МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ХАРЧОВИХ ТЕХНОЛОГІЙ

ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ І ТЕХНОЛОГІЙ «ІНДУСТРІЯ 4.0» ІМ. П.Н. ПЛАТОНОВА

XII МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2019

INFORMATION TECHNOLOGIES AND AUTOMATION– 2019

Збірник доповідей

Частина I

Одеса,
17-18 жовтня 2019

Секція 1

Наукові напрямки:

Комп'ютерні телекомунікаційні мережі та технології

Математичне моделювання та інформаційні технології

Список
скорочень організацій, представники яких взяли участь у конференції

Таблиця 1

Скорочення	Повна назва організації	Місто	Країна
BNTU	Belarusian National Technical University	Minsk	Belarus
CAFU	CRIAME of Armed Forces of Ukraine	Kyiv	Ukraine
DMTSAU	Dmutro Motornyi Tavria State Agrotechnological University	Melitopol	Україна
DNU	Vasyl' Stus Donetsk National University	Вінниця	Україна
EKSTU	East Kazakhstan State Technical University D. Serikbayev	Ust-Kamenogorsk	Kazakhstan
IAEl SB RAS	Institute of Automation and Electrometry of the Siberian Branch of the Russian Academy of Sciences	Novosibirsk	Russia
IRTC IT&S NAS AND MES	International Research and Training Center for Information Technologies and Systems of the National Academy of Sciences (NAS) of Ukraine and Ministry of Education and Science (MES) of Ukraine	Kyiv	Ukraine
KGES	Kharkiv general education school	Kharkov	Україна
LPNUU	Lviv Polytechnic National University	Lviv	Ukraine
NTU “KhPI”	National Technical University "Kharkiv Polytechnic Institute"	Kharkov	Україна
NTU «KPI»	National Technical University "Igor Sikorsky Kyiv Polytechnic Institute"	Kyiv	Ukraine
NU «OMA»	Національний університет «Одеська морська академія»	Одеса	Україна
NULESU	National University of Life and Environmental Sciences of Ukraine	Kyiv	Ukraine
NUOS	NATIONAL UNIVERSITY OF SHIPBUILDIN NAMED BY ADM. MAKAROV	Nikolaev	Ukraine
ONAFT	Odessa National Academy of Food Technologies	Odessa	Ukraine
ONU	Odessa I.I.Mechnikov National University	Odessa	Ukraine
SSU	Sukhumi State University	Sukhumi	Georgia
VNTU	Vinnytsia National Technical University	Vinnytsia	Ukraine
БНТУ	Белорусский национальный технический университет	Минск	Белоруссия
ВНТУ	Вінницький національний технічний університет	Вінниця	Україна
ДВНЗ «КНУ»	Державний вищий навчальний заклад «Криворізький національний університет»	Кривий Ріг	Україна
ДонНТУ	Донецький національний технічний університет	Покровськ	Україна
ІК НАН України	Інститут кібернетики імені В.М. Глушкова НАН України	Київ	Україна
НТУ «ХПІ»	Национальный технический университет "Харьковский политехнический институт"	Харків	Україна
НТУУ "КПІ"	Національний технічний університет «Київський політехнічний інститут» імені Ігоря Сікорського"	Київ	Україна
НУ «ЛП»	Національний університет «Львівська політехніка»	Львів	Україна
ОДАТРЯ	Одеська державна академія технічного регулювання та якості	Одеса	Україна

Продовження таблиці 1

Скорочення	Повна назва організації	Місто	Країна
ОНАЗ	Одеська національна Академія зв'язку ім. О.С. Попова	Одеса	Україна
ОНАПТ	Одесская национальная академия пищевых технологий	Одесса	Украина
ОНАХТ	Одеська національна академія піщевіх технологій	Одеса	Україна
ОНПУ	Одеський національний політехнічний університет	Одеса	Україна
ОНУ	Одеський національний університет імені І. І. Мечникова	Одеса	Україна
ОТК ОНАХТ	Одеський технічний коледж Одеської національної академії харчових технологій	Одеса	Україна
ПНПУ	Південноукраїнський національний педагогічний університет ім. К.Д. Ушинського	Одеса	Україна
ХНУРЕ	Харківський національний університет радіоелектроніки	Харків	Україна
ХРТК	Харківський радіотехнічний технікум	Харків	Україна
ЦНДІ ОВТ ЗС України	Центральний науково-дослідний інститут озброєння та військової техніки Збройних Сил України	Київ	Україна
ЮНПУ	Южноукраинский национальный педагогический университет им. К.Д.Ушинского	Одесса	Украина

UDC 004.05:004.4

KOMLEVA N. O., POPOV S. S.
ONPU, ICS (Odessa)

QUALITY ATTRIBUTES OF FORMAL GRAMMARS AND LANGUAGES IN TRANSLATOR ENGINEERING

THE PAPER PROPOSES A SYSTEM OF QUALITY ATTRIBUTES FOR FORMAL GRAMMARS AND LANGUAGES THAT ARE USED TO BUILD THE COMPILER IN ORDER TO TRANSLATE A PROGRAM OR TRANSLATE SENTENCES FROM A NATURAL LANGUAGE. FOUR CATEGORIES OF QUALITY ATTRIBUTES ARE PROPOSED: FOR GRAMMAR SYMBOLS, FOR TOKENS, FOR THE INITIAL SYMBOL, FOR GRAMMAR RULES.

Introduction. A translator is a program that translates a program in a programming language into a program consisting of machine instructions. The translator usually also performs error diagnostics, generates identifier tables, etc. The language in which the input program is presented is called the source language, and the program itself is called the source code. The output language is called the target language or object code.

In the general case, the concept of translation refers not only to programming languages, but also to other languages – both formal computer and natural (Ukrainian, Russian, English, etc.).

Description of programming languages and natural languages is based on the theory of formal languages. This theory is the foundation for organizing syntactic parsing. According to the theory of formal languages, any language is a set of chains of finite length in a given alphabet. The generation mechanism allows you to describe a language using a system of rules called a grammar. Language chains (sentences) are built in accordance with these rules. The advantage of defining a language using grammars is that operations during syntactic parsing can be formalized and simplified [1].

In order to make it possible to define a language using a grammar, the rules of this grammar must satisfy certain requirements. Failure to comply with these requirements leads to problems during the operation of certain stages of the translator.

The aim of work. The aim of the work is formation of sets of quality attributes for formal grammars and languages to translate a program or sentences of a natural language.

Research methods. In order to formally define a language, it is necessary to specify some components of this language – the alphabet (the set of valid language characters). Individual characters can be combined into words (tokens). The vocabulary of the language determines the vocabulary of the language. Words can be combined in a sentence (operators, instructions). The correct construction of sentences is determined by the syntax of the language. The semantic content of sentences is determined by the semantics of the language [2].

The compilation process consists of the following steps:

1. Lexical analysis. At this point, the sequence of characters in the source file is converted to a sequence of tokens (language units).
2. Syntactic parsing. The sequence of tokens is converted to a parse tree.
3. Semantic analysis. The parse tree is processed in order to establish its semantics (meaning).
4. Optimization. Redundant constructions are removed and the code is simplified while retaining its meaning.
5. Code generation. The code in the target language is generated from the intermediate representation.

To build a compiler, an unambiguous and accurate definition of the input and output languages is required. Such a task involves determining the rules for constructing permissible constructions (expressions) of a language. Many of these rules are called language syntax. In addition, the assignment should include a description of the purpose and meaning of each language construct. Such a description is called semantics of the language. To construct accurate and unambiguous descriptions, the abstraction method is used, which provides for the selection of the most significant properties of the object in question and the omission of properties less significant for the case under consideration. For example, when building a model of input languages, you can consider the source text as a sequence of characters built according to certain rules, distracting from the nature of the style of the characters and their location on the sheet. Mathematical models using representations of texts in the form of chains of characters are called formal languages and grammars.

A finite set of indivisible characters is called a dictionary or alphabet, and the characters included in the set are called letters of the alphabet. A sequence of letters of an alphabet is called a word or string in that alphabet. The number of letters in a word is called its length.

The formal grammar of G, generates a language, is the following set of four objects: G = {VT, VN, S, P}, where V_T is the terminal alphabet (dictionary) the letters of this alphabet are called terminal symbols; from which chains of language are generated that are generated by grammar; V_N – non-terminal, auxiliary alphabet (dictionary) letters of this alphabet are used in the construction of chains; they can be included in intermediate chains, but should not be included in the result of generation; S is the initial character of the grammar that belongs to V_N, S is the most general language construct from which any string of language can be derived; P is the set of inference rules, or generating rules of the form a->b, where a and b are chains constructed from the letters of the alphabet V_T and V_N, which is called the complete alphabet (dictionary) of the grammar G. The set of finite chains of the terminal alphabet V_T of the grammar G derived from the initial symbol S is called a language, is generated by the grammar G, and is denoted as L (G).

Specifying a programming language, we must at a minimum determine: 1) the set of characters that can be used to write the correct programs; 2) many correct programs; 3) the "meaning" of each correct program.

Determining the set of valid characters is required for the first part of the translator – the lexical analyzer. The block of characters of the defined alphabet is fed to the input of this block. However, some character combinations are considered as single objects. The job of a lexical analyzer is to group together some specific language characters into a single syntactic object called a token. Which objects to consider as tokens depends on the definition of the programming language.

A token is a chain of characters of a language with which we associate a lexical structure consisting of a pair of the form <type of token, some data>. The first component of the pair is a syntactic category, such as a "constant" or "identifier", and the second is a pointer with a cell address that stores information about that particular token. The sequence of such pairs is used hereafter to operate the parser.

The task of programming language syntax is a non-trivial task. Although there are no universally applicable methods, there is a concept of context-free grammar in language theory that can be used to develop the necessary description. When specifying a programming language, a class of valid programs is determined using grammatical rules.

Syntax is used to determine and implement transformations of an input program, or, in other words, a chain of speech. In this case, the designated tree is a syntax tree, which can be conveniently provided to the input program. The process of finding the syntactic structure of a given sentence is called syntactic parsing. The syntactic structure of the sentence helps to understand the relationships between different parts of the sentence. Thus, to be able to describe a new input language, it is necessary to develop grammatical rules that are executed in accordance with certain rules.

Results and discussion. After the basic principles are set out and the notation is introduced, quality attributes for formal grammars and languages can be systematized. For quality attributes, it is necessary to indicate the required values, the observance of which will ensure the correctness of the grammar, and also describe possible errors and ways to eliminate them.

Quality criteria for grammar symbols.

1. The set of terminal symbols V_T should not be empty. In the case of developing a grammar for translating a program written in a particular programming language, the V_T set should include all the characters of the language that any programmer wants to use when writing any program in a given language. When developing a grammar for translating natural language sentences, the V_T set should include all letters, punctuation, and special characters.

If the set V_T is given empty, this means that the grammar describes a language containing only one empty sentence. There is no practical value for such language.

If the V_T set does not contain all the required characters of the language, it is necessary to consider the possibility of its addition.

2. The set of non-terminals V_N characters should not be redundant. It is required to evaluate the need for the introduction of the next non-terminal symbol. Only when the benefits of ease of use exceed the inconvenience of increasing the volume of the grammar, is a non-terminal symbol added to the grammar. The development by a mechanism for evaluating the feasibility of introducing a non-terminal symbol into the grammar is not included in the scope of this article.

The V_N set cannot be empty. The V_N set can consist of one character (the initial character of a grammar) only for simple grammars that do not contain a large set of syntactic constructions.

No element of the set V_N can coincide with any element of the set V_T, that is, the result of the intersection of these two sets is an empty set.

If the set V_N is empty, it means the inability to output any sentence of the language. There is no practical value for such language.

If the set V_N is redundant, then this does not lead to translation errors, but requires additional resources. Redundant elements of the set V_N can be excluded subject to the equivalence conditions of the original and resulting grammars.

If any element of the set V_N coincides with any element of the set V_T, this corresponds to a fatal situation. Such errors should be fixed. They do not allow the use of grammar to display any sentence of the language.

3. Any grammar character must belong to either the set of terminal symbols V_T or the set of non-terminal symbols V_N.

If a symbol is found in the grammar that does not belong to any of these sets, such a situation is fatal.

4. In the general case, not all elements of the sets V_T and V_N are involved in the derivation of a language sentence. In the simplest case, one non-terminal character is involved in the output – the initial grammar character and one terminal character.

Quality criteria for grammar tokens.

1. All tokens must be denoted by unique sequences of characters.

If two different tokens have the same designation, this leads to semantic errors.

2. It should be possible to distinguish grammar tokens. This can be achieved by introducing special rules or by separating grammar tokens with special characters.

If two or more unshared tokens are present in any sentence, then two kinds of errors may occur. An error of the first kind occurs when the sequence of characters of unshared tokens completely coincide with the sequence of characters of some other token. This may cause an error at the parsing stage. An error of the second type occurs if the sequence of characters of unshared tokens does not coincide with any other token. This results in a fatal syntax error.

3. The grammar should contain the rules for the formation of temporary tokens (identifiers). These rules may include valid character sets based on their relative position, maximum token length, etc.

Violation of the rules for the formation of lexemes leads to lexical errors.

Quality criteria for the initial symbol of a grammar.

1. The initial symbol of the grammar must belong to the set of non-terminal symbols.

If it does not belong to the set, then a fatal error occurs.

2. The initial grammar character must be producing. This means that in the grammar there are one or more rules by which the sentences of the language are deduced from the initial character.

If the initial character of the grammar is not productive, then the grammar describes an empty language.

Quality criteria for grammar rules.

1. Grammar should be reversible. This means that two or more rules are not allowed according to which different non-terminal characters produce the same language sentences.

The presence of irreversible rules does not allow formalizing the parsing algorithm.

2. All non-terminal symbols of the grammar must be generative, that is, one or more conclusions of sentences from each non-terminal symbol must exist.

If there are non-producing non-terminal symbols in the grammar, then they should be eliminated.

3. All characters of the grammar, except the initial character, must be reachable. This means that in the process of outputting a sentence, a symbol can become part of this sentence.

If unreachable characters are present in the grammar, they should be eliminated in compliance with the equivalence rules for the source and resulting grammars.

4. Grammar rules should not have infinite recursions.

Having infinite recursion will result in a syntax error when using this rule.

Strict observance of all these criteria is a prerequisite for building a grammar designed to translate a program or sentences of a natural language.

Conclusion. The article discusses the principles of the translator, allocated its main stages – lexical, syntactic and semantic analysis, optimization, code generation. It is shown that for the effective operation of the translator it is advisable to use formal grammars. The basic requirements for these grammars are formulated, quality criteria for symbols, tokens, the initial symbol and grammar rules are set and described.

REFERENCES

[1] A. Meduna “Elements of Compiler Design”, *CRC Press*, 2017.
[2] P. M. Lewis “Compiler Design Theory”, *Addison-Wesley*, 2006.

XII МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2019

INFORMATION TECHNOLOGIES AND AUTOMATION– 2019

ОДЕСА
17– 18 ЖОВТНЯ, 2019

Збірник включає доповіді учасників XII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2019»

Редакційна колегія: Котлик С.В., Хобін В.А., Плотніков В.М.

Комп'ютерний набір і верстка: Соколова О.П.

Відповідальний за випуск: Котлик С.В.