ForTheL — the language of formal theories

Konstantin Vershinin, *Dept. Math-Info, Paris 12 University, 61, avenue du General De Gaulle, 94010 Creteil, France*, verko@logique.jussieu.fr
Andrey Paskevich, *Cybernetics Dept., Taras Shevchenko University of Kyiv, 6, Glushkov avenue, 252187 Kyiv, Ukraine*, andrey@raptor.kiev.ua

The paper describes the language ForTheL (FORmal THEory Language) aimed at representation of formal theories — in strict mathematical sense of the term. Its syntax is strictly formalised in the terms of BNF and its semantics can be precisely defined with the help of syntactical transformation that maps ForTheL-phrases to the formulae of first-order logic with equality. At the same time ForTheL is intended to be close to the natural language of mathematical texts, it formalises and uses expressive means of the human language.

The short history of the language is as follows. In 1960s academician V.Glushkov initiated a project — called "Algoritm Ochevidnosti" — that was aimed at mathematical text processing "in large" on the base of evolving an automated theorem proving system [1]. One of the goals was to help to working mathematicians in constructing and verifying automatically long but in some sense "evident" proofs. The "Algoritm Ochevidnosti" project included three main components: the deductive part, mathematical knowledge base and finally the language for knowledge representation — the component uniting other two. The core of the deductive part was so-called "evidence routine" that established the evidence in the terms of some deduction technique (cf. Lyaletski-Degtyarev calculus [2]). Besides, there were presupposed various means enforcing the evidence routine — the search for the auxiliary lemmata or for some other relevant information, involving the user in the proof search and so on.

The first sketch of the language in question was given in [3]. Then the work was continued by V.Bodnarchuk and finally it was resulted in TL language [4] developed by a research group headed by K.Vershinin in the end of 70s in the frames of the same project "Algoritm Ochevidnosti". Then the project was temporary interrupted and now it is continued.

Thus, the following requirements to the language of the system were formulated. The language should possess the formalised syntax and semantics. It should permit the formulation of the theory axioms, theorems and proofs along with the definitions in order to provide the self-contained texts. Accordingly, the thesaurus of the language should be separated from the grammar to be enrichable. On the other hand the language should be close to the natural language of the mathematical publications. It is required by considerations of the user comfort during text creation or in interactive dialog with the system. Moreover it allows to formulate the tasks not connected with the inference search directly.

So, given a text, the language should provide an initial environment for the evidence routine and for the enforcing means that depends on the problem to be solved. In this environment the proof search, the search for the auxiliary propositions, etc are performed. Starting from theese requirements there were elaborated the main principles of such a language, implemented in TL and then in ForTheL.

Basically a ForTheL text can be viewed as a collection of phrases possessed of some structure that provides certain "navigation rules" in the text. Therefore it seems to be reasonable to distinguish two components of the language which are, in some sense, orthogonal. The first one is the language of the ForTheL-phrases, the second one defines the composition of the whole text — phrase grouping, section naming, variables scoping rules, etc.

The syntax and semantics of the ForTheL-text.

Syntax. Syntactically, the ForTheL-text is an ensemble of sections. Each section, in its turn, can contain the sections of lower level and/or phrases. The boundaries of a section are marked with the separators that additionally define a partial order — a kind of subordination relation — on the totality of all sections of the given text. The syntax of the separators can be arbitrary enough but we prefer to follow the usual practice of numeration (dot notation) accepted in the mathematical texts (see e.g. LNCS style for LaTeX).

Some sections — theorems, proofs, definitions — play the special role in the language. They are distinguished by a special syntax of their separators that however follows the usual mathematical practice, too.

The section containing only phrases will be called "primary". Every phrase either affirms or asserts something. The distinction between affirmations and assertions is also expressed with the simple syntactical means.

Semantics. The semantics of the ForTheL-text is determined by the task formulated by a user. Here are some examples of such a task: "check the correctness of the text", "check the correctness of the statement in the text", "find a counter example for the statement in the text", "find out a subject of the text", "search in the text for some information relevant to ..." and so on.

Consider, for example, the second task. Let the statement in the given text be correct if it can be formally deduced from its "predecessors". Such a specification requires a proper definition of the "logical predecessor" that can be given in a quite simple way with the help of the text inspection — here the separators play the role of road signs — and with the replacement of some sections by their "main affirmations" — thus the whole "theorem" section can be replaced by its formulation.

The syntax and semantics of the ForTheL-phrase.

The language of ForTheL-phrases contains the first-order language as its proper subset. In addition it has two key features — the verbal scripting of language constructions and so-called concepts that are analogues of nouns.

Verbal scripting. The verbal scripting allows to construct ForTheL-phrases like the phrases of the natural language — using subjects, predicates, objects and attributes. ForTheL also provides the short symbolic notation accepted in the mathematical logic.

Logical connectives as well as several predicates and concepts are predefined syntactical units of the ForTheL. The other syntactical units have to be introduced with the help of the definitions.

The predicate and function symbols of the ForTheL are defined in the verbal form, possibly, with the corresponding symbolic notation. The predicates are expressed with the verbs or adjectives, the functions — with the nouns. Thus, the statement $P(A) > n$ can be written as "*power of A is greater than n*". The use of verbal form inside symbolic one is prohibited. Furthermore, when speaking about verbal forms we will use the word "pattern" instead of "function symbol" or "predicate symbol".

The pretty feature of the predicate patterns is that they allow to put more than one term in the first argument place. Such a record is translated as a conjunction of the appropriate literals. "*a, b divide n*" suchwise will become $(n|a \wedge n|b)$. In the future the language will also provide the correct processing of the expressions like "*a, b are equal*" for the symmetric transitive binary predicates.

The ForTheL regulates the use of articles, plural nouns, present participle tense. Passive voice is planned. It's presupposed the permanent extension of the language expressive means with the expressions which are in general use.

ForTheL concepts. The verbal scripting is conditioned by the approaching the ForTheL to the natural language and could be considered as a "cosmetic" feature. However, another key property of ForTheL — the concepts — grounds on it. The use of concepts, at first, corresponds to the human style of narration and, at second, can assist the inference search during ForTheL-text processing.

A concept could be considered as an analogue of a term of the first-order language but its value is not an individual object but a (parametrised) collection of objects. The example of a concept is "*relation on X*", where the variable X is declared as a set (another ForTheL concept). The collection or class of objects should not be identified with the set containing that objects. From the point of view of the ForTheL semantics, the set is an individual object as well as its elements, whereas the class specified by a concept is not an element of the same domain.

With each concept pattern of the arity n the characteristic predicate symbol of the arity $n+1$ is associated. The corresponding relation expresses that an object belongs to the class specified by the concept. This predicate symbol can be specified directly in the text of the definition of the new concept pattern as its symbolic notation. For the concept pattern "*function from $\pi_1$to π_2*" such a predicate symbol could be "π*:* $\pi_1 \rightarrow \pi_2$" (with the π we mark the argument places).

Reduction and expansion of concepts. One can make concept more precise, i.e. narrow the corresponding class, with the special modifiers — attributes and post-conditions. One kind of attributes adds a constraint on the element of the class, e.g. "*empty set*" or "*number dividing N*". Another one adds a constraint on the value of some unary function defined on the class, e.g. "*set of power K*", "*set of finite power*". It's easy to see that the attributes are derived from the predicate patterns. ForTheL grammar allows to use several attributes together. The post-condition is an ordinary ForTheL statement connected to the concept pattern with "*such that*" connective.

One can also expand a concept by decreasing the arity. Thus, for the defined concept pattern "*relation on π*" the incomplete concept "*relation*" will specify the class of all the relations on all the sets from the domain instead of some specific set.

Variables and sorts. In ForTheL a new variable is declared either with the help of the statement that asserts its equality to a certain term or with the help of the statement that asserts its membership of the class specified by a certain concept. Both statements can be expressed explicitly with the help of the predicates "*is equal to*" and "*is a*" (in their verbal forms only), as well as implicitly, by naming of the functions and concepts. For example, the statement "*set _ A contains x*" implicitly declares the A as a set variable. The underscore sign marks naming.

The term "sort" above means — despite usual practice — a concept pattern. When defining the new concepts through defined ones we create the hierarchy of the sorts. The sort of a predicate pattern or a concept pattern is a tuple of the sorts of its argument places, the sort of a function pattern is a tuple of the sorts of its argument places along with the sort of the value of the function.

In the well-formed ForTheL-text every variable, when declared, is associated with its sort — the pattern of the concept specifying the class the variable belongs to. Thus we have an additional correctness control.

Sometimes the hierarchy of the sorts can suggest the right direction of the inference search. The partial order on the set of the sorts can be used by the prover to involve those premises first that contain concepts connected to the goal.

ForTheL terms, or, alternatively, the language objects that one can put into the argument places of the patterns of the predicates, functions and concepts, have the following forms:

- First-order language term.
- Named concept with the terms in the argument places. The named concept implicitly declares the new variable and the term is replaced with this variable.
- Concept with the terms in the argument places prefixed by "*class of*" lexem. The value of the term will be the set consisting of the elements of the class specified by the given concept.
- Concept with the terms in the argument places prefixed by the lexems "*some*" or "*every*". The term is replaced with the new variable of the appropriate sort and receives a logical context expressed with the existential or universal quantifier bounding this variable. The context is then processed depending of the way the term was used. For example, the statement "*all divisors of some number are simple*" will be translated as $\exists n(number(n) \wedge \forall d(divisor(d,n) \supset simple(d)))$.

Quantifier usage. The ForTheL allows to write the quantifiers bounding the concepts: "*for every simple number* _ *P true that P is odd*". The statement given is translated as a formula under the universal quantifier bounding the variable P whose sort is a class of simple numbers. The concept can be unnamed: "*there exists an even simple number*" — then new variable is declared.

For the unary concepts ForTheL provides a synonym for the existential quantifier. The statement "*there exists a divisor of N*" can be written as "*N has a divisor*".

New pattern definitions. In the ForTheL the concept patterns "*class*" and "*set*" and the predicate patterns "*is equal to*", "*belongs to*" and "*is a*" are predefined. As mentioned above, all the other patterns and symbols are defined directly in the ForTheL-text. Though, to formalise a particular theory, it would be more convenient to extend the list of the predefined patterns than to reduce them to the set-theoretical level. Notice that the patterns of the characteristic predicates of the concepts allow to eliminate the predicate "*is a*" during translation.

The definitions of the new patterns are ForTheL phrases of a specific form. Without going into details one can say that predicate patterns are defined by indicating their validity condition, function patterns — by equating them to a term or by indicating the criterion of the equality (possibly with the proof of the functionality), concept patterns — either by narrowing some concept or by indicating the characteristic relation. ForTheL allows the recursive definitions too.

Conclusion.

Since the basis of the language are the fundamental logical and set-theoretical relations, it's suitable for the representation of any (not only mathematical) knowledge if the latter can be formalised in the terms of classical logic. An ForTheL-text can be easily created by a human as well as by a computer program, then it can be passed to a prover or another program, directly or through the Net.

ForTheL can be used as a tool to write and verify mathematical publications and formal specifications, to perform model checking and so on. We also propose to use it as the universal interface of declarative mathematical knowledge bases, as means of the integration of computer mathematical systems.

By now the English and Russian versions of the ForTheL grammar have been developed and software translator has been implemented. The implementation of the original prover that uses ForTheL as a language of input texts and output proofs was recently started. The research is held in the frames of the project "Evidence Algorithm 2000" by the joint group of University of Kiev and Institute of Cybernetics of Kiev.

Appendix A. Concept syntax.

Here we give a fragment of the ForTheL grammar describing the syntax of the concepts.

```
term:           [ ALL | SOME | CLASS OF ] concept
          |     function                            //  power of S
          |     symbolic_term                       //  P(S)

concept:        { left_attr } free_concept { right_attr } [ SUCH THAT statement ]

free_concept:   C_TOK [ _ var ]
          |     C_TOK [ _ var ] C_TOK term
          |     C_TOK [ _ var ] C_TOK term C_TOK term

left_attr:      left_constr

right_attr:     right_constr
          |     OF F_TOK term                      // ... of power N
          |     OF left_constr F_TOK               // ... of nonzero power
          |     OF F_TOK right_constr              // ... of power less than N

left_constr:    [ NOT ] P_TOK                      // empty ...

right_constr:   [ NOT ] EQUAL TO term
          |     [ NOT ] BELONGING TO term
          |     [ NOT ] P_TOK term                 // ... dividing N
          |     [ NOT ] P_TOK P_TOK term           // ... less than N
```

Appendix B. ForTheL-text example.

The valid ForTheL text given here introduces some definitions of the set theory.

```
Definition 1.   Element of A is a set belonging to set _ A.

Definition 2.
  A is empty IFF wrong that exists an element of set _ A.

There exists unique empty set.

Definition 3.   Empty_set (0) is equal to any empty set.
```

```
Definition 4.   Let A, B be sets.
  A is a subset of B (A \sub B) IFF all elements of A belong to B.

Lemma 1. Each set has a subset.
Proof. 0 is a subset of all sets. QED.

Definition 5.
  Power_set of A (PS[A]) is equal to the class of subsets of set _ A.

Definition 6 "Tarski".   Let A be a set.
  A is finite  IFF  for every not empty subset _ K of PS[A]
     there exists subset _ S of A belonging to K  such that
       S is not a subset of any element of K not equal to S.
```

References

[1] V.Glushkov. Foundations of mathematics and automation of deductive constructing. (in Russian) // Voprosy teoriyi i praktiki, Nauka, Moskva, 1986, pp. 460–469.

[2] A.Degtyarev, A.Lyaletski, M.Morokhovets. Evidence Algorithm and Sequent Logical Inference Search. // Lecture Notes in Artificial Intelligence, 1999, **1705**, pp. 44–61.

[3] V.Glushkov, V.Kostyrko, et al. On a language for description of formal theories. (in Russian) // Teoreticheskaya kibernetika, N3, 1970.

[4] V. Glushkov, K. Vershinin, et al. On a formal language for description of mathematical texts. (in Russian) // Avtomatizatsiya poiska dokazatel'stv teorem v matematike, Institute of Cybernetics, Kiev, 1974.