Evidential Paradigm and Intelligent Mathematical Text Processing

Alexander Lyaletski[1], Anatoly Doroshenko[2],
Andrei Paskevich[1,3], and Konstantin Verchinine[3]

[1]Taras Shevchenko Kiev National University, Kiev, Ukraine
[2]Institute of Software Systems of NASU, Kiev, Ukraine
[3]Université Paris XII — Val de Marne, Creteil, France
lav@unicyb.kiev.ua dor@isofts.kiev.ua

Abstract: This paper presents the evidential paradigm of computer-supported mathematical assistance in "doing" mathematics and in reasoning activity. At present, the evidential paradigm is implemented in the form of System for Automated Deduction (SAD). The system is based on the methods of automated theorem proving and is intended for intelligent mathematical text processing. It proves mathematical theorems, verifies validity of self-contained mathematical texts and can be used for inference search in first-order sequent-based logic as well. For human-like representation of mathematical knowledge, SAD exploits an original formal language close to natural languages of scientific publications. Since the problem of automated text verification is of great importance for industrial applications (checking specifications, proving safety properties of network protocols, etc), the paper illustrates some principles and peculiarities of the evidential paradigm by means of exemplifying the verification of a part of a non-trivial mathematical text.

1 Introduction

This paper is devoted to modern vision (called evidential paradigm) of the Evidence Algorithm programme advanced by Academician V. Glushkov as investigations on mechanization of mathematical activity in a broad sense [1]. The main objective of the programme is to provide a mathematician with tools for proof search and verification of the validity of formal texts (theorems) by using the evolutionary developing notion of "evident" (from the point of view of a computer) reasoning step.

The evidential paradigm is intended for simultaneous exploring formalized languages for presenting mathematical texts in the form most appropriate for a user, formalization and evolutionary development of computer-made proof step, information environment having an influence on the evidence of a proof step, and man-assisted search for a proof. It is oriented to integration of deduction and symbolic calculations (cf. [2, 3]).

As the result of the realization of the evidential paradigm, the System for Automated Deduction, SAD, has appeared [4, 5, 6]. Now SAD can be used online via Internet (see: "http://ea.unicyb.kiev.ua") for solving the following problems: (i) automatic and automated sequent inference search in first-order classical logic; (ii) automated theorem proving in the framework of a self-contained text put down in a special human-like formal language; (iii) verification of a self-contained text written down in the special language.

In the case of (i), SAD get a first-order sequent investigated and try to establish its deducibility with the help of a special inference search technique. (In particular, SAD has remote access to the well-known TPTP Problem Library for receiving such tasks.)

When solving problems (ii) and (iii), the following transformations are performed:

Firs of all, a mathematical text under consideration is formalized by a user with the help of the ForTheL language [7], which is formal, on the one hand, and is close to natural languages of mathematical publications, on the other hand.

After this, an initial ForTheL-text is exposed to deductive processing in accordance with a problem solved. In the case of proving the last proposition of a ForTheL-text under consideration, the ForTheL-text is translated in its special representation – ForTheL1-text, and then a proof of the proposition is search for. In the case of verifying a ForTheL-text, a number of propositions to be proven are generated, and some attempts are made to prove every of these propositions.

Note that, irrespective of a considered case, deduction in SAD is based on an original sequent formalism [8, 9, 10]. The sequent formalism was chosen since sequent calculi reflect better the "natural" way of reasoning than resolution-tipe ones. In particular, neither skolemization nor clause-transformations are required, and inference search is made in the signature of an initial theory. The high efficiency of proof search is gained due to the goal-driven rules applications and to a special quantifier handling technique.

In this paper, we focus our attention on the problem of text verification, as it caused essential extensions of all the possibilities of the first implementation of SAD (cf.[5, 6]).

2 Architecture of SAD

The architecture of the current version of SAD is shown below. During any verification session, SAD solves three tasks: (a) translating an input ForTheL-text to its internal representation; (b) determining a sequence of affirmations to be verified (proved); (c) searching for logical inference of every goal using its logical predecessors in the ForTheL-text. There exist four main modules in SAD: [ForTheL], [FOL], [Reason], and [Moses].

Modules [ForTheL] and [FOL]. The modules [ForTheL] and [FOL] perform parsing of ForTheL-texts and first-order texts, respectively. Each of these modules converts its input text to a corresponding internal representation. This conversion preserves the structure of an initial text and translates phrases to first-order formulas. Internal representation of an input text constructed by [ForTheL] or [FOL] serves as an information environment for subsequent actions.

Module [Reason]. This module runs a verification cycle, formulates verification tasks, and tries to solve them with the help of both its own reasoning capabilities and the prover

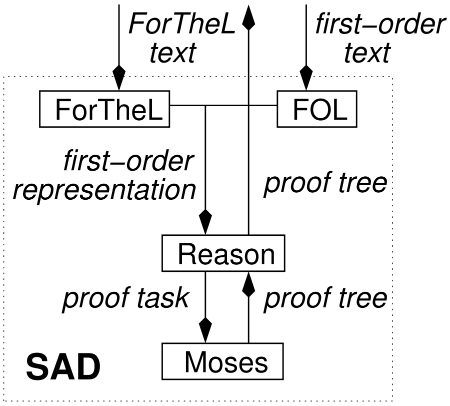

Figure 1: Architecture of SAD

[Moses] of SAD. In particular, [Reason] uses analyzing cases and proving by induction, simplifies a goal using assumptions and affirmations occurring in the text, splits a complex goal to a number of simpler subgoals, and so on. Below, we demonstrate the work of [Reason] on a fragment of a non-trivial mathematical text.

Module [Moses]**.** This module is intended for logical inference search. It is implemented on the base of the calculus GD [4]. In order to provide SAD with equality handling capabilities, a certain modification of Brand's method [11] is implemented. In addition, the paramodulation rule is planed to be built-in into [Moses] on the basis of [12].

Due to the absence of preliminary skolemization, the prover of SAD is capable to use a relevant solver for solving equation systems. The submodule of [Moses] responsible for equation handling acts as a mediator between the prover and an internal or external solver This submodule checks a substitution constructed by a solver for admissibility and generates additional equations if necessary. The procedure computing the most general unifier is used as a default equation solver.

3 Language of SAD

The language ForTheL (Formal Theory Language) [7] is a language with formally defined syntax and semantics. It is intended for representation of mathematical texts consisting of axioms, definitions, theorems and proofs.

A ForTheL-text is a sequence of sections, phrases, and special constructs like pattern introductors. Phrases are either assumptions (then they begin with "`let`" or "`assume`") or affirmations. Sections may be composed from sections of a lower level and phrases. Typical top-level sections are axioms, definitions, and propositions. Typical sections of lower level are proofs and proof cases. Pattern introductors serve to extend the thesaurus of an input text.

The grammar of ForTheL-phrases simulates the grammar of English sentences. Phrases are constructed with the help of nouns (which denote notions (classes) or functions), verbs, and adjectives (which denote predicates), as well as prepositions and conjunctions defining the logical meaning of a complex sentence. Here is a simple ForTheL-affirmation: "`Every closed subset of every compact set is compact.`"

In this example, notions "`set`" and "`subset of _`" and adjective predicates "`closed`" and "`compact`" have to be introduced with pattern introductors. The only predefined ForTheL-pattern is a predicate pattern "`_ is equal to _`".

Every affirmation in a text, e.g. the affirmation of a theorem, can be provided with a proof section. A ForTheL-proof is a sequence of assumptions, affirmations (which may have their own proofs), and proof cases if the proof is held by case analysis. A proof section is considered as a logical predecessor of an affirmation being proved, that is, SAD verifies the proof first and then checks that the affirmation is implied by that proof.

Besides providing a proof section, the verification process can be supported with *references*. After an affirmation, one can list the sections (by their labels) to give them a higher priority during the search for a proof of this affirmation.

4 Text verification in SAD

This section is devoted to the verification procedure of SAD and to the peculiarities of formalization style. We consider a real mathematical problem: Infinite Ramsey's theorem as it is presented in the beginning of Graham's introductory book [13]:

Infinite Ramsey's Theorem. *For all $k, r \in \omega$ and any r-coloring $\xi : \left[{\omega \atop k}\right] \rightarrow [r]$ of the k-element subsets of ω, there is always an infinite subset $S \subseteq \omega$ with all its k-element subsets having the same color.*

This proposition, together with Finite Ramsey's Theorem and Compactness Principle, was formalized and automatically verified in SAD. The whole ForTheL-text consists of 490 lines, 200 of which are used for preliminary facts: general notions of set and number theory, definitions and properties of functions and predicates. The rest 290 lines contain mainly the formalization of three proofs. Note that the proofs of Infinite Ramsey's Theorem and of Compactness Principle (the proof of Finite Ramsey's Theorem is not given) take approximately 130 lines in Graham's book. So, we can consider ForTheL (and the whole system SAD) as a rather economical tool for formalization of texts.

The given-above proposition is rewritten in ForTheL as follows (we replace some ASCII-notation with more readable mathematical characters of $\TeX$).

Theorem RamseyInf. *Let T be a finite set. For all $(k \in \omega)$ and all countable $(S \subseteq \omega)$ for every $(c : [S/k] \rightarrow T)$ there exists an element u of T and a countable $X \subseteq S$ such that for every $(Q \in [X/k])$ $c(Q) = u$.*

We don't give here the whole ForTheL-text. Instead, we will consider a fragment of the proof that illustrates well the peculiarities of text verification in SAD.

The Infinite Ramsey's Theorem is proved by induction on k. In the proof of the induction step, we construct a number of objects (sequences, functions, and sets) and prove their properties. The most important construction in our proof is $\{N_i\}_\omega$, a recursively defined sequence of subsets of ω. This sequence is decreasing, that is $N_{i+1} \subseteq \mathrm{cdr}\ N_i = N_i \backslash \{\min N_i\}$. Below we verify a proof of a simple conjecture of this fact by using SAD:

```
For every ($i, j \in \omega$) if $j \leqslant i$ then $N(i) \subseteq N(j)$.
Proof by induction.
    Let $i, j$ be numbers so that $j \leqslant i$.
    Let $I$ be a number so that $i = \mathrm{succ}\ I$.
    Case $i \leqslant j$. Obvious.
    Case $j \leqslant I$.
        Then $N(I) \subseteq N(j)$ (by IH).
        $\mathrm{cdr}\ (N(I)) \subseteq N(I)$ (by DefDiff).
        $N(i) \subseteq N(j)$ (by SubTrans).
    end.
end.
```

The main affirmation is proved by natural induction. The systems assumes by default that the induction is held by the topmost universally quantified variable in the goal, that is, by i. Also, you can mention the induction term explicitly ("`proof by induction on i+j.`"). The verificator ([Reason]) checks that the first assumption in the proof section introduces the needed variables and corresponds to the conditions in the goal. Then an appropriate induction hypothesis IH(i) is formulated as follows:

$$\forall\, x\ , y \in \omega\ (x < i \supset (y \leqslant x \supset N(x) \subseteq N(y)))$$

Then this hypothesis is added to the logical context of the rest of the proof.

In our proof, the base case ($i = 0$) is obvious, since $N(0) = \omega$ by definition. In this connection, it is omited, and SAD considers the induction step. SAD assumes the existence of the predecessor of i and proves the current goal $N(i) \subseteq N(j)$ by case analysis. Note that the goal was simplified in accordance with the first assumption.

When proof cases are considered, SAD checks that the case analysis is complete, i.e. the corresponding disjunction is valid. In our proof, this disjunction is $\mathrm{succ}\ x \leqslant y \vee y \leqslant x$, and it is easily verified. Then SAD tries to prove the current goal for each case respectively. Note the reference to the induction hypothesis made in the second case section.

5 Conclusion

Necessity for efficient and convenient systems of automated reasoning goes out of the frames of purely scientific applications. It is obvious, in particular, that safety and reliability properties of mission-critical software and hardware should be strictly proved rather than established empirically. These property, while coming from various fields, finally is nothing without precise mathematical statements expressing it in the terms of some specific mathematical theory. Languages of mathematics and of mathematical reasoning provide a universal and powerful basis to formalize such kinds of problems and to search for their solutions. So, the evidential paradigm can be useful for creating and processing, in a convenient and comfortable way, formal texts containing non-trivial mathematical statements and proofs.

The approach suggested can be helpful in attacking such problems as automated theorem proving, verification of mathematical papers, remote training in mathematics, construction of knowledge bases for formal theories, and integration of symbolic calculation with deduction. Also, it can be adapted to solve logical problems of decision making theory, to verify the formal specifications of both software and hardware, and so on.

References

[1] V. Glushkov. Some Problems of Automata Theory and Artificial Intelligence (in Russian). In: Kibernetika, No 2, 1970, 3–13.

[2] Computer Algebra Pages and Servers. http://krum.rz.uni-mannheim.de/cabench/cawww.html

[3] Mechanized Reasoning. http://www-formal.stanford.edu/clt/ARS/ars-db.html

[4] K. Verchinine, A. Degtyarev, A. Lyaletski, and A. Paskevich. SAD, a System for Automated Deduction: a Current State. In: Proceedings of the Workshop on 35 Years of Automating Mathematics, Heriot-Watt University, Edinburgh, Scotland, 10–13 April, 2002, 12 pp.

[5] Z. Aselderov, K. Verchinine, A. Degtyarev, A. Lyaletski, A. Paskevich, and A.Pavlov. Linguistic Tools and Deductive Technique of the System for Automated Deduction. In: Proceedings of of the 3rd International Workshop on the Implementation of Logics, Tbilisi, Georgia, October 14–18, 2002, 21–24.

[6] Z. Aselderov, K. Verchinine, A. Degtyarev, A. Lyaletski, and A. Paskevich. Peculiarities of Mathematical Texts Processing in the System for Automated Deduction, SAD (in Russian). In: Artificial Intelligence (Proc. of the 3rd International Conference ”Artificial Intelligence”, Katsiveli, Ukraine), 2002, No 4, 164–171.

[7] K. Vershinin, A. Paskevich. ForTheL — the Language of Formal Theories. In: IJ Information Theories and Applications, v. 7–3, 2000, 121–127.

[8] A. Degtyarev, A. Lyaletski, M. Morokhovets. Evidence Algorithm and Sequent Logical Inference Search. In: Lecture Notes in Artificial Intelligence, v. 1705, 1999, 44–61.

[9] A. Degtyarev, A. Lyaletski, and M. Morokhovets. On the EA-Style Integrated Processing of Self-Contained Mathematical Texts. In: Symbolic Computation and Automated Reasoning (the book devoted to the CALCULEMUS-2000 Symposium: edited by M. Kerber and M. Kohlhase), A.K. Peters, Ltd, USA, 2001, 126–141.

[10] A. Lyaletski. Evidential Paradigm: the Logical Aspect. In: Cybernetics and System Analysis, Kluwer, vol. 39, No. 5, 2003, pp. 659-667.

[11] D. Brand. Proving Theorems with the Modification Method. In: SIAM Journal of Computing, v. 4, 1975, 412–430.

[12] A. Lyaletski. Computer-oriented calculi of sequent trees. In: Lecture Notes in Computer Science, v. 2942, 2004, 213–230.

[13] R.L. Graham. Rudiments of Ramsey Theory. In: AMS, 1981, 3–6.