Interactive theorem proving and cloud computing

Alexander S. Kleschev

Abstract—In this report various schemes for checking correctness of mathematical proofs are considered. A project of interactive theorem proving based on cloud computing is presented. The main features of this project are extendable language for representing mathematical propositions, extendable calculus of a higher order, two models for intuitive proofs, and also theorems and their proofs represented by mathematical dialect as an input for the system, and automatic transformation of the input into the operational model of intuitive proof.

Index Terms—Interactive theorem proving based on cloud computing, extendable language for representing mathematical propositions, extendable calculus of a higher order, models for intuitive proofs.

By now a fixed scheme has been formed for checking mathematical proofs correctness. In it the responsibility for correctness of a published proof is distributed among the authors, reviewers and interested readers. The proof is considered as correct until an error is detected in it. But the history of mathematics has stories about false results that went undetected for long periods of time. A book written 70 years ago by Lecat [1] contains 130 pages of errors made by major mathematicians up to 1900. The panic at the end of the 19th century because of detecting paradoxes in mathematics resulted in the beginning and following development of a new branch of mathematics which was named “mathematical logic” and of its section called proof theory.

The syntactical direction has become the dominating one in the proof theory. It is based on two theses: any correct mathematical proposition can be formalized; any correct full mathematical proof can be formalized (Hilbert) [2]. Proof theory has suggested another improved scheme for checking mathematical proofs correctness [3]. It reduces this problem to checking of correctness for full formalized proofs. This check can be performed by a mathematician as well as a computer program. But in practice this scheme had not been used up to the appearance of computers because of high complexity of building and formalizing full proofs.

In the late 1950s researches on artificial intelligence began. Within the framework of this field a new scheme was suggested and investigated [4]. This scheme is a radical improving of the above one. According to it the basic complex work is transferred from a mathematician to an automatic theorem proving system, and the problem of full formalized proof check is reduced to the problem of full proof search for a formalized theorem. Intensive theoretical investigation and implementation of this scheme began since the 1960s and continues up till now. However, despite certain successes this scheme is of limited usefulness because of high computational complexity of the procedures which it implements.

In the 1970s implementation of a compromise scheme for checking mathematical proofs correctness began [5]. According to it the main body of complex work is distributed between a mathematician and an interactive theorem proving system. The mathematician has to build full proofs and the interactive system makes the process of their formalizing easier. In the simplest case, when the proof is represented as an inference, such system realizes a computer game in which a user is to build the formula coinciding with the proved theorem. In the more advanced systems evolved in the 1980s decomposition was added to inference, and the important concept of goal list was introduced. The mathematician can decompose a complex goal into simpler goals or prove it by an inference. The first goal is the theorem to be proved. Intensive theoretical investigation and verification of pipelined processors, and so on. Although the basic aim of their implementation has been and is their application in mathematical research, it is the field in which they are not usable yet. Some designers of interactive systems upbraid mathematicians for it, accusing them of base ingratitude. More realistically minded specialists reason that such systems have to go through a long way before they become usable in mathematics. The aim of this report is to show some problems on this way and present a new project of such system containing some solutions of these problems.

In the early systems the first order predicate calculus

Abstract—In this report various schemes for checking correctness of mathematical proofs are considered. A project of interactive theorem proving based on cloud computing is presented. The main features of this project are extendable language for representing mathematical propositions, extendable calculus of a higher order, two models for intuitive proofs, and also theorems and their proofs represented by mathematical dialect as an input for the system, and automatic transformation of the input into the operational model of intuitive proof.

Index Terms—Interactive theorem proving based on cloud computing, extendable language for representing mathematical propositions, extendable calculus of a higher order, models for intuitive proofs.

By now a fixed scheme has been formed for checking mathematical proofs correctness. In it the responsibility for correctness of a published proof is distributed among the authors, reviewers and interested readers. The proof is considered as correct until an error is detected in it. But the history of mathematics has stories about false results that went undetected for long periods of time. A book written 70 years ago by Lecat [1] contains 130 pages of errors made by major mathematicians up to 1900. The panic at the end of the 19th century because of detecting paradoxes in mathematics resulted in the beginning and following development of a new branch of mathematics which was named “mathematical logic” and of its section called proof theory.

The syntactical direction has become the dominating one in the proof theory. It is based on two theses: any correct mathematical proposition can be formalized; any correct full mathematical proof can be formalized (Hilbert) [2]. Proof theory has suggested another improved scheme for checking mathematical proofs correctness [3]. It reduces this problem to checking of correctness for full formalized proofs. This check can be performed by a mathematician as well as a computer program. But in practice this scheme had not been used up to the appearance of computers because of high complexity of building and formalizing full proofs.

In the late 1950s researches on artificial intelligence began. Within the framework of this field a new scheme was suggested and investigated [4]. This scheme is a radical improving of the above one. According to it the basic complex work is transferred from a mathematician to an automatic theorem proving system, and the problem of full formalized proof check is reduced to the problem of full proof search for a formalized theorem. Intensive theoretical investigation and implementation of this scheme began since the 1960s and continues up till now. However, despite certain successes this scheme is of limited usefulness because of high computational complexity of the procedures which it implements.

In the 1970s implementation of a compromise scheme for checking mathematical proofs correctness began [5]. According to it the main body of complex work is distributed between a mathematician and an interactive theorem proving system. The mathematician has to build full proofs and the interactive system makes the process of their formalizing easier. In the simplest case, when the proof is represented as an inference, such system realizes a computer game in which a user is to build the formula coinciding with the proved theorem. In the more advanced systems evolved in the 1980s decomposition was added to inference, and the important concept of goal list was introduced. The mathematician can decompose a complex goal into simpler goals or prove it by an inference. The first goal is the theorem to be proved. Intensive theoretical investigation and verification of pipelined processors, and so on. Although the basic aim of their implementation has been and is their application in mathematical research, it is the field in which they are not usable yet. Some designers of interactive systems upbraid mathematicians for it, accusing them of base ingratitude. More realistically minded specialists reason that such systems have to go through a long way before they become usable in mathematics. The aim of this report is to show some problems on this way and present a new project of such system containing some solutions of these problems.

In the early systems the first order predicate calculus
language was used for representation of mathematical propositions. Apart from the fact that this language is unusual for the majority of mathematicians, the use of it demands to introduce many agreements about translation of mathematical dialect propositions into this language. These agreements cannot be supported by an interactive system, so the process of formalization becomes considerably more difficult. In the advanced systems some fixed formal approximations to the mathematical dialect are used. They include arithmetic and set-theoretic formulas similar to the accepted notation. Such formal languages are useful for mathematicians inside of the domain for which they are designed. But outside of it they present the same difficulties as in the previous case. A radical solution of the problem in the presented project is the extendable formal language which is explicitly represented by its context grammar [6, 7].

The formal system in which proofs are formalized is closely connected with the mathematical proposition representation language. Thus the first order predicate calculus is used in early designs as the formal system. It is uncomfortable since the most of its inference rules are not used in real mathematical proofs. In the advanced designs the language for mathematical proposition representation has the second or higher order. Since a complete predicate calculus of a higher order have an infinite set of inference rules, the majority of such designs is based on a finite subset of this set of rules. It is a rather strong restriction. A radical solution of this problem is a possibility to extend the basic set of the inference rules. In some designs such extendability is obtained by using a high level programming language to describe new inference rules. It presents difficulties for understanding and verifying new inference rules. In the presented project a way which is more traditional for mathematical logic is suggested to extend the basic set of inference rules. It is based on using the propositional logic language and a meta-language which is an extension of the mathematical proposition representation language by means of syntactical variables of different types [8].

Forming the complete knowledge base is one of the problems for using both automatic and interactive theorem proving systems. Most of interactive systems are single-user. Any user has to install the system in his personal computer and to fill out its knowledge base by an appropriate editor. Some systems have finished knowledge bases on various branches of mathematics. The most advanced bases contain thousands of axioms, definitions and theorems. But forming the complete knowledge base by every mathematician for ones personal work is not the best solution.

The presented project is based on the conception of cloud computing. Its users are divided into three classes. They are guests, researchers and integrators. Every researcher gets his empty personal base which is the analogue of his desktop. The common knowledge base is available to every researcher for work and to every guest for reading. The integrators, who are responsible for the state of the common knowledge base, can watch all the personal bases of researchers and select the information useful for the common knowledge base from these personal bases [9].

The proofs which are published in literature are called intuitive ones in the proof theory. An intuitive proof is incomplete, as a rule. In it some propositions are not proved and some steps are omitted. Mathematicians consider forming complete proofs on the bases of intuitive ones as a very difficult task. Thus formalization of intuitive proofs is an important problem of the presented project. This problem is not discussed in scientific literature. A simple but rather general formal model of intuitive proofs has been suggested in [10]. During building of an intuitive proof a set of lemmas are formed by the interactive system according to the following rules. If the objective variable v1 with the domain represented by the term t1 and being a part of a substitution is replaced by another objective variable v2 with the domain represented by the term t2 then the lemma t2 ⊆ t1 is formed; if the objective variable v1 with the domain represented by the term t1 and being a part of a substitution is replaced by the term t then the lemma t ∈ t1 is formed; if during building of the intuitive proof of a goal the mathematician selects the alternative of «the proof is obvious» then the goal becomes a lemma.

Formalizing intuitive proofs by the interactive theorem proving system can be performed by the same way as considered above. But a new problem arises. It is checking correctness of intuitive formal proofs. An incomplete proof is correct if it can be extended up to complete one. According to it an intuitive formal proof is correct if complete formal proofs are built for all the lemmas which are formed during the input of the intuitive proof into the interactive system [10]. It has to be performed by the methods of automatic theorem proving.

Hence we recur to the task of automatic theorem proving but in the easier situation. More simple lemmas rather then initial theorems have to be proved automatically. Since in this case the problems of computational complexity keep all the same, it is possible to use a combination of the following four methods for solving this task. They are associative searching the lemma in the set of earlier proved lemmas, searching the proof by analogy, blind searching and interactive building the proof. The first and the second methods constrict the area of application for the blind search. Their use is based on the observation that the same lemmas arise in the process of proving different theorems, or the proofs of these lemmas are analogous often. The last-named method is used by integrator if the blind search fails in the assigned number of steps. The integrator has to build a complete proof for this lemma by the interactive system or to decide that this lemma is false [10].

Let us enlarge upon methods of searching formal proofs by analogy just in more details. According to the present ideas analogy between proofs is an equivalence relation [11]. Two proofs are analogous if they belong to the same equivalence class. There are three approaches to theorem proving by analogy. In the first one the equivalence classes are determined.
by a program written by a programmer in a high level programming language. The system builds complete formal proofs in the course of running these programs. In such a manner only the task of deduction is solved [12]. In the second approach the task of initial proof generalization is solved, that is, the task of induction. The equivalence class is obtained by introducing the second order variables into the initial proof. Searching the proof, that is solving deduction task, is reduced to searching a substitution instead of these variables. The abduction task is considered as forming lemmas. If these lemmas are correct then a proof of initial theorem can be deduced. This approach is developed only for resolution principle case [13]. The third approach is developed within the framework of the presented project [14] and simulates a deeper analogy. For the extendable mathematical proposition representation language and for the extendable calculus it permits to solve the induction task. The equivalence class is obtained by introducing global syntactical variables into the initial proof. Searching the proof, that is solving deduction task, is reduced to searching a substitution using the knowledge base. The abduction task is considered as forming lemmas automatically or interactively. The new deduction task is searching new theorems together with proofs using the knowledge base. These theorems are analogous to early proved ones.

The formal model of intuitive proofs discussed above contains some elements which are not found in informal intuitive proofs. The mathematician has to select an applied inference rule from the knowledge base at every step of building the proof. If the proof is in the form of inference then he/she has to indicate the values of its premises. To bring a formal model of intuitive proofs closer to the form of informal intuitive proofs, an operational model of intuitive proofs is suggested within the framework of the presented project. It requires fewer efforts from the mathematician to transform an informal intuitive proof into it. To do it the mathematician has to divide the proof into steps, to formalize all the mathematical propositions of every step, and to associate one or a few operations from the extendable set of operations with every step. Let us consider an example of an intuitive proof and its operational model.

Theorem. If \( x_n \) is tending to the limit \( a \), and \( a > p \) then all the values of \( x_n \) also will be greater than \( p \) beginning with a certain \( n \).

\[
\lim x = a & a > p \Rightarrow \exists N: \forall n: x(n) > p.
\]

Proof method: proof of implication (propositions \( \lim x = a \) and \( a > p \) are added to the list of correct propositions, the initial goal is replaced by the new goal \( \exists N: \forall n: x(n) > p \) that should be presented to the user by the interactive theorem proving system).

Proof method (for the goal \( \exists N: \forall n: x(n) > p \)): sequence of steps.

Step 1. Let \( x_n \) have the limit \( a \) (in the operational model of this proof this step can be omitted since this supposition has been put forward after the choice of the proof method for the initial goal).

Step 2. For any \( p < a \) it is easily to select \( \varepsilon > 0 \) in such a way as it took place \( a - \varepsilon > p \); to do this it is suffice to take \( \varepsilon < a - p \).

Operation: prove \( p < a \Rightarrow \exists \varepsilon: \varepsilon > 0 & \varepsilon < a - p \) (the proposition is proved by the automatic theorem proving subsystem; to pursue a proof, this subsystem first is used propositions from the list of correct propositions, and then it is used propositions from the knowledge base; next the proposition of this step is added to the list of correct propositions).

Step 3. But according to the definition of limit it can be found such a number \( N \), that for \( n > N \) the inequality \( x_n > a - \varepsilon \) is fulfilled.

Operation: prove \( \exists N: \forall n: x(n) > a - \varepsilon \).

Step 4. And consequently the inequality \( x_n > p \) is fulfilled a fortiori.

Operation: prove \( x(n) > p \).

The end of the sequence (it is checked whether or not the goal for which the sequence is the proof belongs to the list of correct propositions; if it does not belong, as in the given case, then this goal is proved by the automatic theorem proving subsystem).

If the interactive theorem proving system builds operational models then the scheme for checking intuitive proofs correctness is similar in the above one. But within the framework of this scheme the mathematician makes less complex intellectual work. It is seen from the scenario of such a system. The mathematician has only to divide the intuitive proof into simple steps, to select an appropriate operation at every step, and to input formalized propositions of this step which are operands of this operation using a picture.

However we can be assured that such a scheme for checking intuitive proofs correctness also will be considered as unacceptable for using in real scientific researches. Because of this another scheme is suggested within the framework of the presented project. It relieves the mathematician of any additional work, and provides solving the problem of checking intuitive proofs correctness. According to this scheme the text of an intuitive proof is automatically transformed into the operational model of this proof. It leads us to the problem of mathematical text analysis, i.e. to the problem of computer understanding mathematical dialect.

Every analyzed text consists of the theorem formulation, of explicit or implicit transfer from the theorem to its proof and of the intuitive proof. Any intuitive proof is a sequence of phrases (steps). Each phrase (step) is a sequence of steps connected by copulas or an elementary step. Every elementary step contains information about the operation of this step and parameters of
this operation which are mathematical propositions. These mathematical propositions can be represented in natural language wholly or in part, and also can be formal as a whole. Preliminary experiments have showed that subjects of these texts are strongly fixed. Unlike the other texts of business prose, these texts contain no phrases content of which is beyond these subjects. Semantics of these texts can be completely described by the operational model of intuitive proofs. A special feature of these texts is the fact that they are written in the combination of three languages. These languages are the informal language for intuitive proofs representation, the informal language for mathematical propositions representation, and the formal language for mathematical propositions representation.

We can remember the famous work by Terry Winograd [15] who was concerned with the similar problem and coped with it successfully. But the completely procedural approach used by him for solving this problem is considered as unacceptable. In the presented project a possibility is investigated to simulate this combination of three languages by a two-level context-sensitive generative grammar. The first level of the grammar generates operational models of intuitive proofs and the formalized mathematical propositions in them. The second level of the grammar generates intuitive proofs and mathematical propositions in mathematical dialect using these operational models. Mathematical texts analysis is the inverse task for this grammar.

Completing this report I will risk to put forward a hypothesis that any system pretending to be relevant in mathematical research for checking proofs correctness will correspond to the following scheme. The input of such system is a text containing a theorem and its intuitive proof represented in the form appropriate for scientific publication. The system transforms this text into a formal model of the theorem and its proof and checks the correctness of the proof within the framework of the formal model. This check is supported by the subsystem for the search of complete formal proofs. The transformation of the input text into its formal model is supported by the grammar of the mathematical dialect. The correctness check and proof search is supported by a knowledge base. The grammar of the mathematical dialect and the knowledge base cannot be fixed. Thus it is necessary to control them. This scheme is the basis for the presented project.

REFERENCES