A Concept of Software Shell for Interactive Mathematical Proof Verification Systems

Alexander S. Kleschev[1], Philip M. Moskalenko[1], Vadim A. Timchenko[1]

[1]Institute of Automation and Control Processes Far Eastern Branch of the Russian Academy of Sciences, Vladivostok, Russia, philipmm@dvo.ru, vadim@dvo.ru

Abstract

The concept of software shell for interactive mathematical proof verification systems is presented. The underlying formal logical system, which is approximated to the mathematical practice of constructing proofs, and the mechanisms for its extension are considered. Languages for representation of bases of formalized mathematical knowledge and reasoning methods, as well as the general syntactic structure of proofs are described.

1 Introduction

Validation (verification) of intuitive proofs of theorems is one of the most important problems arising in mathematical research [1]. Proofs, which are published in the mathematical literature, are referred to as intuitive in the proof theory. Only a complete proof can be considered a correct one, when it is performed within the formal system, for which the following statement is true: if a proof can be constructed for a mathematical proposition, then the latter is true [2]. However, in mathematical practice, which is related to the manual theorem proof construction, such proofs are not constructed because of their cumbersomeness, as well as of the excessive complexity of this process. In 1994, the QED-manifesto [3] was published by anonymous authors, which sets forth ambitious goals – building a corpus of mechanically (automatically) verified mathematics, including the formalization of mathematical proofs, and checking of their correctness.

To date, a promising direction in the use of computers for solving the problem of ensuring the correctness of intuitive proof is the development of software shells (or logical frameworks) for interactive theorem proof construction systems (interactive theorem provers – ITP) creation (PVS, Twelf, Coq, HOL Light, Isabelle/HOL, LCF, etc.) [4, 5]. In such software shells metalanguages are offered for the formalization of mathematical knowledge and deductive systems (calculi). As such metalanguages, programming languages are usually used, that are based on a functional-imperative paradigm, or on some variant of higher order predicate logic, or on some variant of a strictly typed λ-calculus with a polymorphic system of dependent types, standard syntax of arithmetic, set-theoretic and, possibly, other expressions. An advantage of the developed ITPs is that for their underlying formal systems it is true that the truth of a mathematical proposition results from the possibility of constructing its proof. Many of the modern ITPs have been successfully used to verify proofs of complex mathematical theorems [1].

However, the existing ITPs are still unclaimed by most mathematicians in their research. This was confirmed by a revised manifesto [6] published in 2007, in which, although the goals of the original manifesto were confirmed, it was also stated that during the time elapsed since its publication, no significant progress was made in achieving its goals. One of the main reasons for this state of affairs was also pointed out there: formalized mathematics is completely different from the real one. On this basis, it can be concluded that the main obstacle to the extensive use of ITPs is that they use formal logical systems that are far from the visions of mathematicians who perform such work. Performing formalization of mathematical theorems and their proofs is still hard, and there is still a steep learning curve that prevents the mathematical community from adopting this style of work with proofs [1].

In the series of works [7-9] the idea of approximating the formal model underlying the ITP to mathematical practice was expressed. This paper describes the concept of a software shell for interactive mathematical proof verification systems, and a formal logical system, which is approximated to the mathematical practice of

In: Sergey I. Smagin, Alexander A. Zatsarinnyy (eds.): V International Conference Information Technologies and High-Performance Computing (ITHPC-2019), Khabarovsk, Russia, 16-19 Sep, 2019, published at http://ceur-ws.org

constructing proofs. The mechanisms for its extension that can be used as the basis for such a shell are also described.

2 Software Shell Concept

As known, mathematical dialect and methods of mathematical reasoning are not only unfixed, explicitly specified, but continue to evolve with the development of mathematics. Therefore, there is no possibility to construct such a fixed formal system that could be used as a basis for ITP, and which "projection" from the mathematical dialect language would not be extremely cumbersome and difficult to carry out for most mathematicians. In order to be able to approximate formal logical systems to mathematical practice, they must be extensible. The possibility of extending them should be provided not only by the developers of the ITP, but first of all by the users of such systems – members of the mathematical community.

One of the ways to provide such an opportunity is to develop a software shell (an instrumental metasystem) for them that would allow creating applied ITPs, which are already based on the core of a formal system approximate to mathematical practice, and providing *mechanisms for extending* this system. The following components of formal systems should be extensible and changeable: *the language for mathematical knowledge representation*, which describes the axioms, theorems, lemmas, definitions, etc., and the set of *formalized methods of reasoning*, available for mathematicians for constructing theorem proofs. In order to achieve this, the methods of reasoning must be presented explicitly (in declarative form), and *the language for formalized reasoning method representation* must be extensible. Note that these languages are equivalent to the logical language of higher orders and do not support graphic images, geometric figures and interpretations of various constructions: commutative diagrams, Euler-Venn diagrams, etc.

To ensure the extensibility of formal systems at their development stage, an approach based on context-dependent grammars and ontologies is used. Extensibility is achieved by the fact that context-dependent grammars of the abstract syntax for the languages of mathematical knowledge representation and formalized reasoning methods have an explicit declarative representation specified in accordance with the metamodel [10, 11]. Due to this, firstly, the reasoning methods have an explicit declarative representation in the ITP, and, therefore, users can change their set, as well as the reasoning methods themselves; secondly, users have the opportunity to include new rules into the grammar of the language for mathematical knowledge presentation or modify existing ones. The same goes for context conditions. Note that when the bases of the formalized reasoning methods are open to mathematicians for modification and, accordingly, the calculus is extensible, then the question concerning the theorem on the consequence of the truth of a mathematical proposition from its provability remains open. It is obvious that for such a formal logical system, the validity of this theorem is not guaranteed (moreover, it turns out to be inapplicable to this formal system). However, this problem can be addressed to specialists in mathematical logic, whose task is to study and verify the reasoning methods proposed by mathematicians.

The rest of the paper is devoted to a brief description of the *core* and the *extension mechanisms* of the formal system, which can be used as a theoretical basis for applied ITPs created with the use of the shell. Like any formal-logical model, the described formal system defines a formal language for the mathematical knowledge representation – axioms, definitions, theorems, lemmas and other mathematical statements; and the calculus (consistent with this language), in which the correct complete proofs should be constructed. The description of calculus includes the definition of methods for proving mathematical statements; the reasoning methods available to mathematicians in the process of constructing theorem proofs, and the formal language for their representation; proof ontology model – a structure for representing a complete proof.

Finally, it is important to note that the reasoning practice of modern mathematics, which is based on the principle of structuralism, as well as the concepts of isomorphism, equivalence (identity) of mathematical objects [12], is beyond the scope of the work.

3 Ontology Model of Mathematical Knowledge Base

For the purpose of structured storage and accumulation of mathematical knowledge, this section introduces an ontology model that specifies the network structure of various mathematical sections and knowledge bases.

Each section of mathematics has its own name and may contain the following: a set of definitions allowing to introduce new ones to refer to defined concepts, as well as set meanings of these definitions; a set of axioms; a set of theorems and lemmas, each of which can have a set of corollaries; a set of named subsections (Figure 1). Hereinafter, the notation of the labeling of vertices and arcs for the digraphs shown in the figures coincides with the notation used in [10, 12], where its semantics is described in detail.

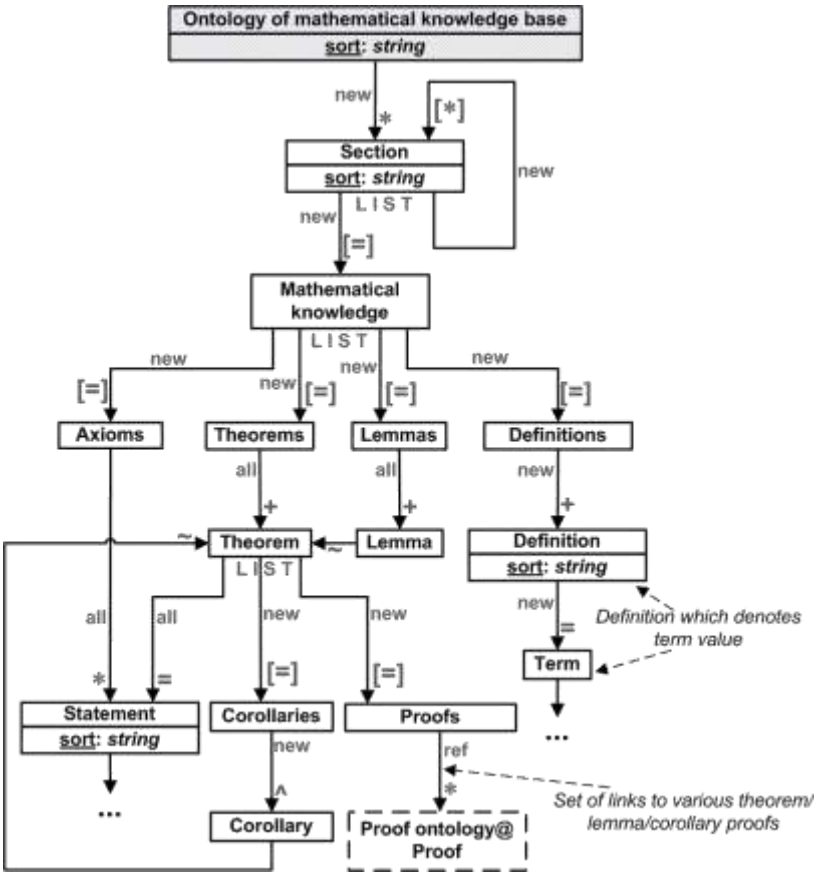

Figure 1: Structure specification for sections of mathematics

4 The Language for Mathematical Knowledge Representation

Figure 2 shows a fragment of a generative graph grammar describing the ***abstract syntax*** of the language for mathematical statement representation. Each *mathematical statement* is represented by the vertex of the grammar digraph "*Statement*" and has the form: $(v_1: t_1)$ … $(v_n: t_n)$ *f*. Here *f* – a mathematical formula containing occurrences of object variables $v_1, \ldots, v_n$, and $(v_i: t_i)$ $(i = 0, \ldots, n)$ – a description of the object variable v_i, t_i – a mathematical term whose value is the set (the range of possible values of the variable v_i). Mathematical statement models the sentence of a mathematical dialect: «for any values of v_1 from t_1, …, v_n from t_n – *f* is a correct formula».

The following types of terms are distinguished: terms representing arithmetic expressions; various types of intervals, as well as the sets that they form; quantified terms (with finite and infinite ranges of possible values of variables); terms representing set operations, as well as labeling of some predetermined sets; terms representing mappings; conditional term which models a sentence of mathematical dialect: «if f_1, then t_1, … if f_n, then t_n».

The language for mathematical statement representation contains the following ***context conditions***.

Context conditions for object variables.

1. For each using occurrence of a variable in the body of a statement or of some quantified construction, there is a defining occurrence of this variable in the prefix of this statement or this quantifier construction.

2. For each defining occurrence of a variable in the prefix of a statement or some quantified construction, there exists a using occurrence of this variable in the body of this statement or the body of this quantified construction.

3. A term, which is a range of possible values of a defined variable, cannot contain occurrences of the same variable.

Context conditions for matching the number of formal and actual parameters.

1. If the *predicate application* has the form $\rho(t_1, \ldots, t_m)$, then:

- if ρ is a *definition*, then the *definitions* set of some section of mathematical knowledge must contain a definition of the form $\rho \equiv \lambda(v_1 : tt_1)$ … $(v_m : tt_m)$ *f*, where *f* – a formula, which contains occurrences of object variables $v_1, \ldots, v_m$, $m \geq 1$;

- else if ρ is a *variable*, then its declaration must have a form $\rho : tt_1 \times \ldots \times tt_m \rightarrow$ *set of logical values*, if $m > 1$, or $\rho : tt_1 \rightarrow$ *set of logical values*, if $m = 1$.

2. If *function application* has a form $\rho(t_1, \ldots, t_m)$, then:

- if ρ is a *definition*, then the *definitions* set of some section of mathematical knowledge must contain a definition of the form $\rho \equiv \lambda(v_1 : tt_1) \ldots (v_m : tt_m)\ t$, where t – a term, which contains occurrences of object variables $v_1, \ldots, v_m$, $m \geq 1$;

- else if ρ is a *variable*, then its declaration must have the form $\rho : tt_1 \times \ldots \times tt_m \to tt$, if $m > 1$, of $\rho : tt_1 \to tt$, if $m = 1$.

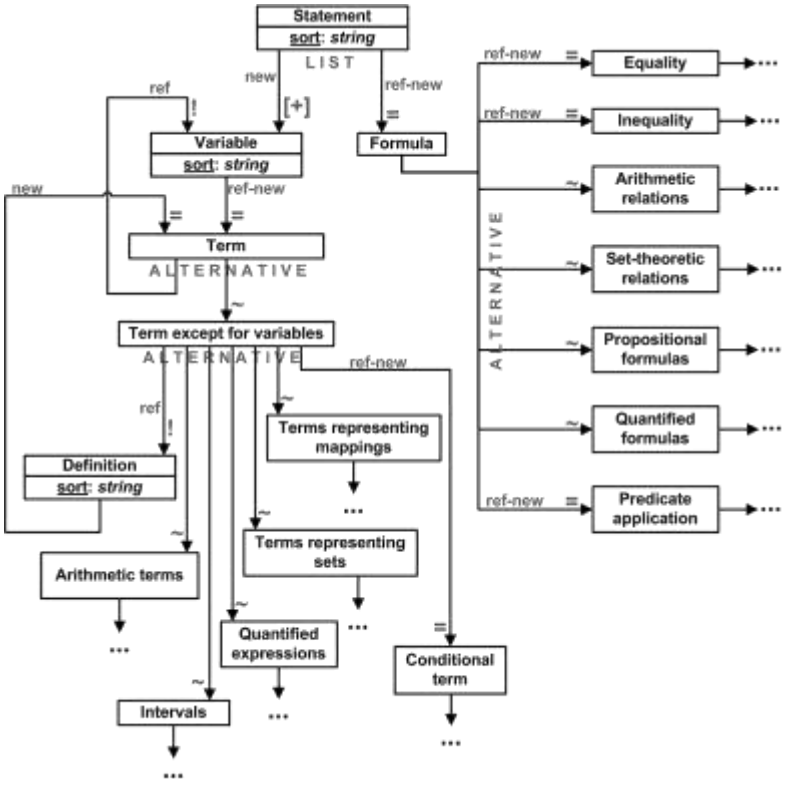

Figure 2: Fragment of generative graph grammar describing the abstract syntax of the language for mathematical statement representation

The *extension* of the language for mathematical statement representation is provided by the addition of new types of terms and formulas. It consists of adding a new construct to the grammar digraph of the language, describing the abstract syntax of this new construct, adding the necessary rules to the text grammar of the language, and possibly describing the context conditions associated with the new construct.

The mathematical knowledge base is formed in accordance with its ontology model. *Extension* of the mathematical knowledge base is to add new mathematical knowledge – definitions, axioms, theorems and lemmas (as well as their corollaries) to existing sections of mathematics; and to create new sections / subsections of mathematics and filling them with relevant mathematical knowledge.

5 Proof Methods

Calculus, within which a proof is constructed, includes methods of proving *goals* based on three rules. Before describing the rules themselves, we define the concept of a *goal* and a *true statement*. A *goal* is a pair: *mathematical statement* and a possibly empty *list of assumptions* – a set of mathematical statements, which truth results in the truth of this statement.

By a *true statement* we mean a mathematical axiom, a definition, a theorem/lemma (as well as its corollary), for which a proof has already been constructed, or a *method of reasoning* (see section 6 for more details). Definitions and axioms are considered correct by agreement between members of the mathematical community.

Next, we describe the inference rules of the calculus.

1*. *The rule of implication (natural deduction). It allows to reduce the proving of the goal, for which the *mathematical statement* has the form of implication $f_1 \,\&\, \ldots \,\&\, f_k \Rightarrow f$, and a *list of assumptions* is $p_1, \ldots, p_n$ ($n \geq 0$), to the proving of a goal which *mathematical statement* is the consequent of this implication – f, and a *list of assumptions* is $p_1, \ldots, p_n, f_1, \ldots, f_k$.

2*. *Matching rule. If φ is a true statement, and there is a substitution of θ instead of variables in φ, such that the result of applying this substitution to φ coincides with the *mathematical statement* f of the goal being proved or is

syntactically equivalent to *f*, then *f* is true (*f* is a *concretization* of φ). The *list of assumptions* for the goal is empty in this case. It should be noted that the matching is a unidirectional version of unification [13].

3. ***Modus ponens.*** It is used to *decompose a goal*, *decompose* some *assumption* from the list of assumptions of a goal, or to perform *inference*.

3.1. *Goal decomposition*. If it is necessary to prove a goal which *list of assumptions* is $p_1, \ldots, p_n$, $(n \geq 0)$, and *mathematical statement f* can be matched with the consequent of a true statement, having the form of $\varphi_1 \,\&\, \ldots \,\&\, \varphi_m \Rightarrow \varphi$, and θ – is the most general unifier of *f* and φ, then this proving is reduced to proving of goals, which mathematical *statements* $f_1, \ldots, f_m$ are respectively the results of applying the substitution θ to statements in the antecedent of this $f_1 = \varphi_1\theta, \ldots, f_m = \varphi_m\theta$, and a *list of assumptions* of each goal is $p_1, \ldots, p_n$.

3.2. *Assumption decomposition*. If it is necessary to prove a goal which *mathematical statement* is *f*, and a *list of assumptions* is $p_1, \ldots, p_i, \ldots, p_n$ $(n \geq 1)$, and assumption p_i can be matched with the RHS of the equivalence φ of the true statement, which has a form of $\varphi_1 \mid \ldots \mid \varphi_m \Leftrightarrow \varphi$ $(\varphi_1 \vee \ldots \vee \varphi_m \Leftrightarrow \varphi)$, $(m \geq 2)$, and θ is the most general unifier of p_i and φ, then this proving is reduced to proving of *m* goals, which *mathematical statement* is *f*, and *lists of assumptions* are $p_1, \ldots, pp_1, \ldots, p_n$, where pp_1 is the result of applying the substitution θ to φ_1; …; $p_1, \ldots, pp_m, \ldots, p_n$, where pp_m is the result of applying the substitution θ to φ_m.

3.3. *Inference*, which is a sequence of *inference steps*.

An inference step is the following. Let the mathematical statement f_i $(i = 1, \ldots, m)$ be either an axiom, or a definition, or a proven theorem/lemma (or its proven corollary), or a statement from the list of assumptions of the goal being proved, or the result of one of the previous inference steps. Then if the statement $f_1 \,\&\, \ldots \,\&\, f_m$ can be matched with the antecedent of a true statement that has the form $\varphi_1 \,\&\, \ldots \,\&\, \varphi_m \Rightarrow \varphi$, and θ is the most general unifier for $f_1 \,\&\, \ldots \,\&\, f_m$ and $\varphi_1 \,\&\, \ldots \,\&\, \varphi_m$, then the statement *f* is true, which is the result of applying the substitution θ to φ (*inference step result*). The goal is considered proven if the result of the last inference step of the conclusion is a statement that coincides with the *mathematical statement* of the goal, or is syntactically equivalent to it.

In the case of goal decomposition (***3.1***) and inference (***3.3***), the ***Modus ponens*** application can be generalized to the case when the implication is replaced by equivalence.

6 Formalized Reasoning Methods

Methods of reasoning, which are used in the methods of proof, are divided into *two classes*:

- based on propositional tautologies underlying logical reasoning;
- based on mathematical principles and statements about syntactic transformations of mathematical expressions.

These methods of reasoning are represented explicitly by propositional formulas, which are tautologies, and metamathematical statements, respectively. Metamathematical statements are considered correct if their validity is established on the basis of an intuitive or conventional (as a result of an agreement between members of the mathematical community) criterion. Thus, the proof is considered correct if the validity of all metamathematical statements used in it is accepted.

Each *propositional tautology* has the form: $v_1 \ldots v_n\ pf$, where *pf* is a propositional formula containing the occurrences of propositional variables $v_1, \ldots v_n$, and v_i $(i = 1, \ldots, n)$ is the description of the propositional variable v_i, $n \geq 1$. The value of a propositional variable can be one of the logical constants – *true* or *false*.

There are the following ***context conditions*** for propositional variables.

1. For each using occurrence of a variable in a propositional formula, there is a defining occurrence of this variable in the prefix of this formula.

2. For each defining occurrence of a variable in the prefix of a formula, there exists a using occurrence of this variable in the body of this formula.

The generative graph grammar, which describes the ***abstract syntax*** of the language for metamathematical statement representation (metalanguage), is shown in figure 3. Each *metamathematical statement* is represented by a vertex of the graph grammar «*Mathematical statement*» and has a form: $(v_1{:}\ t_1) \ldots (v_n{:}\ t_n)\ \boldsymbol{t_1} \ldots \boldsymbol{t_k}\ \boldsymbol{f_1} \ldots \boldsymbol{f_s}\ \boldsymbol{i_1} \ldots \boldsymbol{i_p}\ \boldsymbol{r_1} \ldots \boldsymbol{r_q}\ f$. Here *f* – a mathematical formula, containing occurrences of object variables $v_1, \ldots, v_n$, as well as occurrences of syntactic variables $\boldsymbol{t_1}, \ldots, \boldsymbol{t_k}$ of type ***t***, syntactic variables $\boldsymbol{f_1}, \ldots, \boldsymbol{f_s}$ of type ***f***, syntactic variables $\boldsymbol{i_1}, \ldots, \boldsymbol{i_p}$ of type ***i*** and syntactic variables $\boldsymbol{r_1}, \ldots, \boldsymbol{r_q}$ of type ***r***. Herein $(v_i{:}\ t_i)$ $(i = 0, \ldots, n)$ is a description of the object variable v_i, t_i is a mathematical term, which value is a set (range of possible values of the object variable v_i); $\boldsymbol{t_1} \ldots \boldsymbol{t_k}$ are descriptions of syntactic variables of type ***t*** $(k \geq 0)$, which values are terms; $\boldsymbol{f_1} \ldots \boldsymbol{f_s}$ are descriptions of syntactic variables of type ***f*** $(s \geq 0)$, which values are formulas; $\boldsymbol{i_1} \ldots \boldsymbol{i_p}$ are descriptions of syntactic variables of type ***i*** $(p \geq 0)$, which values are integer constants; $\boldsymbol{r_1} \ldots \boldsymbol{r_q}$ are descriptions of syntactic variables of type ***r*** $(q \geq 0)$, which values are real constants; where $k + s + p + q > 0$.

The metalanguage is a superstructure on the language for mathematical statement representation and is obtained by extending the «*Formula*» and the «*Term*» constructions of this language. The «*Formula*» construction is extended by adding two alternatives: «*Syntactic variable of type* ***f***» and «*Modified syntactic variable of type* ***f***». The «*Term*» construction is extended by adding four alternatives: «*Syntactic variable of type* ***i***», «*Syntactic variable of type* ***r***», «*Syntactic variable of type* ***t***», and «*Modified syntactic variable of type* ***t***».

The modified syntactic variable contains a modifier in addition to the name. The variable is a term or formula depending on its type. The modifier consists of modifier elements, each of which is a term or a formula. The value of such a syntactic variable is a syntactic construction, which corresponds to the variable's type, and contains formal parameters. Each modifier element corresponds to its own formal parameter, which can be included in the value of

the syntactic variable one or more times. Modifier elements with the same sequence number in different occurrences of the modified syntactic variable in the same metamathematical statement correspond to the same formal parameter. The modifier elements themselves are the actual parameters. The value of the occurrence of the modified syntax variable in the metamathematical statement is the value of this syntax variable, in which all formal parameters are replaced by the actual ones.

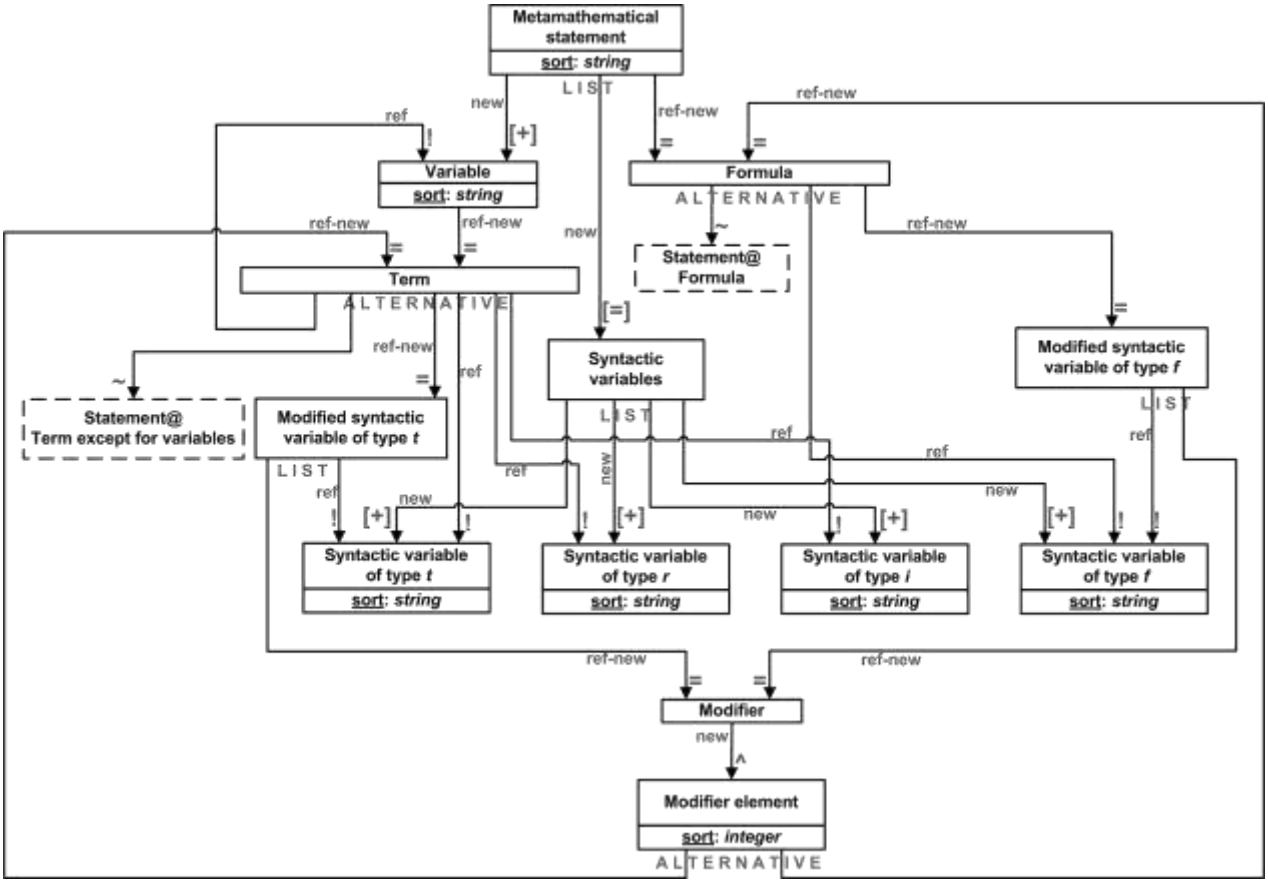

Figure 3: Generative graph grammar describing the abstract syntax of the language for mathematical statement representation

Metamathematical statements are considered valid for any legal values of syntactic (including modified) variables.

For the language for metamathematical statement representation, the following ***context conditions*** for syntactic variables are formulated:

1. each metamathematical statement must include at least one syntactic variable;
2. for each using occurrence of a syntactic variable in a term or a formula of a metamathematical statement, there is a defining occurrence of this variable in the prefix of this statement;
3. for each defining occurrence of a syntactic variable in the prefix of a metamathematical statement, there is a using occurrence of this variable in the term or formula of this statement;
4. a modifier of a modified syntactic variable cannot contain both direct and indirect occurrences of the same syntactic variable;
5. a syntactic variable can be included in a metamathematical statement either unmodified or modified (but not in both ways);
6. if a syntactic variable is modified, then it must be included in the metamathematical statement at least two times;
7. the number of modifier elements in all occurrences of the same modified syntactic variable in the metamathematical statement is the same, and at appropriate places in the modifier there must be either terms or formulas.

The definition of a metalanguage as a superstructure on the language for mathematical statement representation (that is, as its *extended language*) is a sufficient condition for the metalanguage to ***extend automatically*** when the language for mathematical knowledge representation extends. This is true due to the method of defining the «*Formula*» and the «*Term*» metalanguage constructions through the same constructions of the mathematical knowledge representation language, which are extended by the corresponding syntax variables (see figure 3), and also by adhering the context condition **1** for the syntactic variables. Thus, the extension of these two languages occurs *simultaneously*.

The base of formalized reasoning methods consists of a variety of those, modeled by propositional tautologies, and a variety of those, modeled by metamathematical statements. ***Extension*** of the base of formalized reasoning methods is in extending the set of propositional tautologies and/or the set of metamathematical statements. The set of propositional tautologies is *extensible* due to the possibility of formulating new propositional formulas, which, in case of successful automatic verification of their validity, are added to this set. The set of metamathematical statements is *extensible* due to the possibility of formulating new statements that are included in this set. In particular, it should be noted that if new quantifiers are added to the language for mathematical knowledge representation, it becomes necessary to add metamathematical statements about the properties of these quantifiers.

7 Proof Model

The complete proof is a syntactic structure, which is a set of related goals. The first goal is the *theorem/lemma* being proved, for which there is no list of assumptions. Each goal has a proof method, associated with it. The set of admissible methods for proving a goal is an improper subset of the set of methods described in Section 5. The number of methods can vary from three to five – depending on the syntactic form of the goal's *mathematical statement* (whether it has the form of implication or not) and the presence or absence of its assumptions. The syntactic structure of each proof method is based on its semantics (see Section 5). In the case of using the ***Modus ponens*** rule for *goal decomposition* or *inference*, the following is taken into account: the form of *true statement* (implication or equivalence) and the rules for choosing values for the premises. A premise is conjunct from the implication antecedent or from the equivalence side, which is considered to be an antecedent.

In the case of goal decomposition, the values for the premises can be chosen from the statements stored in the mathematical knowledge base, in the set of proven statements, and in the list of assumptions of the goal being proved (if it has assumptions).

In the case of inference, the values for premises can be selected from all three sets of statements mentioned above, and from statements that are results of already performed inference steps (if at least one inference step has been performed).

8 Conclusion

The paper presents the concept of a software shell for systems of intuitive mathematical proof verification. The core of the formal logical system, which is approximated to the mathematical practice of constructing intuitive mathematical proofs, is specified, and the mechanisms for its extension are considered. This formal system can be used as the basis for proposed shell. The declarative specifications of extensible languages for representation of mathematical knowledge and formalized methods of reasoning, as well as model of complete proofs, have been developed. The language for formalized reasoning method representation consists of two sublanguages: the language for propositional tautology representation and the metalanguage.

Only the language for mathematical knowledge representation has mechanisms of extension. The metalanguage, by the method of definition, extends automatically as the former is extended. The extensibility of the language for mathematical knowledge representation is provided by the extensibility of the set of definitions, allowing to introduce new ones to designate the defined concepts, as well as the extensibility of its grammar. The latter is achieved through the aids for describing the syntax of new constructions of the language for mathematical statement representation, as well as, if necessary, context conditions. The calculus, within which proof is constructed, is presented explicitly and is extensible.

The solution for the problem of the consequence of the mathematical proposition truth from its provability may be entrusted to specialists in mathematical logic. Their task in this case is to verify the methods of reasoning specified by mathematicians on the metalanguage. As a result of the analysis a class of axioms can be distinguished among the metamathematical statements, and the rest are classified either as requiring proof or as incorrect. Thus, proofs that use incorrect or unproved metamathematical statements cannot be considered true.

The results obtained in this research can be used in a project for the development of a QED-system as well as in projects for the development of controlled ITPs, which are approximations to such project.

Acknowledgements

This work was partially supported by RFBR (projects nos. 17-07-00299 and 19-07-00244) and by PFI "Far East" (project no 18-5-07).

References

1. Maric, F.: A Survey of Interactive Theorem Proving. Zbornik Radova; 18(26): 173-223 (2015)

2. Mendelson, E.: Introduction to mathematical logic [In Russian]. - Moscow: Nauka; (1976)

3. The QED Manifesto. Automated Deduction, Springer-Verlag, Lecture Notes in Artificial Intelligence; 814: 238-251, (1994) http://www.cs.ru.nl/~freek/qed/qed.html

4. Asperti, A.: A Survey on Interactive Theorem Proving. (2009). http://www.cs.unibo.it/~asperti/SLIDES/itp.pdf

5. Harrison, J., Urban, J., Wiedijk, F.: History of Interactive Theorem Proving. In Jörg Siekmann (ed.), Handbook of the History of Logic, Computational Logic. Elsevier; 9: 135-214 (2014)

6. Wiedijk, F.: The QED Manifesto Revisited. Studies in Logic, Grammar and Rhetoric; 10: 121-133. (2007) http://mizar.org/trybulec65/8.pdf

7. Gavrilova, T.L., Kleschev, A.S.: An internal model of mathematical practice for systems of theorem provings automated construction. Part 1. General description of the model [In Russian]. Control Sciences; 4: 32-35 (2006)

8. Gavrilova, T.L., Kleschev, A.S.: An internal model of mathematical practice for systems of theorem provings automated construction. Part 2. A model of mathematical dialect [In Russian]. Control Sciences; 5: 68-73 (2006)

9. Gavrilova, T.L., Kleschev, A.S.: An internal model of mathematical practice for systems of theorem provings automated construction. Part 3. A model of proof [In Russian]. Control Sciences; 6: 69-71 (2006)

10. Gribova, V.V., Kleshchev, A.S., Moskalenko, F.M., Timchenko, V.A.: A Two-level Model of Information Units with Complex Structure that Correspond to the Questioning Metaphor // Automatic Documentation and Mathematical Linguistics. Vol. 49. No.5 (2015)

11. Gribova, V.V., Kleshchev, A.S., Moskalenko, F.M., Timchenko, V.A.: A Model for Generation of Directed Graphs of Information by the Directed Graph of Metainformation for a Two-Level Model of Information Units with a Complex Structure // Automatic Documentation and Mathematical Linguistics, Vol. 49, No.6 (2015)

12. Homotopy Type Theory: Univalent Foundations of Mathematics. The Univalent Foundations Program, Institute for Advanced Study; (2013). http://homotopytypetheory.org/book

13. Knight, K.: Unification: A Multidisciplinary Survey. ACM Computing Surveys; 21(1): 93-124 (1989)