I.J. Mathematical Sciences and Computing, **2018, 2, 1-11**
Published Online April 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2018.02.01

Available online at ***http://www.mecs-press.net/ijmsc***

Evidential Paradigm and SAD Systems: Features and Peculiarities

Alexander Lyaletski[a,*], Alexandre Lyaletsky[b], Andrei Paskevich[c]

[a] *Institute of Mathematics and Computer Science, 5, Academiei street, Chisinau, MD 2028, Moldova*
[b] *Taras Shevchenko National University of Kyiv, 60, Volodymyrska Street, Kyiv, 01033, Ukraine*
[c] *LRI, Paris-Sud University, bât. 650 Ada Lovelace, Orsay, 91405, France*

Received: 12 September 2017; Accepted: 13 February 2018; Published: 08 April 2018

Abstract

Research on automated reasoning systems based on a number of paradigms that support human activity in formalized text processing began in the late 1950s – early 1960s, when computer performance and memory space became sufficient for programming of complex intelligent processes. The so-called evidential paradigm was among them and it can be viewed as a way for integrating all reasonable paradigms oriented to the development of computer languages for representing formalized texts in the form most suitable for a user, formalization and development of the evidence of a computer-made proof step, creation of the information environment having influence on a current evidence of a machine proof step, and an active human-machine interaction. This work contains a brief description of the evidential paradigm and its implementation in the form of intelligent systems intended for the symbolic and deductive processing of mathematical texts focusing main attention on their features and peculiarities.

Index Terms: Automated reasoning, automated theorem proving, classical logic, intuitionistic logic, modal logic, Evidence Algorithm, formal language, formalized text, induction, number computation, symbolic transformation, deduction, paradigm, proof search, sequent calculus.

1. Introduction

Investigations on artificial intelligence concern not only the creation of devices that fully or partially simulate the human physical activity, they also touch upon questions and problems relating to the ability of a human to make reasoning in the framework of formalized languages suitable both for a user and computer.

Moving in this direction and in the direction of design and creation of high-performance intelligent systems,

* Corresponding author. Tel.: +380442293003
E-mail address: alyaletski@yahoo.com

now called automated reasoning systems and computer algebra systems, started in the late 1950s – early 1960s, when computers gained sufficiently high speed, information capacity, and program flexibility that the programming of complex human-like processes became possible. This led to the appearance of several paradigms of the intelligent computer processing of formalized knowledge.

Effective processing of formalized knowledge requires in-depth studies in the field of automated deduction and construction of linguistic tools designed to support everyday mental activity of an individual, which brings us to the task of finding a favorable combination of theory and practice.

Attempts to find a reasonable solution for this problem led to the creation and development of a number of specific paradigms having their reflection in intelligent systems, such as automated theorem-proving systems and computer algebra systems.

In Ukraine, such investigations were initiated in the beginning of 1960s by V.M. Glushkov, who announced the programme under the name Evidence Algorithm (EA) [1][1] intended for making research on automated theorem proving in:

(i) Formal languages for presenting texts (not only mathematical) in the form most suitable for a user,
(ii) Formalization and simultaneous development of a computer-made proof step,
(iii) Information environment (now called knowledge bases) having an influence on a current evidence of a computer-made proof step, and
(iv) Interactive human-computer proof search mode.

The modern vision of the EA programme was called the evidential paradigm [4].

The implementation of Glushkov's approach in the framework of the evidential paradigm has found its partial reflection in the form of the so-called English and Russian SAD systems intended for the presentation and processing of (formalized) mathematical texts. This paper is devoted to a brief description of the features and peculiarities of the evidential paradigm and English SAD system (System for Automated Deduction with the input language ForTheL [5] being a formal part of usual English and accessible at http://nevidal.org/), the prototype of which was Russian SAD [6] (see below for some details).

2. Remarks on Paradigms of Formalized Knowledge Processing

A number of paradigms – numerical, symbolic (analytical), deductive, inductive and integrative – are widely used to solve problems related to processing of formalized knowledge. In this section, we shall define the place of the evidential paradigm among them.

The *numerical paradigm* provides methods and tools to find approximate or exact solutions for problems of pure or applied mathematics. It is based on the construction of finite sequences of operations over finite sets of numbers. Usually, we start by constructing a mathematical (continuous) model for a task under consideration. After that, we perform transition from the continuous model to its discrete representation, which allows us to construct a variety of numerical algorithms, which are then implemented in software.

The *symbolic* (*analytical*) *paradigm* uses the ability of a computer to perform complex symbolic transformations, do numerical calculations, construct graphs of functions, transform curves, present certain processes as analytical models, etc. It is well-consistent with the empirical experience of a person, which is algorithmic and acquired by a human trying to find mathematical solutions for various natural science problems. The paradigm focuses on construction and use of computer algebra systems, and it emerged in the mid-1960's as a natural extension of the numerical paradigm, when high-performance computing became possible.

The *deductive paradigm* is based on the declarative way of representing and processing computer knowledge. In this paradigm, existing knowledge takes the form of formalized statements (axioms, definitions, theorems, etc.) and new information is obtained from the available one with the help of certain inference rules applied to

[1] A detailed enough description of EA and the investigations connected with it can be found in [2] and [3].

construct (deduce) new statements. Systems intended for representing and processing knowledge based on this paradigm are called reasoning systems, most of which exist as automated theorem-proving systems. Indeed, the deductive approach is the most relevant and efficient for solving tasks requiring reasoning "from general to particular" used in many application areas.

It is clear that in order for a reasoning system to provide deductive assistance to a human user, it needs to develop a corresponding linguistic "shell", correctly formulate axioms, hypotheses, and inference rules, and have the ability to generate assumptions that can be used.

The *inductive paradigm* is used on the transition from individual observations to general conclusions in the case of the natural sciences. In mathematics, it is based on the induction method applied to verify a given property for partially or fully ordered sets of objects.

The *integration paradigm* brings together all of the paradigms mentioned above. It can be divided into two types: (a) integration of the heterogeneous components and sub-systems at the design stage, and (b) integration at the operation stage, i.e. combining existing computer systems into a new one. A strong interest in the development of such an approach was caused by a widespread use of the Internet for data exchange.

As examples of the integration at the operation stage, one can cite Open Mechanized Reasoning Systems [7] and the OpenMath project [8] which offer specification and communication languages for theorem proving and symbolic computation.

The most famous representatives of the integration at the design stage are the QED project [9] as well as the Mizar [10] and Theorema [11] systems.

To this last category also belongs the Evidence Algorithm: it can be considered as the first representative of the integration paradigm at the design stage, which gave a reason for calling it the "evidential paradigm".

3. Evidential Paradigm Timeline

Below we give a short history of the evidential paradigm: its past and present beginning with its appearance in 1962 and ending by this time (see [2] for details).

Beginning of investigations (1962–1969). At that time, the first steps were made in the field of automated theorem proving in the EA style. The research efforts in automated theorem proving were mainly focused on the items (i) and (ii) from the Evidence Algorithm programme (see above).

As a result, the signature of the first-order language was extended by the symbol for the relation of set-theoretic membership and a dedicated, sound and complete, proof search procedure called the AGS (Auxiliary Goal Search) calculus was proposed in [12]. It was goal-oriented and used a Kanger-like approach to quantifier manipulation instead of skolemization. Another feature of AGS consisted in the separation of equality handling (equation solving) from deduction. In the year 1969, the propositional part of AGS was implemented on the BESM-2 computer.

1970–1992. During this period, the main efforts were still focused on the directions (i) and (ii), practically without taking (iii) and (iv) into account. The research in the framework of the evidential paradigm can be divided into two phases: foundational and implementational.

The *foundational phase* involved the construction of a formal first-order natural-like (Russian) language, which led to the creation of the so-called TL language (see below) and investigations on inference search in the classical first-order logic splitting into two directions: the development of the procedures based on resolution with paramodulation, and modification of AGS through incorporation of a new original notion of an admissible substitution [13,14] leading to an essential improvement of quantifier manipulation efficiency and, as a result, to the increased performance of sequent-based inference search in classical first-order logic.

These investigations were made in 1970–1975. In 1976, the *implementational phase* was started. The theoretical results obtained earlier provided the basis for the design and implementation of the Russian SAD system, which was firstly presented in 1978 at the All-Union symposium on artificial intelligence and automatizing of mathematical research (Kiev, USSR). From 1978 to 1982, the SAD system was being improved by adding various resolution-type strategies and AGS modifications to increase performance.

From 1983 to 1992, the Russian SAD system had been equipping with various heuristic techniques used by human mathematicians to prove mathematical statements. Numerous experiments had been made with the system at that time. In 1992, the works on the Russian SAD system was stopped because of the shutdown of the production of the ES-series computers, on which the system was implemented.

1998–present. In 1998, the research in the framework of the evidential paradigm was restarted due to the involvement of some of the former developers of Russian SAD in the Intas project "Rewriting techniques and efficient theorem proving" (1998-2000). Nourishing upon the ideas of the Evidence Algorithm project and the experience accumulated in developing and exploiting Russian SAD, the first version of the English SAD system had been designed and implemented by 2002 [15].

The English SAD system has a three-level architecture (see below) with the ForTheL language (being an English modification of TL) as its input language. Since its inception and up to 2008, the SAD system underwent a number of modifications that facilitate SAD interaction with the user and increase its efficiency. In 2008, the development of the SAD system was stopped. Since then, the research on the evidential paradigm focused exclusively on the theoretical aspects, specifically on computer-oriented proof search in classical and non-classical first-order logics [22] (see the section "Deductive features and peculiarities" for details).

4. Evidential Paradigm and Formal Text Processing: Features and Peculiarities

The evidential paradigm is mainly intended for formalized text processing such as proving a given theorem or verifying a user-written proof of one. According to the evidential paradigm, the scheme of processing of a formalized (not necessarily mathematical) text is as follows (cf. [2]):

Text prepared by a user to prove/verify a theorem
$\Rightarrow$ (using a parser) *A self-contained first-order text*
$\Rightarrow$ (using a prover) *A computer-made proof/verification*
$\Rightarrow$ (using an editor) *Text in a form comprehensible for a human*.

As a result, the following challenges arise in the framework of the evidential paradigm: design and implementation of formal mathematical languages, construction of deductive methods and tools for logical inference search, creation and use of an information environment, development of interactive modes.

Let us give a brief description of each of these challenges and some of the ways to overcome them. We focus our attention mostly on the linguistic and deductive tools fulfilling the requirements of the evidential paradigm and implemented in the English SAD system.

4.1. Linguistic Features and Peculiarities

We start with the list of requirements for a natural-like language that should be satisfied according to the evidential paradigm.

The syntax and semantics of this language must be formally defined. For a given mathematical theory, this language should allow writing its axioms, its auxiliary propositions, lemmas, theorems and their proofs, as well as new definitions to ensure the self-containedness of a text. The language thesaurus should be extensible and separated from the language grammar. In addition, it should be close to the language of usual mathematical publications in order to provide convenience and comfort to a human user in creating and processing a text in an interactive mode. Besides, it should permit writing formulas of the first-order language in order to establish their validity in classical first-order logic. Moreover, it should permit to formulate tasks that are not directly related to the proof search.

The first sketch of such a language (based on the Russian language) was proposed in 1970 [16]. Its final version called Theory Language (TL) was published in [17].

In 2000, a new, improved, English-language version of TL called ForTheL (FORmal THEory Language) [5]

was designed. The main objective of ForTheL (and TL) was to provide an initial mathematical environment for solving a deduction/verification task under consideration as well as for the further development of evidential engines and strengthening of their capabilities.

Like a usual mathematical text, a ForTheL-text consists of definitions, assumptions, affirmations, theorems, proofs, etc. The syntax of a ForTheL sentence follows the rules of the English grammar and uses the notion of a section. ForTheL sections are: sentences, sentences with proofs, cases, and top-level sections: axioms, signature extensions, definitions, lemmas, and theorems. A top-level section is a sequence of assumptions concluded by an affirmation. Proofs attached to affirmations and selections are simply sequences of low-level sections.

Sentences are constructed from such syntactical units as statements, predicates, notions (that denote classes of objects) and terms (that denote individual entities). Units are composed of syntactical primitives: nouns which form notions (e.g. "subgroup of") or terms (e.g. "closure of"), verbs and adjectives which form predicates (such as "belongs to", "compact" and others), symbolic primitives using a concise symbolic notation for predicates and functions and allowing to construct usual first-order formulas in the form of ForTheL statements. Naturally, just a little fragment of English is formalized in the syntax of ForTheL.

There exist three kinds of sentences in ForTheL: assumptions, selections, and affirmations. Assumptions serve to declare variables or to provide some hypotheses for a subsequent text. For example, the following sentences are typical assumptions: "Any subgroup of any group is a group.", "Assume that m is less than n.". Selections state the existence of representatives of notions and also can be used to declare variables. Here follows an example of a selection: "Take an even prime number N.". Affirmations are simply statements, e.g. "If p divides $n + p$ then p divides n".

The semantics of a ForTheL sentence is determined by a series of transformations converting it to the corresponding first-order formula for processing it by the logical tools of English SAD.

4.2. Deductive Features and Peculiarities

From the very beginning, Evidence Algorithm paid great attention to the development of machine proof search methods suitable for various fields of mathematics and allowing (formalized) human reasoning techniques. According to EA and the evidential paradigm, deductive methods and tools should satisfy the following requirements:

1) The syntactical form of the initial problem should be preserved,
2) Deduction should be performed in the signature of the initial theory, in particular, preliminary skolemization should not be required,
3) Proof search should be goal-oriented,
4) Equality handling should be separated from the deductive process.

With these requirements in mind, the sequent-based approach has been selected as the basis for the logical engines within the framework of the evidential paradigm. This provided a satisfactory answer to the first requirement. Furthermore:

- To satisfy 2), an original technique for the optimization of quantifier manipulation using a new notion of an admissible substitution has been developed,
- To satisfy 3), the generation of auxiliary goals during proof search has to be "driven" by the goal formula,
- To satisfy 4), special methods for equality processing and equation soling have been developed (admitting the use of computer algebra systems and problem solvers if necessary).

As we mentioned earlier, the investigations in this direction started in 1963, when the problem of automated theorem proving in mathematics was formulated by V.M. Glushkov. This led to the appearance of the AGS

calculus for proof search in classical first-order logic. It used the Kanger approach to quantifier manipulation and was less efficient than usual resolution-based methods.

Attempts to overcome this disadvantage led to the construction of an original sequent calculus [18, 19] based on the above-mentioned new notion of admissible substitution distinguished from notions that were used before its appearance. It was incorporated in Russian SAD and its implementation demonstrated the usefulness of a deductive approach of this type in construction and use of such calculi.

These investigations have stopped before 1992 and were resumed in 1998, when the authors became involved in the Intas project 96-0760 "Rewriting Techniques and Efficient Theorem Proving" (1998–2000). This project gave an impulse for modifying the calculus from [19] in several directions, one of which concerns classical logic (see, e.g., [20]) and the others concern non-classical ones. As a result, new research efforts have been carried out to obtain a wide spectrum of results on efficient inference search in classical and intuitionistic logics as well as in their modal extensions. Note that these results for classical logic were used in designing and implementing the logical engine of the English SAD system.

It is easy to show that the new notion of admissibility is not enough for constructing sound calculi in the intuitionistic case. This situation can be corrected by introducing the notion of compatibility firstly used in [21] to construct a sound and complete tableau calculus with free variables for intuitionistic logic and later, for a number of calculi for modal extensions of classical and intuitionistic logics [22].

4.3. Information Environment and Interactive Modes

Three types of information environments (now called knowledge bases) are distinguished in the evidential paradigm. The first one is a *proof environment* intended for keeping all the data necessary for solving a problem under consideration. By now, it takes form of a self-contained ForTheL-text prepared by a user. All mathematical facts accumulated during previous sessions and needed to solving a problem constitute the second type of environment called an *internal environment*. The whole information that can be received via Internet by using mathematical services existing in the Web give the third type, a so-called *external environment*.

Interactive proof search modes should serve for interfacing systems (in particular, the English SAD system) with various mathematical services existing in an external information environment. They can be designed and implemented only after fixing the form of the internal and proof environments.

5. English SAD: Features and Peculiarities of Implementation

The English SAD system satisfies well the requirements of the evidential paradigm and possesses a three-level structure intended for solving tasks of automated theorem proving and text verification [23].

The *first level* contains a parser module that analyzes an input ForTheL-text, its structure, and its logical content encoded in the ForTheL statements. After that, the input text is translated into its internal representation for further processing.

The result of this translation is a list of goal statements that should be deduced from their predecessors. Note that there is a module for parsing first-order logic formulas written in a variant of ForTheL and this module can be used whenever it is convenient.

At the *second level*, the goal statements generated during the first stage are sequentially processed by the foreground reasoner of SAD in order to reduce them to a number of subtasks for a prover. This is done either by splitting a goal into several simpler subgoals or by generating an alternative subgoal.

Proof search tasks are solved at the *third level* by a background prover, which can be either the native prover of English SAD or one of the external provers. The native prover is based on a special goal-driven sequent calculus for classical first-order logic with equality. Note that the calculus exploits the above-mentioned notion of an admissible substitution allowing to preserve the initial signature of a task under consideration so that accumulated equations can be sent to a specialized solver or an external computer algebra system. Also note

that English SAD was implemented in such a way that its external first-order prover can be Vampire [24], SPASS [25], Otter [26], E prover [27], or Prover9 [28].

At the final stage, SAD prints the result of its session on text verification or on automated theorem proving.

By now a number of non-trivial experiments has been carried out using the English SAD system. They are related to proof search in first-order logic, theorem proving in the context of a self-contained ForTheL-text, and verification of self-contained ForTheL-texts.

Some interesting examples on verification are:

- Ramsey's finite and infinite theorems,
- Cauchy-Bouniakowsky-Schwarz inequality from mathematical analysis,
- Chinese remainder theorem,
- Bezout's identity in terms of abstract rings,
- Some properties of finite groups,
- Tarski's fixed point theorem,
- Furstenberg's proof of the infinitude of primes.

Below, we use the ForTheL formalization of Furstenberg's proof of the infinitude of primes to illustrate some of the peculiarities of the text presentation in ForTheL and the process of verification in English SAD.

6. Example of Verification in English SAD

In 1955, Hillel (Harry) Furstenberg published an elegant twelve-line topological proof of the infinitude of prime numbers [29]. Here is a concise explanation of the argument:

Bi-infinite arithmetic sequences $\mathbf{A}(p,q) = p\mathbf{Z} + q$, where $p \neq 0$, form a basis for a topology on the set of integers $\mathbf{Z}$. In this topology, a sequence $\mathbf{A}(p,q)$ is both open and closed, as its complement is the union of other arithmetic sequences with the same difference p. However, the set of multiples of all primes $U = \mathrm{U}_{\text{p prime}}\, \mathbf{A}(p, 0)$ cannot be closed, since the only integers not in U are 1 and −1, and $\{-1, 1\}$ is not an open set. Consequently, there is an infinite amount of prime numbers.

To make this text suitable for formalization and verification we must provide a necessary mathematical background: basic properties of integers and sets, as well as the basic notions of topology. We are content with postulating these preliminaries as a set of definitions and axioms, without deriving them from a small set of first principles. Here is our list of ingredients:

- Algebraic properties of addition, subtraction, and multiplication of integers,
- Definitions of divisor and equality modulo in $\mathbf{Z}$,
- The fact that any integer other than 1 and −1 has a prime divisor,
- Elements of set theory: membership, subset relation, and finiteness,
- Operations on sets of integers: union, intersection, and infinitary union.

SAD has limited tools for dealing with higher-order properties, and so we make a number of simplifications to work around these limitations:

1. We describe the topology by defining open sets directly, without formalizing the notion of a topological basis.
2. We prove that the union of two closed sets is closed and then postulate that a finite union of closed sets is closed. While it is possible to prove the latter property in SAD by induction, we decided not to develop this part in the example for the sake of readability.

3. We only prove that the set S = {**A**(p,0) | p is prime} is infinite. The infinitude of primes follows from that and the fact that an image of a finite set is finite. While the latter property is easily stated in terms of sets and functions represented as first-order objects, SAD would not recognize S as an image of the set of all primes.

It remains to express the mathematical notation in the ASCII character set, and we obtain the self-contained ForTheL-text[1] given below. Note that we provide an excerpt without the preliminaries. In it, the names *a, b, c* are reserved for integers, *p, q* for non-zero integers, and *A, B* for subsets of **Z**. The set **Z** itself is denoted by INT. The set membership is denoted by <<, the not-equality relation by !=, the subset relation by [=, the set union by \-/, the set intersection by /-\. The operation ~ is the set complement in **Z** and "iff" is an abbreviation for "if and only if".

```
Definition ArSeq. ArSeq(p,a) = {b | b = a (mod p)}.
Definition  Open. A is open iff for any a << A there exists p such that ArSeq(p,a) [= A.
Definition Closed. A is closed iff ~A is open.
Lemma EmptyOpen. The empty set is open.
Lemma INTOpen. INT is open.
Lemma UnionOpen. Let S be a set such that all elements of S are open subsets of INT.
Then \-/ S is open.
Lemma InterOpen. Let A,B be  open  subsets  of  INT. Then A /-\ B is open.
Lemma  UnionClosed. Let A,B  be  closed  subsets  of  INT. Then A \-/ B is closed.
Axiom FUnionClosed. Let S be a finite set such that all elements of S are closed subsets of INT.
Then \-/ S is closed.
Lemma ArSeqClosed. ArSeq(p,a) is a closed subset of INT.
Indeed if b << ~ArSeq(p,a) and c = b (mod p) then c << ~ ArSeq(p,a).
Theorem Fuerstenberg.  Let S = {ArSeq(r, 0) | r is a prime}. Then S is infinite.
Proof.
    Let U =  \-/ S. We have ~U = {1, -1}.
    Indeed c belongs to U iff c has a prime divisor.
    {1, -1} is not an open set.
    Therefore U is not closed and S is infinite.
qed.
```

The ForTheL-text is more rigid and structured than the human-written proof above. Some natural expressions have to be reformulated to fit into the small fragment of English covered by ForTheL. Nevertheless, the meaning is still clear and a casual reader does not have to be an expert in ForTheL to understand it.

The first attempt to verify the proof fails at several steps, which turns out to be too difficult for the automated provers used by SAD. To make these steps easier ("more evident"), we have to provide the system with some guidance.

The simplest way to assist a prover is to prune the set of premises. ForTheL allows us to annotate an assertion with a list of relevant top-level facts (axioms, definitions, lemmas). After this SAD will try to prove the assertion after discarding (some of) the non-listed premises. In our example, lemma InterOpen can be proved automatically if rewritten as follows:

```
Lemma InterOpen. Let A,B be  open  subsets  of INT.
Then A /-\ B is open (by ZeroDiv,EquModMul).
```

where ZeroDiv and EquModMul are two properties about integer numbers from the preliminaries.

When a suitable list of relevant premises is difficult to find or is simply too long and too heavy to actually help the prover, the user can provide an intermediate lemma or write a proof for the problematic statement. We

[1]Available on the SAD website at http://nevidal.org/help-txt.en.html

already saw two short proofs (starting with the keyword "Indeed") attached to lemma ArSeqClosed and to the second sentence in the main proof. It turns out that lemma UnionClosed requires the same too:

```
Lemma UnionClosed. Let A,B be  closed subsets  of  INT. Then A \-/ B is closed.
Indeed ~A,~B  [= INT and ~(A \-/ B) = ~A  /-\ ~B.
```

This proof is quite trivial and it is somewhat disappointing that the automated prover did not find it on its own.

The last difficult step is the assertion that {−1, 1} is not open. By using the definition of the topology, we can show that {−1, 1} cannot hold any sequence $\mathbf{A}(p,1)$ for a non-zero p. Indeed, both $1 + p$ and $1 - p$ are in the sequence and they cannot be both equal to −1. In ForTheL, this can be written as follows:

```
{1, -1} is not an open set.
Proof.
    Assume the contrary.
    Take p such that ArSeq(p, 1) [= {1, -1}.
     1 + p and 1 - p belong to ArSeq(p, 1). We have 1 + p != 1 and 1 - p != 1.
    We have 1 + p != -1 or 1 - p != -1.
    Thus we have a contradiction.
end.
```

This at last makes the proof fully verifiable. In its final state, the ForTheL-text with Furstenberg's proof consists of 114 non-blank lines: 79 lines of preliminaries (before the definition of ArSeq) and 35 lines of the proof proper. The text contains 33 goals, including the lemmas in the preliminaries, and is fully verified by English SAD using the version v3.7 of the SPASS prover in about 3 minutes on an Intel Core system. By using splitting, definition expansion, and various strategies of the context pruning, the internal reasoning engine of English SAD generated 155 different proof obligations during the session. Of these, 15 were discharged without calling the prover, and 71 were successfully proved by SPASS.

7. Conclusion

We have shown that the evidential paradigm is well in line with the modern trends in the processing of formalized (not necessarily mathematical) texts and that the English SAD system looks fruitful from the point of view of implementing the existing principles of the construction of computer mathematical services that are able to carry out numerical calculations and/or symbolic transformations and/or deductive computations.

We envision the following directions of the future work on English SAD.

At the linguistic level: to create a Russian-language version of ForTheL and an automated translator between the two variants of ForTheL. As a result, this will expand the circle of potential users of the multilingual SAD system.

At the deductive level: using investigations on automated proof search in classical and non-classical sequent logics, to build tools that enable assembling of one or another SAD prover from separate modules depending on the subject domain under consideration and/or user desires.

At the heuristic level: to develop an interface for the interaction of English SAD with known solvers and computer algebra systems to provide the English SAD system with remote tools for solving separate equations and equation systems during processing of an input ForTheL-text.

We hope that ideas and approaches of the evidential paradigm presented here will serve as a foundation for an infrastructure for the remote presentation and complex processing of mathematical knowledge and will be useful in both academical and teaching activities.

References

[1] Glushkov V.M. Some problems in automata theory and artificial intelligence. Cybernetics and System Analysis. 1970:6(2):P. 17-27.

[2] Lyaletski A., Morokhovets M., Paskevich A. Kyiv School of Automated Theorem Proving: a Historical Chronicle. In book: "Logic in Central and Eastern Europe: History, Science, and Discourse". USA:University Press of America; 2012:431-469.

[3] Letichevsky A.A., Lyaletski A.V., Morokhovets M.K. Glushkov's Evidence Algorithm. Cybernetics and System Analysis. 2013:49(4):3-16.

[4] Lyaletski A., Morokhovets M. Evidential paradigm: a current state. Programme of the International Conference "Mathematical Challenges of the 21st Century". USA: University of California, Los Angeles. 2000:48.

[5] Vershinin K., Paskevich A. ForTheL – the language of formal theories. International Journal of Information Theories and Applications. 2000:7(3):120-126.

[6] Kapitonova Yu.V., Vershinin K.P., Degtyarev A.I., Zhezherun A.P., Lyaletski A.V. System for processing mathematical texts. Cybernetics and System Analysis. 1979:15(2):209-210.

[7] Armando A., Bertoli P., Coglio A., Giunchiglia F., Meseguer J., Ranise S., Talcott C. Open Mechanized Reasoning Systems: A preliminary report. Proceedings of the Workshop on Integration of Deduction Systems (CADE 15). 1998.

[8] OpenMath.org: https://www.openmath.org

[9] The QED Manifesto: https://www.cs.ru.nl/~freek/qed/qed.html

[10] Mizar Home Page: http://mizar.uwb.edu.pl

[11] The Theorema system: https://www.risc.jku.at/research/theorema/software/

[12] Anufriyev F.V. An algorithm of theorem proving in logical calculi. , GIC, AS of UkrSSR, Kiev: Theory of Automata. 1969:3-26. (In Russian).

[13] Lyaletski A.V. On variant of Herbrand theorem for formulas in prefix form, Cybernetics and System Analysis (Historical Archive). 1981:17(1):125-129.

[14] Lyaletski A. Gentzen calculi and admissible substitutions. Actes Preliminaires du Symposium Franco-Sovietique "Informatika-91", Grenoble, France: 1991:99- 111.

[15] Lyaletski A., Verchinine K., Degtyarev A., Paskevich A. System for Automated Deduction (SAD): Linguistic and deductive peculiarities. Advances in Soft Computing: Intelligent Information Systems, Proceedings of the 11th International Symposium, Sopot, Poland. 2002:413–422.

[16] 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 System Analysis. 1972:8(5):730-739.

[17] Glushkov V.M., Vershinin K.P., Kapitonova Yu.V., Letichevsky A.A., Malevanyi N.P., Kostyrko V.F. On a formal language for representation of mathematical texts. GIC, AS of UkrSSR, Kiev: Automated Proof Search in Mathematics. 1974:3-36. (In Russian).

[18] Degtyarev A.I., Lyaletski A.V. Deductive tools of the Evidence Algorithm. Abstracts of the All-union Conference "Methods of Mathematical Logic in Artificial Intelligence and Systematic Programming", Palanga, Lithuania. 1980: I: 89-90. (In Russian).

[19] Lyaletski A.V., Degtyarev A.I. Logical inference in System for Automated Deduction, SAD. GIC, AS of UkrSSR, Kiev: Mathematical Foundations of Artificial Intelligence Systems. 1981: 3-11. (In Russian).

[20] Degtyarev A., Lyaletski A., Morokhovets M. Evidence Algorithm and sequent logical inference search. Lecture Notes in Computer Science. 1999:1705:44-61.

[21] Konev B., Lyaletski A. Tableau proof search with admissible substitutions. Proceedings of the International Workshop on First-order Theorem Proving, Koblenz, Germany. 2005:35-50.

[22] Lyaletski A. Mathematical text processing in EA-style: a sequent aspect. Journal of Formalized Reasoning

(Special Issue: Twenty Years of the QED Manifesto). 2016:9(1):235-264.
[23] Verchinine K., Lyaletski A., Paskevich A. System for Automated Deduction (SAD): a tool for proof verification. Lecture Notes in Computer Science. 2007:4603:398-403.
[24] Vampire's Home Page: http://www.vprover.org
[25] Classic SPASS Theorem Prover: https://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/classic-spass-theorem-prover/
[26] Otter and Mace2: https://www.cs.unm.edu/ mccune/otter/
[27] The E Theorem prover: http://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html
[28] Prover9 and Mace4: https://www.cs.unm.edu/~mccune/mace4/
[29] Furstenberg H. On the infinitude of primes. American Mathematical Monthly. 1955:62(5): 353.

Authors' Profiles

Alexander Lyaletski (born 1947) is a research associate of the Institute of mathematics and computer science, Chisinau, Republic of Moldova. The areas of main interests: automated reasoning, proof theory, symbolic and computational logics.

Alexandre Lyaletsky (born 1982) is a researcher at the Faculty of Mechanics and Mathematics, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine. The areas of main interests: the untyped lambda-calculus, classical first-order logic, general algebra, group theory.

Andrei Paskevich (born 1979) is an associate professor at the Paris-Sud University, Orsay, France. The areas of main interests: formal verification and automated reasoning.

How to cite this paper: Alexander Lyaletski, Alexandre Lyaletsky, Andrei Paskevich,"Evidential Paradigm and SAD Systems: Features and Peculiarities", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.4, No.2, pp.1-11, 2018.DOI: 10.5815/ijmsc.2018.02.01