Mathematical Foundations of Computer Science 1989

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

379

A. Kreczmar G. Mirkowska (Eds.)

MFCS '89

Mathematical Foundations of Computer Science 1989

Porąbka-Kozubnik, Poland
August 28 – September 1, 1989
Proceedings

Springer-Verlag
Berlin Heidelberg New York London Paris Tokyo Hong Kong

Volume Editors

Antoni Kreczmar
Grażyna Mirkowska
University of Warsaw, Institute of Informatics
PkiN, room 850
PL–00–901 Warsaw, Poland

CR Subject Classification (1987): D.2.4, F.1.1, F.2–4, G.2.2, G.4, I.2.2

ISBN 3-540-51486-4 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-51486-4 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210 – Printed on acid-free paper

PREFACE

The present volume contains papers selected for presentation at the 14th Symposium on Mathematical Foundations of Computer Science, MFCS '89, held in Porąbka-Kozubnik, Poland, August 28 – September 1, 1989.

The symposium is the fourteenth in the series of international meetings which have taken place since 1972. The first meeting was organized in Jabłonna, Poland and aimed at attracting computer scientists from West and East, both terms being understood in as general a sense as possible. The symposium focused the attention of its participants on theoretical issues of computer science. The next meetings were organized alternately in Czechoslovakia and Poland till 1981 and then every other year in Czechoslovakia only. The present conference aims at resuscitating the long tradition of alternating the organization of MFCS between Poland and Czechoslovakia.

Principal areas of the MFCS '89 conference include: logics of programs, parallel and distributed computations, deductive databases, automata and formal languages, algorithms and data structures, software specification and validity, complexity and computability theory.

The Proceedings include invited papers and communications. The latter have been selected by the International Program Committee from 102 submitted papers.

The Program Committee of MFCS '89 consists of : A. Arnold (Bordeaux), A. Blikle (Warsaw), J. de Bakker (Amsterdam), M. Chytil (Prague), P. van Emde Boas (Amsterdam), R. Freivalds (Riga), H. Genrich (Bonn), J. Gruška (Bratislava), H. Langmaack (Kiel), B. Monien (Paderborn), P. Mosses (Aarhus), G. Mirkowska (Warsaw), M. Protasi (Pisa), A. Salwicki (Warsaw), W. Wechler (Dresden).

The editors wish to thank all the members of the Program Committee for their meritorious work in evaluating the submitted papers. We would also like to thank all referees who assisted the members of the Program Committee:

I. Aalbersberg, E. H. Aarts, D. Ackermann, E. Astesiano, P. Atzeni, L. Banachowski, D. Bini, Ch. Blaue, J. Błażewicz, F. Boer, A. Borzyszkowski, G. Boudol, H. D. Burkhard, K-H. Buth, B. Buth, A. Caprani, I. Castellani, B. Courcelle, B. Chlebus, L. Czaja, P. Degano, K. Diks, P. Ďuriš, E. Fachini, R. Feldmann, G. Gambosi, W. Goerigk, U. Goltz, M. Grabowski, J. F. Groote, M. Hass, L. Holenderski, H. J. Hoogeboom, H. Hungar, J. Hromkovič, P. Kandzia, B. Kanger, A. Kelemenová, F. Kluźniak, J. N. Kok, B. Konikowska, V. Koubek, R. Koymans, J. Knoop, I. Kramosil, A. Kreczmar, M. Křivánek, A. Marchetti-Spaccamela, M. Krivanek, M. Lenzerini, M. Liśkiewicz, A. Litwiniuk, G. Longo, K. Loryś, F. Luccio, R. Lueling, W. Lukaszawicz, G. Mascari, A. Masini, J. Matoušek, A. Mazurkiewicz, B. Mayoh, Y. Métivier, E. Meyer, J. Milewski, U. Montanari, A. W. Mostowski, H. Müller, P. Mysliwietz, M. Napoli, M. Nielsen, R. De Nicola, D. Niwiński, E. Ochmanski, E. Orlowska, R. Orsini, L. Pacholski, F. Parisi-Presicce, W. Pawłowski, W. Penczek, H. P. Pfahler, M. Piotrow, W. Preilowski, I. Privare, H. Reichel, W. Reisig, L. Ricci, L. Rudak, J. J. Rutten, P. Ružička, M. Ryćko, W. Rytter, A. Salibra, G. Sénizergues, E. M. Schmidt, F. Simon, S. Skyum, M. Ślusarek, E. Smith, L. Stapp, M. Steinby, J. Steiner, P. Štěpánek, I. H. Sudborough, A. Szałas, D. Szczepańska, O. S vykora, M. Sysło, A. Tarlecki, G. Tel, D. Uhlig, W. Unger, P. Urzyczyn, B. Vauquelin, L. Voelkel, F. J. Vries, I. Vrťo, Vyskoč, J. Warpechowska, G. Wechsung, J. Wiedermann, M. Wiegers, M. Will, T. Zeugmann, J. Winkowski, G. Winskel, S. Yoccoz, K. Zorychta.

MFCS '89 has been organized by the Institute of Informatics, University of Warsaw, in co-operation with the Association of Information Processing and with the financial support of the Polish Ministry of National Education (Grant RPI 09).

The Organizing Committee of MFCS '89 consists of K. Diks, M. Grabowski, A. Kreczmar, G. Mirkowska, A. Szałas.

We thank all the authors of the submitted papers for their valuable contributions and Springer-Verlag for their excellent co-operation in the publication of this volume.

Warsaw, May 1989

A. Kreczmar

G. Mirkowska

TABLE OF CONTENTS

FROM SPECIFICATION LANGUAGES TO SPECIFICATION KNOWLEDGE BASES: THE PTO APPROACH

Valery N. Agafonov
Tsentrprogrammsystem (Centre of Program Systems)
170650 Kalinin USSR

Abstract: We discuss the conceptual aspect of the situation in which program specifications are developed and used. Then we show limitations of specification languages as a way to organize conceptual means used in developing specifications and using them. Finally, we describe an approach to overcome these limitations by means of the PTO knowledge based system which can also serve for providing specifiers and users of specifications with the necessary elements of the mathematical culture.

1. The conceptual aspect of the specification situation

I should like to show limitations of specification languages as an instrument for working with mathematical notions in the course of developing and using program specifications and then to point such a direction that following it we can overcome these limitations to a considerable extent. With that end in view, we have to begin from the very beginning and to take up the question: what are the essence and the purpose of specifications?

First of all it is important to realize the delusiveness of simple and short answers to this question which are widespread in the literature. A typical specialist in the field of mathematical foundations of computer science will say most likely that a specification of a program is just a mathematically precise formulation of the problem which must be solved by the program. Since the main property of any program is its correctness and the correctness makes no sense without a specification then the essence and the purpose of specifications for such a person are in making it possible to construct correct programs or to verify programs. It is the truth, but only a part of the truth and not the whole of it.

It is often said that a specification has to describe what the program does, but not how it does this. Here is also a portion of the truth, but a portion of the falsehood as well, for there exist situations when the essence of a problem is just in how something is done.

In order to approximate to a more truthful answer to the above question, one has to distinguish and to estimate essential aspects of the specification situation, that is, the situation in which program specifications are developed and used. At least the following four aspects are worthy of a careful analysis:

1) the described problem: its nature and the natural range of notions in which the problem arises;

2) the people who take part in developing and using specifications - and especially their conceptual worlds or the stocks of notions which they have at their disposal;

3) the spectrum of specifications which are possible for a given problem, and the criteria for the choice of appropriate specifications from the spectrum;

4) the conceptual means (notions, mathematical structures, ways of description, etc.) which could come useful for description of the given problem and which exist in the literature, in specification languages or in the heads of the people involved or not involved in a given problem.

In itself the notion of program specification must be considered as relative - with respect to the specification situation including the mentioned aspects. The ideal specification in a given concrete situation would be such a description which will be accepted by given people with the available to them conceptual basis as the most straightforward, simple, natural, and clear formulation of the given problem. Therefore, we can say that the essence and the purpose of specification are in achieving understanding and explanation of a considered problem by such means which are convenient or at least acceptable for the given persons in the given situation. It can turn out that the same description in different situations does not meet the ideals of these situations. In order to bring together a specification and an ideal, one has to change the specification or the situation, or both. This mutual adjustment of specifications and situations is an important feature of my approach.

In section 1 I discuss the aspect of the specification situation which concerns the conceptual means. In section 2 specification languages as a way of the organization of conceptual means are considered, and in section 3 I present my approach to the organization of conceptual means by a specification knowledge base called PTO.

Conceptual means really used in program specifications or potentially useful are extremely varied. I systematized them in survey [1] and book [2] having distinguished and described several families of classes of conceptual means which are one of ways of organizing conceptual means in the PTO system. The first family of classes (in PTO its name is IHC - "Internally Homogeneous Classes") consists of classes of internally homogeneous means that have a definite mathematical unity and group around some central idea. It includes such classes as:

a) table means (notions grouping around the idea of a table);

b) equalities and rules of substitution (notions of an equality or an equation with various kinds of semantics (including semantics in terms of rules of substitution) and related notions of rewriting systems, production systems, their variants and generalizations),

c) logical means (means of the first-order logic and other logics),

d) graph means (graphs, trees, networks, diagrams with different semantical superstructures),

e) operations and expressions (operations on objects of the most various nature and ways of combining them by expressions),

f) procedural means (actions changing states and means of ordering and combining actions),

g) means of modularization, typing and structuring (notions of a type, a schema, a module, a frame, a mathematical structure),

h) means of naming (ways of naming, various variants of the notion of a variable, scopes of names, etc.),

i) axiomatic means (notions related to axiomatic methods of description).

For each of the above listed classes the principle according to which means are included in the class is fairly clear from the mentioned name of the class and several examples of means included in it, though this principle can not be formulated as a mathematical definition and there is no practical necessity in such a definition. Classes may intersect, but basically they are different. Each class is the embodiment of some essential layer of human thinking, a special way or a style of expressing human thought.

Another family (in PTO called GMN) - "General Mathematical Notions") includes three classes of means used for describing three fundamental kinds of mathematical objects: a) functions, b) sets, c) relations. It is done to explicitly show particular significance of functions, sets, and relations in specifications and also demonstrate the variety of specific forms and ways of their description.

Similarly, the GCSN family ("General Computer Science Notions") of classes of notions playing a fundamental role in computer science is formed. It includes, for example, the MOC class ("Models of Computations") in which each notion is a model of computations (finite automaton, pushdown automaton, model of computations in terms of re-write rules, etc.). This class demonstrates the variety of models of computations and ways of their description. The other two classes of this family (DESM and OPSM) consist of notions related to denotational semantics and operational semantics, respectively.

In a real situation the stock of notions available to given people is limited by a part of the above listed classes. Therefore, explicitly distinguishing them helps widen the horizon and activate search and use of adequate means of description.

An entirely different principle of constructing a family of classes is to form each class from the notions useful for describing problems of some application domain. In the family called LAP ("Large Areas of Programming") classes correspond to large areas established in programming and information processing, such as a) languages and language processors (compilers, etc.), b) data bases and knowledge bases, c) data processing oriented to the structure of the data processed, d) process control and action control (real time systems, etc.). We mean the areas which are fairly general and universal. Distinguishing classes of notions corresponding to such areas considerably orders the variety of conceptual means.

The above mentioned classifications embrace the notions from which specifications are constructed and composed of. Besides these notions, I distinguish and systematize the conceptual means (ideas, considerations, observations, recommendations) which are not included in specifications, but concern ways of constructing specifications and the situations in which specifications are developed and used (i.e. specifications are constructed not of them, but by them). The conceptual means of such a kind we call pragmatic and in PTO they are grouped into a specific family of articles (PRAGM) which includes the following classes of articles:

a) logical and mathematical discipline requirements which should be met in the definitions construction,

b) recommendations on how to write mathematics intelligibly and develop the description of a problem and things related to it,

c) systematized results of psychological experiments and empirical observations of experienced, knowledgeable persons which shed light on how people understand definitions constructed by these or those means, what difficulties they encounter, and what mistakes they make and how often.

Thus, our systematization of conceptual means embraces both the internal aspect of specifications (notions within specifications) and the external aspect of specifications (ideas and considerations outside specifications). Both these aspects are important and should be considered when specifications are constructed and used. I deliberately support them by the PTO system.

2. Specification languages as a way of organization of conceptual means

To organize something means, in general, to introduce some order, system or discipline useful for achieving some purpose. Here we consider the organization of conceptual means for achieving the purpose of specification discussed in section 1. A widely used and already traditional way of the organization is a specification language.

I distinguish two aspects in a specification language - the formalization aspect and the aspect of combining particular conceptual means in the language. The essence of the former is to fix the syntax and semantics of the language by a mathematically precise way. The essence of the latter is to select conceptual means included in the language and to bring them into a system in accordance with the principles on which the language is based.

The formalization aspect includes a degree of formalization. The minimal degree of formalization is to fix only syntax, while the description of semantics is not strictly regulated and remains on a more or less informal and intuitive level. The maximal degree of formalization is completely formalized syntax and semantics.

In principle, the latter provides preciseness of specifications, but can have a negative impact on their understandability. The problem is to formalize the description to such an extent that provides preciseness with the least detriment to understandability. Formality is not identical to preciseness. A formal language is an extremely refined form of the mathematical language, which is such a specific part of the natural language, which is based on mathematical objects and ways of manipulating them.

The mathematical language is distinguished from a formal language by flexibility, variability, absence of rigid fixation of syntax and semantics. An incompletely formalized specification language can appear to be closer to the mathematical language by its "explanatory power", by the provided capability to write understandable and sufficiently precise specifications. However, if we want to use specifications as program prototypes executable on a computer, then those aspects of a problem, which should be represented by a prototype, should be completely formalized in the language.

Thus, it is desirable to control the degree of formalization. The fixation of a language limits or excludes the capability of such a control.

The choice of details of a formalization of syntax and semantics is arbitrary to a great extent. Some details should be fixed only because in a completely formalized language everything should be fixed, though it is not caused by the needs of the problems description. There are different styles or kinds of the syntax and semantics description. Fixing a language, the description style is usually fixed, thus limiting capabilities to understand language constructions. The fixation of these or those features of a language almost always means the choice of one alternative from several possible and acceptable alternatives (points of view) and the rejection of the remaining which, thus, are hushed up by the langugge. Therefore, the fixation, as it is, required for providing unambiguous understanding of language constructions produces some undesirable "side effect", which has a negative impact on understandability and communicativeness of specifications.

More essential is the aspect of combining particular conceptual means. Studying various specification languages, I distinguished three general principles according to which a set of means included in the language is formed. The first principle I call "<u>specialization by domain</u>". According to this principle, the means included in a language are oriented to the description of a particular problem or subject domain, i.e. they are the means adequate, if possible, to the problems of this domain. When it is said about a specialized programming or specification language, usually it means just such a specialization.

An entirely different kind of specialization is "<u>specialization by means</u>". Here the principle of the language construction is to take as a basis of a language one kind (a specific class) of conceptual means. Such a language is oriented to means, but not to a domain.

The third principle of the language construction is to develop a <u>universal</u> or <u>general-purpose</u> <u>language</u>. This language is not specialized neither by means, nor by domain, it includes essentially heterogeneous means and does not limit its ápplication area in advance. To such languages also belong the "wide spectrum" languages including means of different levels - from purely declarative to machine-oriented.

To emphasize the difference between the language, in which heterogeneous means of several classes are used, and the language based on the means of one class, I call the former a polylanguage and the latter a monolanguage. Usually the languages specialized by domain and always the languages pretending to be universal are polylanguages. For each class of means of the above mentioned IHC family (besides the class of means of naming, which, in some way or other, are used in all languages) there are monolanguages specialized by means of this class - for example, table languages, equational languages, network languages, logical languages, etc. Many polylanguages can be characterized by pointing to, combinations of means of what classes are used in them. Thus, the correlation of languages with the above classification of conceptual means clarifies their position in the world of conceptual means of specification, and this classification serves as a coordinate system of this world.

The correlation of languages with the classification of means clearly shows the essential limitations of the set of means used in each particular language, - even in a polylanguage pretending to be universal.

Expressiveness of a language with respect to a given problem or subject domain is determined by the availability of conceptual means adequate for this domain in the language. A language specialized by means provides expressiveness only for a relatively narrow range of problems and domains to which the means embedded in it are adequate. The shortage of expressiveness induces language designers to create polylanguages. In its turn, for each polylanguage there always are problems and ideas which induce the designers to enrich it by new means or to turn to a new language with a richer set of means. It should be noticed that these means must be provided by the language directly, rather than be, in principle, constructed from the means directly included in the language - as, for example, in extendible languages or languages with the capability to define abstract data types. The latter languages allow us, in principle, to describe the desired notions, but the notions themselves are outside of the languages, they should be either invented, or taken from or found somewhere.

Besides providing better expressiveness, a polylanguage is also an attempt to embrace, systematize, and precisely describe some variety of ideas. It also can be considered as a tool for communication between people and a common basis for mutual understanding. Thus, a polylanguage is intended to meet three fundamental needs in: expressiveness, systematization and a common basis for mutual understanding. The function of a language as a means of exchanging ideas and achieving mutual understanding between people dealing with program specifications is, surely, expressed stronger in polylanguages than in specialized languages, but not strongly enough. It is important that they do not contain, firstly, the means which allow people to show relationships between different notions, points of view, and representations, and, secondly, the means related to methodology and pragmatics of specification development (above they were distinguished into the PRAGM family).

In order to understand each other, people should be able to recognize or to show relationships between different points of view and different representations. Such a purpose is not stated for specification languages at all. However, it is very important, because the achievement of it strongly influences understanding and explanation of specifications.

3. The specification knowledge base PTO

PTO is the name of a knowledge based system for versatile support of the program specification (the first letters of words "Practical Theory of Definitions" in Russian). The system is intended for the following basic variants of use:

1) orientation in the field of program specification and tutoring in the specification as an activity,

2) receiving information about principles and methods of developing and using specifications,

3) receiving information about the notion the user interested in (defining it more precisely, examples of its use, its synonyms and homonyms, forms of its recording, its relationships to other notions, points of view on it, literature on it),

4) choice of notions and forms of their recording for describing a particular problem,

5) choice of notions and forms of their recording for developing a specification language or for describing a class of problems,

6) receiving information about existing specification languages suitable for user's problems,

7) search of a suitable system for supporting the program specification,

8) choice of suitable representations of the notions which the user is interested in, including representations used for prototyping,

9) search of bibliography on the aspect of the program specification, in which the user is interested,

10) search of contacts with knowledgeable people in the field of