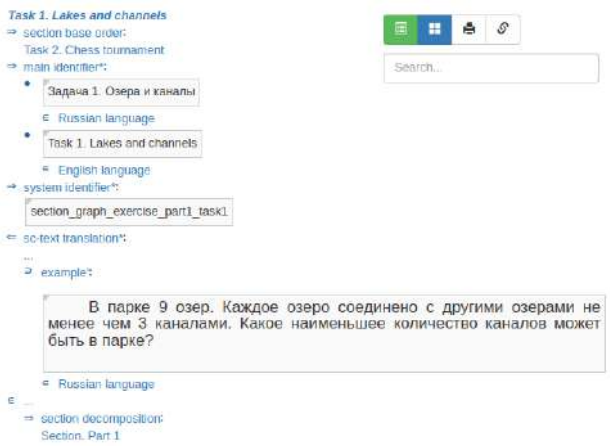

Figure 5. Exercise example

1) Correctness of solved problems is a key aspect when evaluating a *problem solver* in *discrete mathematics*. This means that the *problem solver* must be able to correctly solve any problem in its subject domain. This includes not only the ability to solve problems, but also the ability to adapt to different types of problems and conditions. If there are several ways to solve a problem, the solver should be able to choose between them and apply the most appropriate method depending on the specific task conditions. This may involve analyzing the complexity of different methods, evaluating their effectiveness, and determining the most optimal approach.
2) *problem solver* in an *intelligent tutoring system* should have the ability to describe in detail and step-by-step the process of solving the current problem. This includes a full or partial description of all algorithms used, which is critical for students to understand the logic and methodology of problem solving.
   Describing the problem solving process in detail helps students understand how to apply theoretical knowledge in practice. It also helps them develop critical thinking and analytical skills, as they can follow the problem solving process and understand how each step affects the final result.
   Describing the algorithms used is also an important part of the learning process. It helps students understand how different algorithms work and how they can be applied to solve specific problems. It can also help them develop programming and algorithmic thinking skills.
3) *problem solver* in an *intelligent tutoring system* should be user-friendly and feature-rich to ensure effective and productive learning. Usability may reduce the attractiveness of the *problem solver* as a learning tool, as it may increase the time and effort required to complete tasks, and thus may discourage users. Multifunctionality is also an important aspect of a *problem solver*. This means that a *problem solver* should have a wide range of features that can help users solve different types of tasks. Limited functionality may make the *problem solver* less useful to users, as it may not be able to solve all types of problems that users encounter.

Intelligent tutoring system for discrete mathematics problem solver consists of the the following modules:

- module for solving problems;
- module for generating and evaluating problem complexity;
- module for checking the correctness of the solution.

In the context of developing a *problem solver for an intelligent tutoring system for discrete mathematics*, *multiagent approach* can be used to implement different modules of the system, such as *problem solving module*, *solution correctness checking module*, *problems generation and complexity evaluation module* and others. Each module can be represented as an *agent* that performs its functions and interacts with other *agents* to achieve the goal of the system.

Designing a problem solving module. This module is part of the *intelligent tutoring systems for discrete mathematics* and provides solutions to problems based on a class of problems.

The functioning of this module consists of the interaction of *agents* from the following *sc-agent* hierarchy:

Abstract non-atomic sc-agent of problem solving
⇒ *decomposition of abstract sc-agent**:
{• *Abstract sc-agent of task specification generation by template*
• *Abstract sc-agent of solving a complex problem*
• *Abstract non-atomic sc-agent of solving a simple problem*
⇒ *decomposition of abstract sc-agent**:
{• *Abstract sc-agent of finding the relation of a given sc-element with a given concept*
• *Abstract sc-agent of unisg an unary operation*
• *Abstract sc-agent of using a binary operation*
}
}

Designing a module for generating and evaluating problem complexity. This module is an important part of the *intelligent tutoring systems for discrete mathematics* and provides problem generation and complexity evaluation. It is a tool that allows the generation of a variety of problems, taking into account different parameters and requirements, and at the same time estimating their complexity in order to adapt problems to the learners' level of knowledge.

Problem generation is an important function of this module, providing the ability to create tasks using specified parameters such as problem type, number of variables, constraints and other factors. This module generates unique tasks each time, promoting variety and fun for learners.

Problem difficulty evaluation is another important feature of the module, based on analyzing the generated problems and determining their difficulty based on specified criteria. The evaluation criteria may include the number of steps to solve, the use of complex algorithms or mathematical concepts. Conducting such an evaluation allows system to objectively assess the complexity of tasks and compare them.

Designing a module for checking the correctness of the solution. This module is part of the *intelligent tutoring systems for discrete mathematics* and is designed to check the correctness of the solution of problems. It analyzes the solution provided by the user and checks if it corresponds to possible solutions of the problem.

The main functions of the module are:

- module analyzes the structure of the solution, checks the presence of the necessary blocks of the solution, the correctness of their location and links;
- module checks the logic of the solution, analyzes the correctness of algorithms and logical operations used in problem solving;
- module checks the correct syntax of the program code in the solution;
- module checks the answer by comparing the obtained result with the expected one and determines whether the solution is correct or not.

Advantages of the solution correctness checking module:

- module allows system to automatically analyze the solution, which significantly speeds up the verification process and reduces the probability of errors;
- module is based on the specified conditions and requirements, which allows system to make an objective assessment of the correctness of the solution;
- module conducts a detailed check of all aspects of the solution, including structure, logic and syntax, which allows system to identify and point out errors.

The use of the solution checking module:

- the user provides their solution in the form of program code or algorithm;
- the solution validation module analyzes the provided solution with using specified algorithms and rules;
- the module displays the result of the check, indicating the detected errors or confirming the correctness of the solution.

V. User interface

The interface for *intelligent tutoring systems for discrete mathematics* is of particular interest because graphical representation of the main objects of study of graph theory and set theory, the two main components of *discrete mathematics*, graphs and sets, is the most convenient and effective for human understanding.

There are many software solutions related to the visualization of graph and set structures, but most of them are focused on solving specific highly specialized problems. In this connection, when it is necessary to solve a new problem, or conceptually the same problem, but from another subject area, the development of a new software solution for the task becomes the only way out, including the construction of its own visualization.

Proceeding from the fact that the most effective learning takes place in practice, when solving specific and possibly real-life problems, and through the acquisition of relevant experience, the tutoring system should be able to provide all the necessary tools and elements of the graphical interface for the appropriate practice of learners.

In this regard, a virtual space for working with graphs was developed - an element of the graphical interface, which contains a graph visualizer and editor, as well as elements of control and manipulation of graphs. This interface element allows creating, editing, and loading graphs stored in the *sc-memory* of the *sc-machine*, a software implementation of the semantic network storage and processing. In addition, the controls, which are graphical interpretations of the corresponding elements of the *knowledge base* stored in the *sc-memory*, allow to perform a certain set of actions on the graphs contained in the workspace. It is important to note that this set is defined exclusively by the description of the corresponding actions in the *knowledge base*, thus ensuring automatic and dynamic integration of new actions.

The graph editor used in the interface was designed to support the SCg alphabet and behave similarly to the SCg-editor during user interaction, but many improvements were introduced as well. The following were implemented: moving with touchbar, zooming with "pinch" on touchbar, etc.

In addition, the developed graph editor was designed in accordance with a modular architecture, where each element is an independent component responsible exclusively for its functions and extending the capabilities of the basic graph structure visualization component [11]. Such a solution is particularly time-consuming during design and initial development, but it allows to extend the capabilities of the editor in the future using the developed internal programming interfaces for connecting new components or plug-ins, without any modification of the source code of the editor or the visualizer. As a result, the implemented graph editor offers developers interested in using it not only the simplest and most efficient way to extend the editor functionality, but also an ability to customize the editor according to specific requirements,

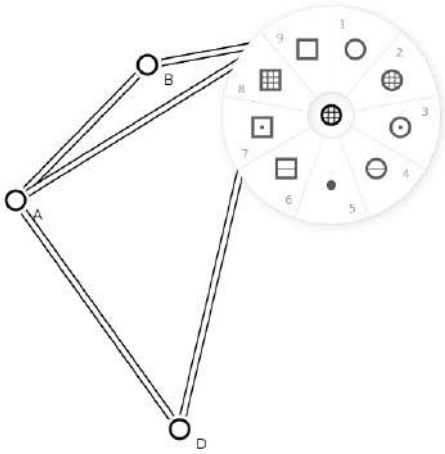

Figure 6. Graph editor example

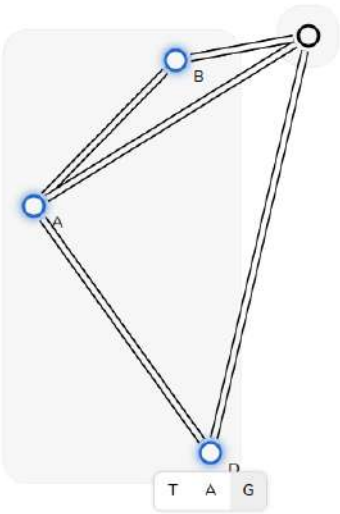

Figure 7. Floating menu example

excluding certain plug-ins built into the editor by default or changing their configuration.

An example of a graph editor is shown in the Figure 6.

One of the improvements of the graph editor is also a floating menu, supplied as a plugin (independent component) of the editor as an alternative to the classic menu. This decision can be justified by the following laws.

Fits Law allows to quantify the fact that the farther an object is from the current cursor position or the smaller the size of this object, the more time the user will need to move the cursor to it.

Hick's Law quantifies the observation that the more options of a given type you provide, the longer it takes to choose [12].

The floating menu appears automatically near the user's cursor when selecting certain objects, thereby minimizing the distance required for the cursor to overcome to perform a particular action on the selected objects. In addition, the menu is automatically hidden after receiving a signal indicating cursor movement away from it, provided that the user has pointed to this menu at least once, thus informing the user of its existence on the one hand, but not interfering with the user in his work on the other hand, conditionally guaranteeing that the user has paid attention to the existing menu under the selected objects.

An example of the floating menu is shown in the Figure 7.

In addition, depending on the type of selected objects in the floating menu, the system will offer only those actions that can be performed only on objects of the selected type. Thus, this solution allows system to significantly reduce the number of options for selecting an action and, consequently, the time required for the user to make a choice, which significantly improves the usability of the interface.

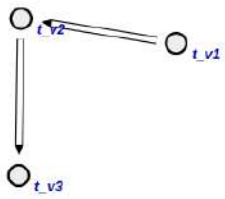

Figure 8. First graph

VI. Demonstration of results

As an example, the problem of determining whether the union of two graphs, shown in Figure 8 and Figure 9, is a tree is given.

Figure 10 shows a task template for determining whether the union of two graphs is a tree. This template specifies the input arguments of the problem and the goal of the solution.

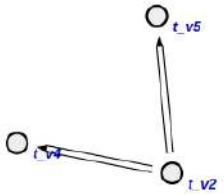

Figure 9. Second graph

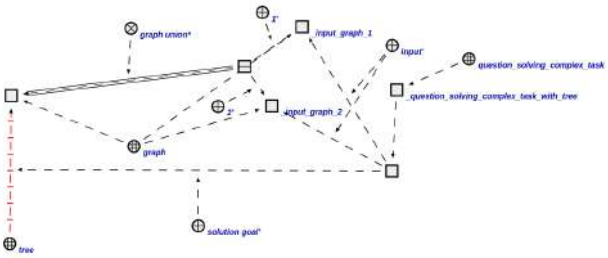

Figure 10. Task template

```
...
⇒ solution result*:
  (tree ∋ ...)
⇒ task solution*:
  ...
```

Figure 11. Result of problem

The result is shown in the figure 11. First the *sc-agent of task specification generation by template* created a specification of a task and then it called *sc-agent of solving a complex problem* which solved the problem because the *knowledge base* had the statement that *the tree is a connected acyclic graph*, the *problem solver* knows how to determine whether the graph is connected or not, whether the graph is acyclic or has cycle and the *problem solver* knows how to find the union of two graphs.

VII. Conclusion

Discrete mathematics is applied in various fields including logistics, geographical information systems, computer science, modeling of physical and mathematical phenomena, as well as sociology, biology, chemistry and economics, among others. Therefore, the development of an *intelligent system* to solve problems directly or indirectly related to discrete mathematics is of great relevance and importance in modern society.

Based on this work, the main components of *intelligent tutoring systems for discrete mathematics* such as *knowledge base*, *problem solver* and *user interface* have been identified and described. In addition to this, the requirements that all the components of the *intelligent tutoring systems* should follow and their functions were also identified.

Based on all the above, a prototype of *intelligent tutoring systems for discrete mathematics* has been developed. However, this is only the beginning, and options for further development include the implementation of user tutoring and a personalized learning approach.

Acknowledgment

The authors would like to thank the research groups of the Department of Intelligent Information Technologies of the Belarusian State University of Informatics and Radioelectronics

References

[1] A. Liliya, “Social technologies and processes,” *International scientific journal "BULLETIN OF SCIENCE" No. 1*, pp. 149–155, 2018.

[2] (2024, March) Wolfram|Alpha. [Online]. Available: https://www.wolframalpha.com/

[3] (2024, March) ALEKS - Assessment and LEarning in Knowledge Spaces. [Online]. Available: https://www.aleks.com/

[4] (2024, March) The WeBWorK Project. [Online]. Available: https://openwebwork.org/

[5] (2024, March) Maple T.A. [Online]. Available: https://www.maplesoft.com/

[6] V. Golenkov, N. Gulyakina, I. Davydenko, and D. Shunkevich, “Semanticheskie tekhnologii proektirovaniya intellektual’nyh sistem i semanticheskie associativnye komp’yutery [Semantic technologies of intelligent systems design and semantic associative computers],” *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, pp. 42–50, 2019, (In Russ.).

[7] V. V. Golenkov and N. A. Gulyakina, “Graphodynamic models of parallel knowledge processing: principles of construction, implementation and design,” *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, pp. 23–52, 2012.

[8] K. Bantsevich, “Structure of knowledge bases of next-generation intelligent computer systems: a hierarchical system of subject domains and their corresponding ontologies,” *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, pp. 87–98, 2022.

[9] D. Shunkevich, “Agentno-orientirovannye reshateli zadach intellektual’nyh sistem [Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems],” *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual’nykh system [Open semantic technologies for intelligent systems]*, pp. 119–132, 2018.

[10] A. S. Sharapov, “Designing knowledge bases of intelligent learning systems based on ostis technology,” *A young scientist*, pp. 17–19, 2023. [Online]. Available: https://moluch.ru/archive/478/105252/

[11] W. G. Griswold, M. Shonle, K. Sullivan, Y. Song, N. Tewari, Y. Cai, and H. Rajan, “Modular software design with crosscutting interfaces,” *IEEE software*, vol. 23, no. 1, pp. 51–60, 2006.

[12] V. V. Golenkov, N. A. Gulyakina, D. G. Kolb, “Intelligent user interface,” pp. 35–36, 2013. [Online]. Available: https://libeldoc.bsuir.by/bitstream/123456789/998/2/Golenkov_Int.pdf

ИНТЕЛЛЕКТУАЛЬНАЯ ОБУЧАЮЩАЯ СИСТЕМА ПО ДИСКРЕТНОЙ МАТЕМАТИКЕ

Шурмель К. А., Шарапов А. С.,
Самохвал Е. С., Тищенко В. Н.

В статье представлена модель интеллектуальной обучающей системы по дискретной математике. Модель такой системы использует методы и средства, рассчитанные на построение интеллектуальных обучающих систем по любой дисциплине и простую интеграцию новых дисциплин в существующую обучающую систему.

Received 01.04.2024