APPLICATION OF THE FUNCTIONAL PROGRAMMING TOOLS IN THE TASKS OF LANGUAGE AND INTERLANGUAGE STRUCTURES REPRESENTATION

Peter Ermakov
Institute for Informatics Problems,
The Russian Academy of Sciences
IPI RAN
Moscow, Russia
petcazay@gmail.com

Olga Kozhunova
Institute for Informatics Problems,
The Russian Academy of Sciences
IPI RAN
Moscow, Russia
kozhunovka@mail.ru

Abstract. The paper considers issues of formal methods in the tasks of knowledge representation including optimization of formal grammars by means of functional programming languages. One of the possible applications of the formal notation given is illustrated by the task of parallel natural texts analysis and comparison.

Keywords: functional programming tools, language structures representation, parallel texts analysis and comparison

I. Introduction

At present the problem of machine text analysis in natural language is one of the key challenges in the field of information technologies and research. For its solution one involves various interdisciplinary approaches and methods. This caused by the complex character of the research led in the direction: several disciplines are engaged, namely, computer linguistics, artificial intelligence, and formal mathematical methods.

Among linguistic resources that have been developed as a result of such research in the fields of natural language analysis and knowledge acquisition are well-known electronic dictionaries, syntactic parsers, language ontologies, generators of syntactic trees: WordNet, EuroWordNet, Ontolingua, Russian dictionary RusLan, а также Penn Treebank, Ragel, Syntax Definition Formalism, Spirit Parser Framework, SYNTAX, Yacc, etc. This list is quite incomplete, though, it only demostrates variety of approaches and instruments involved in the procedures of natural language mechanisms analysis and modelling.

Thus, automatization of a set of processes within the language (grammar analysis, syntactic and semantic representations, etc.) and of its interaction with human activities (speech recognition, machine translation, parallel texts comparative analysis, and so on) is one of the up-to-date relevant tasks for several disciplines and domains simultaneously. [i, ii].

But the level of sophistication of the existing approaches to language structures and processes representation scales up with the growing demands to the language models. This induces a search of new approaches to language structures representation and hybridization of the well-functioning old methods.

As an example one can study the difficult and long way of universal grammar search to use it in natural language representations which is dramatic for the optimization of machine translation systems, syntactic and semantic text analysis, texts comparative analysis and preset concepts and relations acquisition, etc. One of the principal founders of the approach to modelling of natural language units, relations, and mechanisms of their interaction by means of formal grammars was Noam Chomsky [iii]. Time passed, and many followers of his approach appeared. They started the above mentioned long search. It led them to the gradual enumeration and revision of the existing grammar types, then — to their complication and adaptation to possible applications, and finally — to dissatisfaction from the formal representation they got as a result, and address to statistical methods that prevailed in this field up to the moment. But soon many experts in their independent research found out that statistical methods don't solve the problem of completeness and accuracy of the natural language structures representation. For this reason today more and more specialists address to hybrid methods which are untrivial in construction but more precise accordingly to the applications needed. Besides, validation of their implementation is to be carried out through all the stages of the language representations, its structures analysis, comparison with the existing patterns, and so on [iv].

Thus, as it was mentioned above, formal grammars is a mathematical apparatus meant for the text analysis. The most interest form the perspective of the problem considered attract regular and context-free grammars [iii]. For each of them a relevant analytical engine exists thanks to which parsing might be conducted automatically, i.e. by means of computational procedures. Each analytical engine possesses a set of advantages and disadvantages. For instance, some of the disadvantages are complex structure, infeasibility of the universal engine design and need in its constant rebuild according to a specific application.

This involves many questions, particularly: is it possible to build a universal mechanism of the natural language structures representation using the existing formal methods and techniques? Is there any probability of minimizing work of experts in the procedures of formal representations and natural language structures comparison and their verification?

In an attempt to answer these and other questions this paper suggests an approach which allows representing natural language grammars (as well as formal ones) in a form not demanding an analytical engine design for text parsing and analysis. Rejection of analytical engines use will give us an opportunity to get rid of a set of technical problems associated with their complex implementation. For instance, it is suggested to enhance formal grammars with the functional programming languages and their tools. One of the possible applications of the suggested formal representation is illustrated by an example of the comparison of natural language parallel texts.

II. Formal Grammars and Analytical Engines

Consider a selected at random formal grammar $G = (V_T, V_N, S, P)$, where V_T is a finite set of terminals, V_N is a finite set of nonterminals, start symbol $S \in V_N$ and P is a finite set of productions of a form $\alpha \to \beta : \alpha, \beta \in (V_T \cup V_N) \wedge \alpha \neq \varnothing \wedge \exists v \in V_N : v \in \alpha$.

According to Chomsky classification, formal grammars are divided into four types [iii]:

- unrestricted (type 0)
- context-sensitive (type 1)
- context-free (type 2)
- regular (type 3)

The first two types (type 0 and type 1) have no application due to their complexity, except for the context-sensitive grammars which might be used when analyzing natural languages texts excluding the task of compilators building.

The types 2 and 3 on the contrary have plenty of various applications. For example, context-free grammars are used when describing computer languages syntax. Regular grammars are applied for description of the unary constructions: identifiers, strings, constants, assembler languages, command processors, etc.

An analytical engine for regular grammars is finite state automaton. Equivalence of the regular grammar and finite state automaton is proved in the Kleene theorem [v], which allows assuming the concepts of regular grammar, regular expression, and finite state automaton equivalent.

The field of application of regular grammars in the tasks of natural language structures recognition is rather limited. This is related to the uneasiness of finding a regular expression for describing any of the formal languages, not to mention natural language and its structures. We'd like to note though, that regular expressions is a very convenient tool for analyzing short and highly formalized language constructions.

At present basic instruments of formal and natural languages analysis are context-free grammars. Analytical engine for context-free grammars is a one-sided nondeterministic automaton with stack outer memory. In the most trivial case of such automaton's algorithm implementation, it is characterized by an exponential complexity. But iterative upgrade of the algorithm may lead us to its polynomial time value depending on the length of the input set of symbols and necessary for its analysis [vi].

Among the existing context-free languages one can distinguish a class of deterministic context-free languages, which are interpreted by deterministic automaton with stack outer memory. The principal feature of these languages is their unambiguity: it is proved that one can always build an unambiguous grammar for any deterministic context-free language [vi, vii]. Since the languages are unambiguous, they are most useful when it comes to building compilators [vi].

Moreover, among all the deterministic context-free languages exist such classes of languages which allow building a linear recognizer for them. This is the recognizer which time value related to the time for decision-making about a set of symbols belonging to a language has a linear dependency on the chain length [vi]. Syntactic constructions in the majority of the existing programming languages might be classified as ones of the class mentioned. This aspect is very important when developing up-to-date high-speed compilers.

As a rule, the more complex from the mathematical perspective an analytical engine is, the more challenging is its technical implementation. Case with finite state automata (both with memory and without it) isn't an exception. It's well-known that having formal grammar one can build a automaton accepting it [vii, viii]. But this automaton possesses a set of disadvantages when it comes to its application to natural language grammars. For instance, rules of natural language include a plenty of features of natural language structures which are attributive by their nature. Processing of such transformation rules implemented using finite state automaton may lead to the increase of the automaton states and to growth of amount of the transformation rules when changing the states. Moreover, when implementing the above mentioned transformation rules one needs to create a problem-oriented analytical engine to handle processing of the input set of symbols of the natural language structures and to apply interpreting rules to them.

III. Computational and Functional Grammars

In order to minimize above mentioned challenges the authors suggest an approach which allows representing of natural language grammar rules as functions in mathematical sense, that is $A \xrightarrow{f} B$ or $y = f(x): x \in A, y \in B$. For making it more convenient and clear to handle natural language grammar rules as mathematical functions (i.e. as a transformation which is written as $A \xrightarrow{f} B$) we suggest to use such tools as n-tuples to keep necessary attributive characteristics (for example, gender, singular\plural, etc.). Representation of the rules as functions also gives an opportunity to use instruments of the functional programming to build systems of grammar parsing and analysis and interlingual transfer not loosing efforts to build such an analytical engine as a finite state automaton.

Consider the approach in more detail.

Representation of grammar rules, or transformation rules, as mathematical functions has the following advantages in comparison with formal grammar apparatus:

- Usage of functional programming tools to build systems of transfer immediately;
- Possibility of higher-order function applications.

It's worth noting that by concept "system of transfer" in the paper we understand software implementation of an analytic engine for processing of transformations expressed as functions.

We'll illustrate it by an example: consider a simple sentence in English «I can swim» and its translation into Russian «Я умею плавать».

Examine transformations expressed as mathematical functions. For the example given we use syntax of the functional programming language Erlang:

(1)
trans(i) → я,
trans(can) → мочь,
trans(swim) → плыть;

On applying the functional programming tools, one can split an input sentence into separate words (this mechanism isn't given in detail in the paper). Having applied the above given rules immediately, the following result might be obtained:

(2) я мочь плыть.

Such "translation" is quite unfit to the translation taking into account all the links between natural language structures. However, we'd emphasize that to build such a system of transfer no additional tools were involved except for the system of rules. The whole mechanism of transfer was provided with the functional programming language.

Application of higher-order functions gives an opportunity to pass a function as a parameter to other functions. This allows, for instance, handling the normalization (i.e. putting into normal form – for example, infinitive verb form, noun in singular, etc.) of any natural language structure before transfer.

Using n-tuples as a form of representation of natural language structures enables to generalize attributive characteristics of words and use pattern alternations of such structures in prospect (see an example below). Consider an example of a function which in a case with its argument a noun in singular leaves it without any modifications, but in a case with it in plural, adds an "s" inflexion to the end of the word:

(3)
func({X, noun, singular}) → X,
func({X, noun, plural}) → X ++ «s»;

Such function might be of use, say, when generating text in English.

That's why we'd like to consider such representation form of natural language information as n-tuples in detail. As one can see from the example above, this approach gives an opportunity of arrangement of attributive features and its further use in transformations. Apart from this, using n-tuples for storage of attributive features of language structures enables us to extract functions according to language structures of various levels of abstraction (for instance, a word, a phrase, a sentence, etc). any of such language structures has its own set of attributes, hence there should be functions which have words, phrases, and so on, as their arguments.

At first sight the above given approach to representation of natural language grammar rules apparently generates a huge amount of transformations even for a narrow domain. It is the case. But one should note that since system of transfer built using functional programming tools cannot be considered a finite state automaton, and, thus, doesn't have any states and rules of transformations between the states, so the amount of transformations isn't dramatic. Absolute clearness (absence of the inner state) of the analytical engine gives an opportunity to perform analysis and synthesis of transformations by an expert in the field of computer linguistics in a convenient mode.

The second distinctive feature of the suggested approach is quite technical one. It is essential to design and implement a relevant analytic engine for every information system which tasks correlate with natural language structures analysis based on mathematical apparatus of formal grammars. As it was said above, this task is rather complex and laborious. However, in the case of functional representation of grammar rules as an analytical machine the environment of the functional programming might be used (for instance, Erlang, Haskell) [ix].

But here's a more formal description of the suggested approach.

Firstly, introduce a concept “Computational and Functional Grammar”. Consider a formal grammar $G = (V_T, V_N, S, P)$, where P is $\alpha \to \beta : \alpha, \beta \in (V_T \cup V_N) \wedge \alpha \neq \varnothing \wedge \exists v \in V_N : v \in \alpha$.

Computational and Functional Grammar (CFG) is a form of notation of the above mentioned formal grammar $G_{FC} = (T, f, A)$, where T is a finite set of parametric n-tuples, f – transformation function $f : T \to T$, A – finite set of atoms.

By parametric n-tuple we mean an n-tuple elements of which might be elements of the sets V_N, V_T, A and special symbol «_».

By atom we assume any unambiguously determined identificator such that $A \cap (V_N \cup V_T) = \varnothing$. Atoms are meant for setting attributive characteristics of natural language words (gender, singular/plural, case, etc.). They are specific instrument for simplifying of natural language structures analysis in particular.

We'd like to emphasize that atoms and atomic structures are included into CFG description. Thus, all possible attributive characteristics of language structures are defined in the Grammar. That is, CFG is a formalism sufficient for natural language structures analysis, and there is no necessity for using any of other mathematical tools in addition to it.

Function f in Computational and Functional Grammars is defined in the table form similar with the Backus-Naur form which is used for setting the rules in context-free grammars.

The symbol «_» is suggested to denote an n-tuple element which value one may ignore when defining the transformation function. Thus, illustrate the sense of the special symbol «_» by an example.

Consider a set of atoms *A* = {*noun, verb, plural, singular, ok, not ok*} and a function defined as follows:

(4)
f({noun, _}) → {ok},
f({verb, _}) → {not ok};

Then f({noun, plural}) = {ok} and f({noun, singular}) = {ok}, i.e. the function’s value will be {ok} regardless of noun is in singular or in plural.

To illustrate the difference in grammar rules representations by example, we’ll consider a formal grammar G = ({a, the, dog, cat, chased}, {<S>, <NP>, <VP>, <N>, <V>, <DET>}, <S>, P) where P:

(5)
<S> ::= <NP> <VP>,
<NP> ::= <DET> <N>,
<VP> ::= <V> <NP>,
<DET> ::= a | the,
<N> ::= dog | cat,
<V> ::= chased

where $< S >, < NP >, < VP >, < DET > \in V_N$ and $a, the, dog, cat, chased \in V_T$.

In case of functional representation the above defined grammar will be expressed as follows:

$$G_{FC} = (\{\langle S\rangle, \langle NP\rangle, \langle VP\rangle, \langle N\rangle, \langle V\rangle, \langle DET\rangle, a, the, dog, cat, chased\}, f, \varnothing)$$

where f:

(6)
f(<DET>) -> a,
f(<DET>) -> the,
f(<N>) -> dog,
f(<N>) -> cat,
f(<V>) -> chased,
f(<NP>) -> f(<DET>) ++ f(<N>),
f(<VP>) -> f(<V>) ++ f(<NP>),
f(<S>) -> f(<NP>) ++ f(<VP>);

Symbol «++» denotes an operation of concatenation.

One should pay attention that in the considered example the set of atoms A is empty. This points to the fact that such grammar doesn’t take into account attributive characteristics of words and language constructions. One may also notice that the set T in the example above is just a joint alphabet of terminals and nonterminals of the initial grammar G.

IV. Task of parallel texts analysis and comparison

At the contemporary stage of design and development of natural language processing systems the main emphasis is merged towards creation of parallel texts (i.e. texts in several languages equivalent by their contents and representation forms) analysis techniques. It generates a set of tasks concerned with their adequate interpretation and application, above all, those are tasks of machine translation and knowledge processing [x, xi, xii].

For that reason we demonstrate capabilities of the functional programming tools when applied in the task of parallel text analysis and comparison, including the task of interlanguage structures transfer from one language into the other [xiii]. As an example we consider texts of the patent claims (in chemical technologies) in German and English respectively:

(7)

- ***Claim in German****: Verfahren zur Epoxidierung einer organischen Verbindung mit wenigstens einer C C-Doppelbindung mit Wasserstoffperoxid in Gegenwart wenigstens einer katalytisch aktiven Verbindung und wenigstens eines Lösungsmittels, dadurch gekennzeichnet, dass ein Produktgemisch umfassend a-Hydroperoxyalkohole unter Einsatz wenigstens eines Reduktionsmittels reduziert wird.*

- ***Claim in English:*** *A process for the epoxidation of an organic compound having at least one C-C double bond by means of hydrogen peroxide in the presence of at least one catalytically active compound and at least one solvent, wherein a product mixture comprising [alpha]-hydroperoxyalcohols is reduced using at least one reducing agent.*

When consistently comparing the claims given (for instance, for the sake of confirmation of the patent information and data mining in chemistry) the following transformations were detected:

(8)

(a) Verfahren zur Epoxidierung → A process for the epoxidation

N [verb, nom, neutr, sg] + Prep [zu+der, dat, comp, fem, sg] + N [dat, fem, sg] → Art [indef, sg] + N [com, sg] + Prep + Art [def, 0] + N [com,sg]

(b) ein Produktgemisch → a product mixture

Art [indef, masc, nom, sg] + N [comp, nom, neutr, sg] → Art [indef, sg] + N [com, sg] + N [com,sg]

(c) dadurch gekennzeichnet → wherein

Pron + Part [II f, masc, sg] → Adv

The above given transformations are described by means of primitive language markup and a set of grammar attributes. Thus, in case (b) (example (8)) the transfer of German phrase structure «*ein Produktgemisch*» into English one «*a product mixture*» is described as a modification of an article (Art) with attributes «*indef, masc, nom, sg*» (that is indefinite, masculinum, nominative case) and a compound noun N from the left part of the transformation into a phrase structure in English with an article and its attributes «*indef, sg*» and two nouns in the right part of the transformation.

However, this technique of transformations notation (in the example above) possesses some disadvantages, namely, awkwardness of the rule itself and need to interpret it with the help of specifically designed analytic engine.

But using functional programming tools (which are the instruments of language structures representation in the case as well) may give one an opportunity to write down the rule (b) from the example (8), as follows:

(9)

(b) *ein Produktgemisch → a product mixture*

v({«ein»,art,indef,masc,nom,sg}) → «a»,
v({«Productgemisch»,noun,comp,nom,neutr,sg}) → «product mixture»;
fgerman-english(
{X1,art,indef,masc,nom,sg},
{X2,noun,comp,nom,neutr,sg}) →
v({X1,art,indef,masc,nom,sg}) ++
v({X2,noun,comp,nom,neutr,sg});

V. Conclusion

In the paper a new approach to natural language grammar representation as functions in mathematical sense is considered. Also opportunities of applying functional programming tools to building systems of transfer are demonstrated. Practical application of the approach is viewed from the perspective of parallel texts analysis and comparison (texts from patent and scientific fields).

Further research within the approach and associated tasks may be conducted in the following directions:

- Customizing of the existing representations of the natural language grammars to functional form;
- Creation of problem-oriented system of functional programming to make handling of natural language rules more convenient;
- Enhancement of functional programming tools taking into account needs and tasks of computer linguistics.

---

i . Козеренко Е.Б. Лингвистическое моделирование для систем машинного перевода и обработки знаний // Информатика и ее применения, №1, том 1. – М.: Торус, 2007. – С.54-65.

ii . Козеренко Е.Б. Глагольно-именные трансформации при англо-русском машинном переводе // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2007» / Под ред. Л.Л. Иомдина, Н.И. Лауфер, А.С. Нариньяни, В.П. Селегея. - М.: Изд-во РГГУ, 2007. – С. 286-294.

iii Chomsky N. Syntactic Structures. — The Hague: Mouton, 1957.

iv . Jacobs, Roderick A. and Peter S. Rosenbaum. English Transformational Grammar. Blaisdell, 1968.

v Клини С.К. Математическая логика. -М.: изд-во Мир, [1967]1973.

vi Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. Введение в теорию автоматов, языков и вычислений = Introduction to Automata Theory, Languages, and Computation. — М.: «Вильямс», 2002. — С. 528.

vii . А. В. Гладкий, А. Я. Диковский, “Теория формальных грамматик” / Итоги науки и техн. Сер. Теор. вероятн. Мат. стат. Теор. кибернет., 10. – М.: ВИНИТИ, 1972. – С. 107–142.

viii . Кобринский Н.Е., Трахтенброт Б.А. Введение в теорию конечных автоматов. – М.: Гос. издательство физ.-мат. литературы, 1962. – 405 с.

[ix] http://erlang.org, http://haskell.org

[x] . Козеренко Е.Б. Проблема эквивалентности языковых структур при переводе и семантическом выравнивании параллельных текстов // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2006» / Под ред. Л.Л. Иомдина, Н.И. Лауфер, А.С. Нариньяни, В.П. Селегея. - М.: Изд-во РГГУ, 2006. – С.252-258.

[xi] . Nivre J., Boguslavski I., Iomdin L. Parcing the SynTagRus Treebank of Russian \ Proceedings of the International Conference COLING’2008, Manchester, UK, 2008.

[xii] . Macken L., Lefever E., Hoste V. Linguistically-based sub-sentential alignment for terminology extraction from a bilingual automotive corpus \ Proceedings of the International Conference COLING’2008, Manchester, UK, 2008.

[xiii] .Кожунова О.С. Выявление номинализованных конструкций в параллельных текстах патентных документов на русском и немецком языках // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2009» / Под ред. Л.Л. Иомдина, Н.И. Лауфер, А.С. Нариньяни, В.П. Селегея. - М.: Изд-во РГГУ, 2009. – С.185-191.