UDC 004.8:519.7

A.V. LYALETSKI

EVIDENCE ALGORITHM AND SAD SYSTEMS: PAST AND POSSIBLE FUTURE[1]

Abstract. The paper is devoted to the Evidence Algorithm programme on automated reasoning initiated by Academician Glushkov in 1970 and found its expression in the form of the Russian-language and English-language SAD systems intended for automated deduction. Some of their characteristic peculiarities and features are described. Examples demonstrating the possibility of their use for solving mathematical and common problems that require deductive constructions are supplied. Possible ways of the further development of the English-language SAD system are given.

Keywords: Evidence Algorithm, SAD system, automated reasoning, automated theorem proving, prover.

> The center of gravity of work in this area should be shifted from the construction of universal theorem-proving programs to the creation of programming systems and operating systems allowing, if necessary, to quickly code a proof of even a single difficult theorem and being capable, if needed, to work in a realtime with a mathematician proving this theorem.
>
> V.M. Glushkov

INTRODUCTION

In 1970, the Evidence Algorithm programme (EA) initiated by Academician V.M. Glushkov for constructing systems for automated theorem proving in mathematics was first presented in the paper [1]. According to it, such a system should contain an "evidence maintaining engine" possible to make a proof search in a mathematical theory in an environment of a formal mathematical language understood by a computer and being as close to languages of usual mathematical publications as possible. Besides, it should be able to accumulate an acquired knowledge and deepen on its basis its own notion of the evidence of a machine proof step for the enhancing of its deductive capabilities. In the case of necessity, the system should have a possibility to attract to a proof search a human for interactive managing a deductive process.

The beginning of work on EA dates back to 1962, when V.M. Glushkov first spoke about the possibility of a computer to prove a theorem. Since then, Glushkov's programme on automated theorem proving has been carried out at different times with varying degrees of success. In general, the entire time of research on EA can be divided into the following stages (details can be found in [2]):

1962–1969: pre-attempts to follow EA;

1970–1976: theoretical investigations in the framework of EA led to the appearance of the Russian language system;

[1] In honor of 50 years of the Evidence Algorithm announcement.

1976–1978: design and implementation of a Russian language system called System for Automated Deduction (SAD) by V.M. Glushkov in 1980 and having the abbreviation Russian SAD below;

1979–1986: trial operation and improvement of the Russian SAD system by means of the building-in of a human-like proof technique into it;

1992: stopping research on EA;

1998: renewing research on EA in the light of new paradigms and advances in automated reasoning;

2000–2002: design and implementation of an English-language modification of the Russian SAD system that will be denoted by English SAD in what follows;

2002–2008: trial operation and improvements of the English SAD system;

2009–present: theoretical investigations on proof search in classical and non-classical first-order logics.

And although the further development of the English SAD system was stopped in 2008, anyone can carry out a series of experiments with the system available online on the website "nevidal.org/sad.en.html".

EA AND RUSSIAN AND ENGLISH SAD SYSTEMS

According to EA, the scheme of actions of a mathematical computer service could be described as follows. A user communicates with the service using texts written in a high-level natural-type formal language. He may submit to the service a problem like "verify whether the given text is correct", or "what is the given text about", or "how to prove the following proposition", and so on. The text, provided it is syntactically correct, is sent to a service subsystem, a so-called "reasoner". The reasoner makes analysis of a problem under consideration and formulates a number of tasks submitting them to the service's logical engine being, as a rule, a prover. If the prover finishes the job, the result of its work (e.g. a proof verification trace) is displayed to the user and the work is over. If it fails then a diagnostic is made by the service and its result supplies to the reasoner for repairing the situation. In particular, the reasoner can decide that a certain auxiliary proposition might be useful and starts the search for those in existing mathematical archives. After finding it, the service begins a new proof search cycle with a modified problem and the process goes on.

In the framework of such an approach, there were designed and implemented the Russian SAD system in 1978 and the English SAD system in 2002.

The Russian SAD system is focused on automated theorem proving only, while the English SAD system can also verify both mathematical and other formalized texts presented in the so-called ForTheL language (FORmal THEory Language) [3] being as similar to the English language of usual mathematical publications as possible, which indicates the ability of the system for its use as a mathematical assistant in a daily practice of a human.

The ForTheL language can be viewed as a certain English modification of the Russian TL language (Theory Language) [4] being an input language for the Russian SAD system. As for the Russian SAD's evidence maintaining engine, it was realized on the basis of both the resolution technique and the sequent one using instead of the Kanger notion of substitution admissibility, a new notion proposed by the author of this paper in 1975 for optimizing quantifiers handling technique in the case, when the preliminary skolemization becomes an undesirable operation.

After the collapse of the USSR and decommissioning of the ES-lines computers in 1992, the research on EA and use of Russian SAD were stopped. Investigations on the EA programme were renewed in 1998 in the framework of the Intas-project "Rewriting

Technique and Efficient Theorem Proving", which led to the appearance of the English SAD system (System for Automated Deduction) in 2002.

The English SAD system is based on the declarative paradigm of the presentation of mathematical texts, containing axioms, definitions, theorems, proofs, etc., which provides by its input ForTheL language. That is why it can be viewed as an available mathematical assistant for theorem proving/verifying on the computer the correctness of a chain of conclusions that leads to solving a task under consideration.

The English SAD system has linguistic, reasoning, and deductive levels of the processing of input ForTheL-texts.

At the linguistic level, the system's parser makes a syntactical analysis of an input ForTheL-text, determines its structure, and defines its logical content encoded in ForTheL statements. After this, it compiles the ForTheL-text into its internal presentation. The result of translation is a number of goal statements to be sequentially deduced from their predecessors. Besides, there is a module for parsing a particular form of the first-order language, which can be used if necessary.

At the reasoning level, a goal task in question is splitting into a number of subgoal tasks for a prover. For this, English SAD either makes reduction of the main goal to several simpler subgoals or proposes an alternative subgoal. This module is redundant in the case, when the English SAD system solves the problem of proving a theorem.

At the deductive level English SAD can apply one of its provers intended for a proof search in classical first-order logic with equality. The native English SADs prover is based on a goal-driven sequent calculus exploiting the original notion of admissible substitutions. This permits to preserve the initial signature of a task in question so that equations accumulated during proof search can be sent to one or another specialized solver, e.g. to an external computer algebra system. Note that the English SAD system was implemented in such a way that it can use one of the external (w.r.t. English SAD) first-order provers such as Otter [5], SPASS [6], Vampire [7], or E Prover [8].

At the final stage, English SAD outputs the result of its session. Note a user can influence to solving a task under consideration by changing some system parameters.

Now, the English SAD system can perform the following:

- Inference Search: establishing of deducibility of a first-order formula/sequent;
- Theorem Proving: proving of a proposition in an environment of a ForTheL-text;
- Text Verification: verifying of a self-contained mathematical ForTheL-text.

Let us look at some of the distinctive features of the English SAD system.

Linguistic Features of English SAD. The following EA requirements to a formal natural language should be satisfied. It should admit writing such mathematical units as axioms, lemmas, auxiliary assertions, and theorems along with their proofs in order to provide the self-containedness of a text. Besides, it should give the possibility to introduce new definitions and use them. The language thesaurus should be separate from the language grammar, while the language should be close to the usual mathematical language for providing a comfort for a human in online writing and processing of a text. At that, it should give the possibility to write first-order formulas for establishing their validity in classical first-order logic. Additionally, it should admit writing tasks that do not have a direct relationship to the deduction process.

The first sketch of such a language was proposed in 1972, while its final Russian-language version TL was published in [4] in 1974.

The English-language "version" of TL under the name ForTheL was published in [3] in 2000. The main aim of the creation of ForTheL (and TL) was to provide an initial mathematical environment for solving deduction/verification tasks as well as for improving linguistic capabilities of an interface between a human and computer.

As in the case of usual mathematical texts, a ForTheL-text contains definitions, assertions, assumptions, theorems, proofs, etc. At that, the syntax of ForTheL sentences follows the syntax of usual English sentences (while TL simulate the Russian language). A ForTheL-text consists of sections, some service constructors, and phrases, where the phrases are either statements or assumptions. By definition, ForTheL sections can be sentences, cases, proof sentences, and top-level sections presenting signature extensions, definitions, axioms, lemmas, theorems, and proofs. A top-level section presents a sequence of assumptions concluded by an affirmation. Selections and attached proofs relate to sequences of low-level sections.

Such syntactical units as statements, predicates, notions (denoting classes of objects) and terms (denoting individual objects) are related to sentences. They consist of the following syntactical primitives: nouns, which form terms (e.g. "extension of") or notions (e.g. "subset of"), verbs, and adjectives, being predicates (such as "additive", "consists of", etc.), as well as symbolic primitives using symbolic notations for functions and predicates, allowing to construct usual logic first-order formulas in the form of ForTheL statements. Of course, only a little part of usual English is formalized in the ForTheL grammar.

Three kinds of sentences are in ForTheL: selections, assumptions, and affirmations. Assumptions are used for declaring variables or for formulating some hypotheses for a subsequent text; e.g. the following sentences are standard assumptions: "Any subset of any set is a set.". Selections claim the existence of representatives of notions; for example, they can be used for declaring variables. "Take a negative real number N." is an example of a selection. Affirmations are ForTheL statements (e.g. "If m is less than n and n is less than p then m is less than p.").

The ForTheL language also contains means for representing all the first-order logic formulas in the form of ForTheL phrases. The semantics of a ForTheL sentence is defined as a corresponding first-order formula being the result of certain transformations.

The following syntactically correct ForTheL-text containing a theorem together with its proof and concerning properties of compound natural numbers gives some presentation about expressive abilities of ForTheL:

Theorem. For all nonzero natural numbers n, m, p if $p * (m * m) = (n * n)$ then p is compound.
Proof by induction.
Let n, m, p be nonzero natural numbers. Assume that $p * (m * m) = (n * n)$.
Assume that p is prime.
Hence p divides $n * n$ and p divides n. Take $q = n / p$. Then $m * m = p * (q * q)$.
Indeed $p * (m * m) = p * (p * (q * q))$. $m < n$. Indeed $n <= m => n * n <= m * m$.
Hence p is compound.
qed.

It is possible to see that this text is easily readable and understandable by a human, who is not even familiar with the ForTheL language.

Deductive Features of English SAD. Deduction in the ForTheL environment is done in the following way. The ForTheL parser translates a self-contained input ForTheL-text, being a structural set of ForTheL phrases, into its internal presentation, being formulas of first-order logic. These formulas are input data for both the native prover and external provers. At that, the native prover of the English SAD system was designed so that the following deduction features are satisfied:

- deduction is goal-driven;
- the structure of an initial task is preserved;
- deduction is goal-driven;
- equality handling is separated from deduction;
- special equation-solving methods (i.e. the usual unification, AC-unification, etc.) and computer-oriented equality rules (e.g. the paramodulation) can be incorporated;

- heuristic proof methods used by a mathematician in his everyday practice such as the application of definitions and auxiliary assertions can be used;
- flexible interactive modes can be arranged.

All this was provided by using a special sequential formalism (see, for example, [9]), which, on one hand, made it possible to comply with the above-given features and, on the other hand, allowed constructing sufficiently efficient logical methods for finding proofs in the signature of an initial first-order theory. An additional reason for the selection of the sequent formalism was that sequent inferences are more like "natural" than inferences obtained, for example, by the resolution method. This feature becomes important when it comes to a human-computer interaction during deduction.

As it was pointed above, such investigations began in 1962, when V.M. Glushkov offered his view on the problem of automated theorem proving in mathematics latter called Evidence Algorithm, which led to the appearance of a specific logical procedure for proof search in classical first-order logic. It was based on the Kanger approach [10] to quantifier handling and was inferior in efficiency to resolution methods, which is caused by the fact that in the Kanger approach, there exists an additional step-by-step examination w.r.t. different orders of quantifier rules applications, while resolution methods avoid it due to skolemization.

Attempts to overcome this shortcoming led to the appearance of an original sequent calculus, which uses the above-mentioned new notion of admissible substitutions. It was incorporated in the Russian SAD system and its use demonstrated the helpfulness of such an approach to the construction of computer-oriented sequent calculi.

It was mentioned above that investigations on Evidence Algorithm were suspended in 1992 and were renovated only in 1998 in the framework of the Intas project "Rewriting Techniques and Efficient Theorem Proving". The project gave an impetus to the study of the possibility to construct sequent calculi in several directions, one of which touched classical first-order logic and the others also concerned non-classical ones (see, for example, [9,11]). As a result, there has been obtained a wide enough range of calculi for efficient proof search in classical and intuitionistic first-order logics as well as in their modal extensions. The research carried out for classical logic was used in designing and realizing the English SAD logical engine based on a sequent calculus that can be viewed as a further improvement of the calculus used in the Russian SAD system.

Another distinctive feature of the logical engine of English SAD distinguished it from Russian SAD is that in the case of desire, a user can use one of such well-known automated theorem-proving systems as Otter, SPASS, Vampire, and E Prover. In some cases, the made selection can essentially increase the speed of the proof search/proof verification process.

Information Environment Features of English SAD. Now, the information environment (the knowledge base in the current terminology) of English SAD is a set of language units confirming a self-contained text intended for theorem-proving/ text-verifying, some of which can be ontologically connected [12]. This gives the possibility to make essential influence on a machine-made proof step. Note that the English SAD system was designed in such a way that it is possible to connect with various mathematical services including, for example, computer algebra systems and solvers. It is possible to develop the existent information environment in the direction of the creation of program tools for its integration and use with various systems and styles of formal knowledge presentation and processing.

EXAMPLES

Below, two examples of the processing by the English SAD system of ForTheL-texts (available on the English SAD system website at the pages "nevidal.org/help-thm.en.html" and "nevidal.org/help-txt.en.html") are presented. The first demonstrates the ability of English SAD to establish the validity of a childish statement (given in Proposition). It concerns the relationships between animals and plants, is known as Schubert's Steamroller problem, and demonstrates a wide range of possible applications of English SAD not only for finding solutions of purely mathematical problems, but also for solving various "ordinary" logical tasks requiring only a correct deductive reasoning. The second one shows the ability of the English SAD system to verify a proof of a theorem given in its self-contained ForTheL-context. (In the first case, the Moses prover was used and in the second one, the SPASS prover was applied.)

Example of Solving Schubert's Steamroller Problem

[animal/-s] [plant/-s] [eat/-s]
Signature Animal. An animal is a notion.
Signature Plant. A plant is a notion.
Let A, B denote animals. Let P denote a plant.
Signature EatAnimal. A eats B is an atom.
Signature EatPlant. A eats P is an atom.
Signature Smaller. A is smaller than B is an atom.
Axiom CruelWorld. Let B be smaller than A and eat some plant. Then A eats all plants or A eats B.
Signature Bear. A bear is an animal.
Signature Fox. A fox is an animal smaller than any bear.
Signature Bird. A bird is an animal smaller than any fox.
Signature Worm. A worm is an animal smaller than any bird.
Signature Snail. A snail is an animal smaller than any bird.
Signature Millet. A millet is a plant.
Axiom Everybody. There exist a bear and a fox and a bird and a worm and a snail and a millet.
Axiom WormMillet. Every worm eats some millet.
Axiom SnailMillet. Every snail eats some millet.
Axiom BirdWorm. Every bird eats every worm.
Axiom BirdSnail. Every bird eats no snail.
Axiom BearMillet. Every bear eats no millet.
Axiom BearFox. Every bear eats no fox.

Proposition. There exist animals A, B such that A eats B and B eats some millet.

After processing this text, English SAD displays the result of its work with the task specified in Proposition:

[ForTheL] stdin: parsing successful [Reason] stdin: theorem proving started
[Reason] line 32: goal: There exist animals A, B such that A eats B and B eats some millet.
[Reason] stdin: theorem proving successful
[Main] sections 45 – goals 1 – subgoals 3 trivial 1 – proved 1
[Main] symbols 68 - checks 58 – trivial 57 proved 0 – unfolds 0
[Main] parser 00:00.00 – reason 00:00.00 – prover 00:00.00/00:00.00
[Main] total 00:00.01

Example of Verifying a Proof of a Theorem Relating to Number Theory

In this example, iif is an abbreviation for "if and only if".

[**number**/**-s**]
Signature NatSort. A number is a notion. Let A, B, C stand for numbers.
Signature NatZero. The zero is a number. Let X is nonzero stand for X is not equal to zero.
Signature NatSucc. The successor of A is a nonzero number.
Axiom SuccEquSucc. If the successor of A is equal to the successor of B then A and B are equal.
Signature NatSum. The sum of A and B is a number.
Axiom AddZero. The sum of A and zero is equal to A.
Axiom InjAdd. If the sum of A and B is equal to the sum of A and C then B and C are equal.
Definition DefLess. A is less than B iff B is equal to the sum of A and the successor of some number.
Let X is greater than Y stand for Y is less than X.
Theorem NReflLess. A is not less than A.
Proof.
Assume the contrary.

Take a number C such that A is equal to the sum of A and the successor of C.
Then the successor of C is zero (by AddZero, InjAdd).
We have a contradiction.
qed.

After receiving this text, English SAD establishes the correctness of the proposed proof and displays the below-given verification trace ended by statistical data:

[ForTheL] stdin: parsing successful [Reason] stdin: verification started
[Reason] line 63: goal: Take a number C such that A is equal to the sum of A and the successor of C.
[Reason] line 65: goal: Then the successor of C is zero (by AddZero, InjAdd).
[Reason] line 66: goal: We have a contradiction. [Reason] line 60: goal: A is not less than A.
[Reason] stdin: verification successful
[Main] sections 29 – goals 4 – subgoals 5 – trivial 1 – proved 3
[Main] symbols 48 – checks 31 – trivial 31 – proved 0 – unfolds 1
[Main] parser 00:00.00 – reason 00:00.00 – prover 00:03.40/00:00.39
[Main] total 00:03.41

CURRENT STATE AND POSSIBLE FUTURE WORK

At present, the English SAD system (http://nevidal/org) can perform mathematical text processing in the ForTheL environment according to the following scheme:

Text for proving a theorem or for verifying a theorem proof

$\Rightarrow$ (applying the ForTheL parser) *A self-contained set of first-order formulas*

$\Rightarrow$ (applying an English SAD prover) *Proving a theorem or verifying a theorem proof*

$\Rightarrow$ (applying an editor) *Text in a user-friendly form*

By now, a series of experiments have been conducted with the English SAD system. They relates to inference search in classical first-order logic, theorem proving in the ForTheL environment, and verification of self-contained ForTheL-texts. The most interesting examples concern verification, among which can be mentioned: Cauchy–Bouniakowsky–Schwarz inequality for real vectors, Ramsey's finite and infinite theorems, Bezout's identity in terms of abstract rings, Chinese remainder theorem, Newman's Lemma, Tarski's fixed point theorem, Furstenberg's proof of the infinitude of primes.

A trial operation of the English SAD system and a number of current achieve ments in automated reasoning have shown the desirability of improving the capabilities of English SAD in the following ways (studied and not implemented).

On the linguistic level. The nearest objective can be the incorporation of the existing ForTheL language into the LaTeX-environment in order to reach the reading of ForTheL-formulas in the form closest to usual mathematical texts. Besides, there are drafts of Russian and Ukrainian versions of the ForTheL language. Therefore, there exists the possibility to construct the next bidirectional translators: English ForTheL-texts $\Leftrightarrow$ Russian ForTheL-texts, English ForTheL-texts $\Leftrightarrow$ Ukrainian ForTheL-texts, and Russian ForTheL-texts $\Leftrightarrow$ Ukrainian ForTheL-texts, which will give the opportunity for using such a multilingual extension of English SAD by a user, who knows only one of these languages as well as for making an automatic translation from one of these languages into another. (Of course, one can try to construct a German, French, and/or other version of the ForTheL language, thereby strengthening this multilingual English SAD's component.)

On the reasoning level. The improving of heuristic possibilities of English SAD is presupposed to do by incorporating in it the human-like reasoning methods depending on the subject domain under consideration concentrating main attention on inductive theorem proving methods.

On the deductive level. Based on research made on computer-oriented proof search in classical and non-classical sequent logics and described in [9,10], one can try

to construct a toolkit giving the possibility to "puzzle" one or another system logical c depending on a desire of an English SAD user or a subject domain under consideration.

Additionally, it is presupposed to include cloud-based technologies in possible extensions of the English SAD system in order to improve the quality of its use for university e-learning.

CONCLUSIONS

Features of the English SAD system indicate that the system is designed and implemented in a way that takes into account the Evidence Algorithm's theses and modern achievements in the field of the building of computer mathematical services. In this connection, note that the basis of the ForTheL language has fundamental logical and set-theoretic operations and relations. Therefore, it is suitable for representing any (not only mathematical) texts, if they can be formalized by means of classical first-order logic.

Due to the great similarity of ForTheL to the languages of ordinary mathematical publications, the English SAD system working in the ForTheL language environment can be used for e-learning purposes such as verifying the correctness of mathematical ForTheL-texts writing by a student, which gives the possibility to learn him the rules of constructing correct mathematical phrases as well as to verify mathematical proofs written by a student in ForTheL, which leads to teaching him methods for carrying out correct deductive reasoning.

In the long run, the Evidence Algorithm approach and further development of the English SAD system can lead to the creation of an info structure for the remote multilingual presentation and complex processing of mathematical knowledge, which would make it useful for both teaching and academical daily activity of a human.

REFERENCES

1. Glushkov V.M. Some problems in the theories of automata and artificial intelligence. *Cybernetics and Systems Analysis*. 1970. Vol. 6, N 2. P. 17–27. https://doi.org/10.1007/BF01070496.
2. Lyaletski A., Morokhovets M., Paskevich A. Kyiv school of automated theorem proving: a historical chronicle. *Logic in Central and Eastern Europe*: *History*, *Science*, *and Discourse*. Lanham (Md): University Press of America, 2012. P. 431–469.
3. Vershinine K., Paskevich A. ForTheL — the language of formal theories. *International Journal of Information Theories and Applications*. 2000. Vol. 7, N 3. P. 120–126.
4. Glushkov V.M., Kapitonova Yu.V., Letichevskii A.A., Vershinin K.P., Malevanyi N.P. Construction of a practical formal language for mathematical theories. *Cybernetics and Systems Analysis*. 1972. Vol. 8, N 5, P. 730–739. https://doi.org/10.1007/BF01068445.
5. Otter homepage. URL: http://www.mcs.anl.gov/research/projects/other/.
6. SPASS theorm prover. URL: https://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/.
7. Vampire's homepage. URL: http://www.vprover.org/.
8. The E Theorem Prover. URL: https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html.
9. Lyaletski A. Evidence Algorithm and inference search in first-order logics. *Journal of Automated Reasoning*. 2015. Vol. 55. P. 269–284.
10. Kanger S. A simplified proof method for elementary logic. In: *Computer Programming and Formal Systems*. Braffort P., Hirchberg D. (Eds.). Amsterdam: North-Holland Publishing Company, 1963. P. 87–94.

11. Lyaletski A. Mathematical text processing in EA-style: a sequent aspect. *Journal of Formalized Reasoning* (*Special Issue*: *Twenty Years of the QED Manifesto*). 2016. Vol. 9, N 1. P. 235–264.

12. Paskevich A., Verchinine K., Lyaletski A., Anisimov A. Reasoning inside a formula and ontological correctness of a formal mathematical text. In: *Calculemus*/*MKM 2007 Work in Progress*, *RISC-Linz Report Series*. Kauers M., Kerber M., Miner R., Windsteiger W. (Eds.). Hagenberg, Austria. 2007. Vol. 07-06. P. 77–91.

Надійшла до редакції 06.07.2020

О.В. Лялецький

АЛГОРИТМ ОЧЕВИДНОСТИ І СИСТЕМИ SAD: МИНУЛЕ ТА МОЖЛИВЕ МАЙБУТНЄ

Анотація. Роботу присвячено програмі «Алгоритм Очевидності», що була ініційована академіком В.М. Глушковим у 1970 році і знайшла своє втілення у вигляді російськомовної та англомовної систем SAD, призначених для автоматизованого проведення дедукції. Надано опис їхніх характерних рис та особливостей. Наведено приклади, які демонструють можливість їх використання для розв'язання математичних і повсякденних задач, що потребують виконання дедуктивних побудов. Описано можливі шляхи подальшого розвитку англомовної системи SAD.

Ключові слова: Алгоритм Очевидності, система SAD, автоматизація міркувань, автоматизація пошуку доведень теорем, прувер.

А.В. Лялецкий

АЛГОРИТМ ОЧЕВИДНОСТИ И СИСТЕМЫ SAD: ПРОШЛОЕ И ВОЗМОЖНОЕ БУДУЩЕЕ

Аннотация. Работа посвящена программе «Алгоритм Очевидности», инициированной академиком В.М. Глушковым в 1970 г. и нашедшей свое воплощение в виде русскоязычной и англоязычной систем SAD, предназначенных для автоматизированного проведения дедукции. Дано описание их характерных черт и особенностей. Приведены примеры, демонстрирующие возможность их использования для решения математических и повседневных задач, требующих выполнения дедуктивных построений. Описаны возможные пути дальнейшего развития англоязычной системы SAD.

Ключевые слова: Алгоритм Очевидности, система SAD, автоматизация рассуждений, автоматизация поиска доказательств теорем, прувер.

Lyaletski Alexander Vadimovich,
Candidate of Physical and Mathematical Sciences, Senior Researcher, National University of Life and Environmental Sciences of Ukraine, Kyiv, e-mail: a.lyaletski@nubip.edu.ua.