Evidential Paradigm as Formal Knowledge Presentation and Processing

Alexander Lyaletski, Alexandre Lyaletsky, and Andrei Paskevich

Taras Shevchenko National University of Kyiv, Ukraine
Paris-Sud University, Orsay, France
forlav@mail.ru foraal@mail.ru andrei@lri.fr

Abstract. Investigations on the design and creation of high-performance information systems based on a number of paradigms supporting a human activity in formalized text processing were started in the beginning of 1960s – the time of the appearance of computers of such a high performance that programming of complex intelligent processes became possible. The so-called evidential paradigm is among them and it can be considered as a certain way of the integration of all reasonable paradigms intended for the development of languages for presenting formalized text in the form most appropriate for a user, formalization and evolutional development of a computer-made proof step, information environment having an influence on a current evidence of a computer-made proof step, and interactive man-assistant search of a proof. This paper contains a short description of the evidential paradigm and its implementation in the form of systems for presenting and processing formal knowledge.

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

Key Terms. MachineIntelligence

1 Introduction

The term "artificial intelligence" cannot be reduced only to the creation of devices or their components that completely or partially simulate the physical activity of humans; it also touches questions relating to the ability of a human to do reasoning in the framework of formalized languages. Investigations in this direction as well as in the direction of design and creation of high-performance intelligent information systems, now called automated reasoning systems and computer algebra systems started in the beginning of 1960s – the time of the appearance of computing machines of such a high speed, information capacity, and flexibility, that the programming of complex intelligent processes became possible. As a result, several different paradigms of the intelligent processing of computer formalized knowledge have appeared.

ICTERI 2016, Kyiv, Ukraine, June 21-24, 2016

Efficient processing of formalized knowledge presupposes carrying out deep investigations in the fields of automated reasoning and construction of linguistic tools intended for supporting daily mental activity of a human, which presupposes an auspicious combination of theory and practice.

Attempts to find a reasonable balance between theory and practice have lead to creation and development of a number of certain paradigms reflected in intelligent systems such as automated theorem-proving and computer algebra systems.

In Ukraine, such investigations were initiated by Academician V.M.Glushkov in the end of 1960s, who announced the Evidence Algorithm (EA) programme in [1][1] intended for making research on automated theorem proving and symbolic computations simultaneously in languages for presenting formalized texts in the form most appropriate for a user, formalization and evolutionary development of a computer-made proof step, information environment having an influence on a current evidence of a computer-made proof step, and interactive man-assistant search of a proof. In [4], the modern vision of the EA programme was called the *evidential paradigm*.

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 *SAD* system (System for Automated Deduction) intended for the presentation and processing of (formalized) mathematical texts in English and accessible at http://nevidal.org/. This paper is devoted to a short description of the peculiarities and features of the evidential paradigm and SAD.

2 Some remarks on formalized knowledge processing

Let us consider some of the issues, with which computer science is constantly dealing in solving problems associated with the formalized knowledge processing based on the following paradigms: numerical, analytical (symbolic), deductive, inductive and integrative.

The *numerical paradigm* reflects tools and methods for finding approximate or exact solutions of problems of pure or applied mathematics. It is based on constructing finite sequences of numerical calculations and actions over finite sets of numbers.

The *symbolic (analytical) paradigm* is based on the ability of computers to perform complex symbolic transformations, do numerical calculations, plot functions, construct mathematical models of certain processes, and so on. It is focused on the construction and use of computer algebra systems and appeared as an alternative to the numerical paradigm in the middle of 1960s, when computers with high enough performance were constructed.

The *deductive paradigm* reflects the declarative way of the representation and processing of computer knowledge. It is based on the fact that existing knowledge is represented in the form of certain formal texts or their pieces (usually axioms,

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

definitions, propositions, theorems, etc.) and that an additional knowledge is produced by means of applying certain inference (reasoning) rules for deducing new facts. The systems for the representation and processing of knowledge based on this paradigm are called automated reasoning systems, most of which exists in the form of automated theorem-proving systems, as the deductive approach is most relevant and efficient for tasks requiring reasonings "from general to particular" used in many application areas.

The *inductive paradigm* is based: in the natural sciences – on the transition from individual observations to general conclusions, while in mathematics – on the induction method used for the "checking" of a selected property for partially or fully ordered objects.

The *integration paradigm* "brings together" all of the above-mentioned paradigms. It can be divided into two types:

– *integration at the design phase*, when the ability of hierarchical building-in new components and subsystems into a designed system and connecting it to available computer services is provided during the design time of the system and

– *integration at the operation stage*, i.e. combining existing computer services into one (new) system (a special interest to the development of such services has been caused by a wide use of the Internet for the date exchange and transition of a necessary information, which, in its turn, led to the creation of relevant data exchange standards).

As to the integration at the operation stage, there can be mentioned Open Mechanized Reasoning Systems [5] and OpenMath [6] projects, in the framework of which certain specification and communication languages for solving tasks of theorem proving and symbolic computation were constructed.

The most famous representatives of the approach to the integration at the design stage are the QED [7] project as well as the Mizar [8] and Theorema [9] systems.

As for 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 by the evidential paradigm.

3 Peculiarities of evidential paradigm

The evidential paradigm is developed for mainly solving such problems of formalized text processing as proving a theorem under consideration or verifying a given proof of a theorem.

According to the evidential paradigm, the scheme of the processing of formalized (non necessary mathematical) texts is as follows (a more detailed description can be found in [2]):

Text prepared by a user for proving/verifying a theorem ==========> (using a parser)
A first-order self-contained text ==========> *A computer–made proof/verification* ==========> *Text in a form comprehensible for a human*
(using a prover) (using an editor)

Thus, the following problems arise in the framework of the evidential paradigm: creation of formal mathematical languages, construction of deductive methods and tools, creation and usage of information environment, and development of interactive modes.

Let us briefly describe each of them and some of the ways used for solving them. Naturally, the main attention will be paid to the linguistic and deductive tools satisfying the evidential paradigm requirements. In this connection, the other tools are only slightly contoured.

3.1 Linguistic tools

The following requirements were formulated for a language that should be constructed in the framework of the evidential paradigm.

Its syntax and semantics should be formalized. It should allow writing the axioms of a theory under consideration as well as auxiliary propositions, lemmas, theorems, and their proofs to ensure the self-containedness of a text and wording of definitions. The language thesaurus should be extensible and separated from the language grammar. In addition, it should be close to the languages of mathematical publications in order to provide convenience and comfort to a user in creating and processing a text in interactive mode. Besides, it should give the possibility to write formulas of the first-order language. Moreover, it should allow to formulate tasks that are not directly related to the deduction search.

The first sketch of such a language appeared in 1970 [10]. Its final (Russian) version called TL (Theory of Language) was published in [11].

In 2000, a new, improved, English-language version of TL called ForTheL (FORmal THEory Language) was constructed [12]. The main objective of the construction of ForTheL (and TL) was to provide an initial (mathematical) environment for solving a task under consideration as well as for the further development of evidential (logical) engines and strengthening of their capabilities.

Like any 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. Sentences are built of units: 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. “subset of”) or terms (“closure of”), verbs and adjectives which form predicates (“belongs to”, “compact”), symbolic primitives that use a concise symbolic notation for predicates and functions and allow to consider usual quantifier-free first-order formulas as ForTheL statements. Of course, just a little fragment of English is formalized in the syntax of ForTheL.

There are three kinds of sentences in the ForTheL language: assumptions, selections, and affirmations. Assumptions serve to declare variables or to provide some hypotheses for the subsequent text. For example, the following sentences are typical assumptions: “Let S be a finite set.”, “Assume that m is greater than n.”. Selections state the existence of representatives of notions and can be used to declare variables, too. Here follows an example of a selection: “Take an even prime number X.”. Finally , affirmations are simply statements: “If p

divides $n - p$ then p divides n". The semantics of a sentence is determined by a series of transformations converting a ForTheL statement to the corresponding first-order formula for processing it by the deductive tools of SAD.

ForTheL sections are: sentences, sentences with proofs, cases, and top-level sections: axioms, definitions, signature extensions, 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.

3.2 Deductive tools

From the very beginning of its appearance, EA has paid great attention to developing machine proof search methods suitable for the various fields of mathematics and taking into account (informal) human reasoning techniques.

Deductive tools should satisfy the following requirements: (1) the syntactical form of an original task under consideration should be preserved; (2) deduction should be carried out within the signature of an initial theory (i.e. without applying skolemization); (3) proof search should be goal-driven; (4) equality (equations) handling should be separated from a deductive processes.

As a result, the sequent approach was selected as basic for the construction of logical engines in the framework of the evidential paradigm. This permitted to satisfy (1) in a good enough form. Besides:

– for satisfying (2), a technique for the optimization of quantifier handling was proposed on the basis of an original notion of an admissible substitution,

– for satisfying (3), proof search was proposed in the form "driving" the process of an auxiliary goal generation by taking a formula (goal) under consideration into account,

– for satisfying (4), special methods for equality processing and equation solving were developed (allowing the use of algebra systems and problem solvers if necessary).

Investigations in this direction began in 1963, when the problem of automated theorem proving in group theory was formulated by V. Glushkov. As a result, the procedure admitted its interpretation as a specific, sound and complete, sequent calculus later called the AGS (Auxiliary Goals Search) calculus was proposed in [13]. It used Kanger's approach to quantifier handling and, in this connection, it was less efficient that the usual resolution-type methods.

Attempts to overcome this disadvantage of AGS led to the construction of an original sequent calculus [14, 15] on the basis of a new notion of an admissible substitution distinguished from that was used before its appearance. It was implemented in the (Russian) SAD system and its implementation demonstrated the usefulness of deductive approach to constructing such calculi.

Since then, these investigations were stopped until 1998, when the paper's authors started participating in the Intas project 96-0760 "Rewriting Techniques and Efficient Theorem Proving" (1998-2000). This project gave an impulse for modifying the calculus from [15] in several directions, one of which concerns the case of classical logic (see, for example, [16]) and the others – non-classical ones. As a result, a special research was conducted and there was obtained a wide

spectrum of interesting results on efficient enough inference search in classical and intuitionistic logics as well as in their modal extensions. Note that this research for classical logic was used in implementing the logical engine of the English SAD system.

Note that the (new) notion of admissibility is not enough for the construction of sound calculi in the intuitionistic case. This situation can be corrected by using the notion of compatibility firstly used in [17] for the construction of the sound (and complete) tableau calculus with free variables for intuitionistic logic and introduced in a number of calculi for modal extensions of classical and intuitionistic logics [18].

3.3 Information environment and interactive modes

We distinguish *three types* of information environment. The first is a *proof environment* intended for keeping all the data necessary for solving a task under consideration. By now, it presents a certain self-contained ForTheL-text prepared with the help of a user. All mathematical facts accumulated during previous sessions and needed for solving a task form the second type of environment called *internal environment*. The whole information that can be received via Internet by means of using mathematical services existent in Internet give the third type, the so-called *external environment*.

Interactive modes can serve for interfacing a computer assistant (in our case – the SAD system) with both a user and computer mathematical services existent in the external information environment. They can be designed and implemented only after fixing the form of internal and proof environments.

4 SAD system

The earlier-mentioned English SAD system satisfies well enough the requirements of the evidential paradigm and now it can be considered as its three-level implementation that is intended for theorem proving and text verification [19].

At the *first level*, its parser module analyzes an input mathematical text, its structure, and its logical content, encoded in the ForTheL statements. After this, the translation of the text into its internal presentation is made.

The result of translation gives a list of goal statements to be deduced from their predecessors. Note that there exists a module for processing first-order formulas presenting in a "dialect" of ForTheL, which can also be used if convenient.

At the *second level*, the goal statements are generated one-by-one and subsequently processed by the so-called foreground reasoner of SAD for reducing them to a number of subtasks for a prover splitting a goal under consideration to several simpler subgoals or generating an alternative subgoal. The module becomes redundant when SAD solves a problem for automated theorem proving.

Proof search tasks are resolved by a background prover at the *third level*. The SAD native prover is based on a special goal-driven sequent calculus for the classical first-order logic with equality. Note that it exploits the above-mentioned

notion of admissible substitution, which permits to preserve the initial signature of a task so that special accumulated equations can be sent to a specialized solver, e.g. an external computer algebra system (CAS). Additionally note that the SAD system was implemented in such a way that it can use some other external first-order provers, such as Otter [20], SPASS [21], or Vampire [22].

As a final step, the SAD outputs the result of its session.

Finally, note that by now, a number of experiments has been made in the SAD. They are related to: inference search in first-order sequent logic (a user may give his own problems or refer to a problem from the known TPTP problem library), theorem proving in the context of a self-contained ForTheL-text, and verification of self-contained ForTheL-texts.

The most interesting examples on verification are:

- Ramsey's finite and infinite theorems;
- some properties of finite groups;
- theorem that the square root of a prime number is irrational;
- Cauchy-Bouniakowsky-Schwarz inequality from mathematical analysis;
- Chinese remainder theorem and Bezout's identity in terms of abstract rings;
- Tarski's fixed point theorem;
- theorem on the stability of the refinement relation over a set of program specification.

5 Conclusion

The above-given material shows that the evidential paradigm is well in line with the modern trends in the processing of formalized (not obligatory mathematical) texts and that the SAD system looks fruitful from the point of view of implementing the existing principles of the construction of computer mathematical services being defined as information systems that are able to carry out numerical calculations and/or symbolic transformation and/or deductive constructions. The paper's authors hope that the ideas presented in the paper and used in the SAD system, will attract the attention of researchers in the field of artificial intelligence and will be useful in developing computer services intended for formal knowledge presentation and processing.

References

1. V. M. Glushkov. Some problems in automata theory and artificial intelligence. Cybernetics and System Analysis, Vol. 6, No. 2, Springer, New York, 1970, P. 17-27.
2. A. Lyaletski, M. Morokhovets, and A. Paskevich. Kyiv School of Automated Theorem Proving: a Historical Chronicle. In book: *"Logic in Central and Eastern Europe: History, Science, and Discourse"*, University Press of America, USA, P. 431-469, 2012.
3. A. A. Letichevsky, A. V. Lyaletski, and M. K. Morokhovets. Glushkov's Evidence Algorithm. *Cybernetics and System Analysis*, Vol. 49, No. 4, Springer, New York, P. 3-16, 2013.

4. A. Lyaletski and M. Morokhovets. Evidential paradigm: a current state. *Programme of the International Conference "Mathematical Challenges of the 21st Century"*. University of California, Los Angeles, USA, P. 48, 2000.
5. A. Armando, P. Bertoli, A. Coglio, F. Giunchiglia, J. Meseguer, S. Ranise, C. Talcott. Open Mechanized Reasoning Systems: A preliminary report. *Proceedings of the Workshop on Integration of Deduction Systems (CADE 15)*, 1998.
6. OpenMath.org: http://www.openmath.org/
7. The QED Manifesto: http://www.cs.ru.nl/ freek/qed/qed.html
8. Mizar Home Page: http://mizar.uwb.edu.pl/
9. Theorema: https://www.risc.jku.at/research/theorema/
10. V. M. Glushkov, Yu. V. Kapitonova, A. A. Letichevskii, K. P. Vershinin, and N. P. Malevanyi. Construction of a practical formal language for mathematical theories. *Cybernetics and System Analysis*, Vol. 8, No. 5, Springer, 1972, P. 730-739.
11. V. M. Glushkov, K. P. Vershinin, Yu. V. Kapitonova, A. A. Letichevsky, N. P. Malevanyi, and V. F. Kostyrko. On a formal language for representation of mathematical texts. *Auitomated Proof Search in Mathematics*, GIC, AS of UkrSSR, Kiev, 1974, P. 3-36 (in Russian).
12. K. Vershinin and A. Paskevich. ForTheL – the language of formal theories. *International Journal of Information Theories and Applications*, 7(3), 2000, P. 120-126.
13. F. V. Anufriyev. Analgorithm of theorem proving in logical calculi. *Theory of Automata*, GIC, AS of UkrSSR, Kiev, 1969, P. 3-26 (in Russian).
14. A. I. Degtyarev and A. V. Lyaletski. Deductive tools of the Evidence Algorithm. *Abstracts of the All-union Conference "Methods of Mathematical Logic in Artificial Intelligence and Systematic Programming"*, Vol. I, Palanga, Lithuania, P. 89-90, 1980 (in Russian).
15. A. I. Degtyarev and A. V. Lyaletski. Logical inference in the system for automated deduction, SAD. *Mathematical Foundations of Artificial Intelligence Systems*, GIC, AS of UkrSSR, Kiev, P. 3-11, 1981 (in Russian).
16. A. Degtyarev, A. Lyaletski, and M. Morokhovets. Evidence Algorithm and sequent logical inference search. *Lecture Notes in Computer Science*, 1705, Springer-Verlag, P. 44-61, 1999.
17. B. Konev and A. Lyaletski. Tableau proof search with admissible substitutions. *Proceedings of the International Workshop on First-order Theorem Proving*, Koblenz, Germany, P. 35-50, 2005.
18. A. Lyaletski. Mathematical Text Processing in EA-style: a Sequent Aspect. *Journal of Formalized Reeasoning (Special Issue: Twenty Years of the QED Manifesto)*, vol. 9, No. 1, P. 235-264, 2016.
19. K. Verchinine, A. Lyaletski, and A. Paskevich. System for Automated Deduction (SAD): a tool for proof verification. *Lecture Notes in Computer Science*, 4603, Springer, P. 398-403, 2007.
20. Otter prover: https://www.cs.unm.edu/ mccune/otter/
21. SPASS Homepage: http://www.spass-prover.org/
22. Vampire's Home Page: http://www.vprover.org/