Formal Tools for Language Specification

How to read this book?

The republication of this book, after 40 years since its first appearance, aims, among other things, to highlight the resilience of the evolution process of human thought through the creation of the computer and its use in problem-solving. This process has two stages: the first stage began with the use of the computer in engineering and mathematics through the development of technology for *integration of the human problem-solving process into the machine's calculation process. The second stage* began with the development of the concept of artificial intelligence, *by integrating the machine's calculation process, i.e.,* the technology of using the computer, *into the human brain's problem-solving process*. The first stage is characterized by the development of the term called Information Technology, abbreviated IT, while the second stage aims to develop and establish the term Artificial Intelligence, abbreviated AI. In 1983, when this book was first published, IT was in its infancy, so the book was dedicated to discussing the formal tools that facilitated the emergence and establishment of the term IT. Now, after 40 years, the book is still relevant, but the concept of AI is becoming increasingly prominent. Therefore, the republication of a revised edition of this book is justified, but it requires new elements to justify the current direction of research in the field of computing technology. We have introduced these elements by adding the preamble and chapter 8.

The preamble of the book shows the need to redirect research in the field of IT towards simplifying the methodology of using the

computer so that it (the computer) becomes an assistant to the human brain. This can be achieved by developing a methodology for using the computer employing the natural language of the user. Chapter 8 of the book titled **The Computer as an Instrument of the Brain. A project with unlimited potential**

presents an implementation model of the methodology postulated in the preamble. To show the continuity of the ideas discussed in the book we reproduce in this edition the abstract of the book published in 1983.

Chapter 8 begins with a debate justifying the need for developing a new methodology for computer usage in handling abstractions in the human thinking process, as it seems to be executed by the human brain. The new methodology for computer usage is based on employing the user's natural language in problem-solving processes, similar to how all other methodologies for human tool usage, developed to enhance human capability in handling physical elements (such as cars for motion, glasses for vision, medications for pathogens, etc.), utilize natural language. Since all other technologies for human tool usage employ natural language, the computer should be used in the same way. Thus, a methodology for computer usage employing natural language must be developed, parallel to the current methodology, in which the computer is used through programming languages. The use of programming languages was necessary because natural languages are ambiguous and therefore cannot be accurately translated into machine language.

Our observation is that the ambiguity of natural language is transparent to the computer user. This is because the computer user employs a language specific to the domain in which she is using the computer. We have named this language the Domain Algorithmic Language, or DAL for short. Manipulating domain terms also requires a formal representation of domain knowledge. This is achieved by representing the domain of computer usage using an ontology of the domain, where knowledge is stored in the form of pairs (Term, Meaning), where Term is the term from DAL (the natural language used) and Meaning is a Web Service

representing the knowledge denoted by Term. Thus, the problem domain can be maintained in a file accessible to both the computer and its user.

The process of representing the computer application domain as an ontology is actually a formalization process of the domain, discussed in Chapter 8 under the name Computational Emancipation of the Domain. Therefore, there is a need for a virtual machine whose memory is the domain ontology and whose instructions are Web Services. The brain can handle this machine by executing algorithms on the internet, thus making the computer essentially an assistant to the brain. Chapter 8 illustrates the development of this computer usage methodology, describing its implementation for the domain of computer usage in learning algebra in a high school class.

The reader's question of how to read this 600 pages book has several answers. We, in view with the content of the book and the potential use of the material presented, give the following answer. A reader simply interested in using the computer in the problem-solving process, ignoring the myriad of software, known as Software Tools, used for transforming the problem and its solution into executable programs, should read the preamble and chapter 8.

If the reader is interested in computer science, that is, in the science of problem-solving with the computer, then she should read and understand chapter 1 first. This chapter discusses a global concept of language that is intended to be used during communication between different communicators. To our knowledge, this concept is unique and original. Its merit is that it represents a synthesis of the language concept used in the field of computer science, and that both human natural language and various other artificial languages, such as programming languages, are seen in this chapter as particular cases of the language concept

discussed in this chapter. Therefore, the language concept presented in chapter 1 is robust and provides a solid foundation for manipulating language as a computational object in various aspects of this problem.

Depending on the reader's interest, he can continue by addressing either chapter 2 or chapter 6. Chapter 2 discusses the concept of grammar as a mechanism for specifying a language by generating its elements, as well as the Turing Machine as a mechanism for specifying a language by recognizing its elements. Algorithmically speaking, grammar is seen as an algorithm for generating the elements of a language through a rewriting process of a given syntactic category using rewriting rules similar to the rules for rewriting sentences of a natural language from the symbol called a *phrase* using syntactic categories such as noun, verb, adjective, adverb, etc. On the other hand, the Turing Machine appears as an algorithm for recognizing the strings of a language using recognition rules that reduce these strings using simple substitution operations of substrings of a string similar to the simple calculation rules. Furthermore, depending on his interest, the reader can continue with chapter 3, which discusses algorithms for analyzing a language, based on the grammar that specifies that language. These algorithms constitute the foundation of the current technology of computing techniques. The data of these algorithms are:

- A string to be analyzed.
- A set of grammar rules of the form LeftPart = RightPart, where LeftPart is a symbol called a syntactic category, and RightPart is a combination of syntactic categories and given strings called terminals.
- A syntactic category called an axiom or start symbol.

The hypothesis of these algorithms is that the string to be analyzed is an element of the language specified by the given specification rules. This means that the string to be analyzed can be regenerated using the given grammar rules. The regeneration process consists of constructing the derivation tree of the analyzed string from a syntactic category C identified by a specification rule C = RightPart. The symbol C can be replaced with RightPart, or the substring identified by RightPart can be replaced with the symbol C. Thus, the algorithm for constructing the derivation tree of the given string consumes this string sequentially, symbol by symbol, reading it from left to right or from right to left, while building its derivation tree. This operation can be driven by one of two strategies: top-down, starting with the root of this tree marked by the axiom, or bottom-up, starting with its frontier of the tree whose nodes are marked by the words (i.e., the symbols) components of the given string. Because these two strategies are uniquely determined by the data of the problem, the process of constructing the derivation tree of the given string is well-defined, and the string thus generated is obtained by concatenating the leaves of this tree.

If the reader is more interested in approaching the language to be analyzed based on algorithms for recognizing its elements of the Turing Machine type, then the reader should continue to chapter 4.

Chapter 4 is dedicated to discussing algorithms similar to those discussed in Chapter 3 but based on a computational mechanism rooted in the Turing Machine. The basic operation of these algorithms is the process known as pattern matching in strings. This type of algorithms is less discussed in literature. Therefore this may be a fertile field for research and development for students and researchers. The author and his

students at the University of Iowa, Iowa City, USA, have initiated a methodology for software development using similar algorithms whose implementation is based on "pattern matching". The advantage of these algorithms is that they are more efficient because they can utilize all the rules in the specification package and all symbols of the given string in parallel, eliminating the assumption of sequential operation. Of course, this is an original research in which the rules in the package specifying the language are ordered into hierarchy classes, as explained in Chapter 4. The information on which these algorithms are based is known as contexts and non-contexts. Contexts are pairs of strings that if they include the RightPart of a specification rule then this RightPart specifies the LeftPart of this rule. Non-contexts are pairs of strings that if they include the RightPart of a specification rule then this RightPart does not specify the LeftPart of this rule. Chapter 5 of the book contains a collection of algorithms used to compute the contexts and non-contexts of the specification rules.

Chapter 6 of the book contains an original methodology for manipulating abstractions of computation. The fundamental idea of this methodology is considering abstractions of computation as a reflection of the physical world. The physical universe is considered as an open list of hierarchical models, where a physical model is constructed by taking another physical model as its base. The current algebraic model for manipulating abstractions does not distinguish hierarchical levels of models. That is, the current algebraic structures used as models of abstractions of computation are not hierarchized like the physical ones. The connection between the algebras used as models for manipulating abstractions is a functional connection, which preserves algebraic operations, without considering the representation of a model in terms of another given model. Heterogeneous algebra models generalize the functional relationship between algebras but do not discuss the representation of elements of a heterogeneous algebra as elements of another heterogeneous algebra, which does not necessarily belong to the same class of similarity. However, algorithms used by different computer systems, such as programming language translation algorithms, require such a relationship between manipulated abstractions of computation.

The methodology of manipulating abstractions discussed in Chapter 6 of this book lays the foundation for such a relationship between the models of abstractions manipulated by computer science, in which abstractions are systematically constructed based on other abstractions, as the physical universe is built based on a hierarchic model. For example, to construct a house as a physical model, other physical models such as the foundation, walls, roof, etc., are used, each of these models being treated similarly. The fundamental relationship underlying the hierarchy of physical models is abstraction. This means that to build a physical model uing other physical models as components, through abstraction, all properties of the component used are ignored except for being a component of the built model. To use a physical model for a specific purpose, the properties of the model to be used are remembered in the context in which it is used. These forgetting and remembering relationships are formalized in this book using the Heterogeneous Algebraic Structures (HAS) hierarchy. In Chapters 6 and 7, the HAS hierarchy is used as the basis for constructing abstract computation models. The properties of these models are studied using derived operations from heterogeneous algebras, represented in computation abstractions using macro-operations from programming languages. This study, being a premiere, offers an open field for both theoretical research and practical applications. In particular, the use of the computer as a tool of the human brain becomes the basis for the unlimited evolution of software in the process of human evolution.

Prof. Teodor Rus, August 29, 2023

4