COMPUTER-BASED MATHEMATICAL MODELING

An Essay for the Design of Computer-Based Modeling Tools

TONY HÜRLIMANN

Institute of Informatics

Site Regina Mundi

University of Fribourg

1997

Computer-Based Mathematical Modeling

An Essay for the Design of Computer-Based Modeling Tools

Von der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität Freiburg zu genehmigende Habilitationsschrift zur Erlangung der Venia Legendi für das Fach “Informatik”.

Vorgelegt von

Dr. rer. pol. Tony Hürlimann

aus Walchwil/ZG

Gutachter: Prof. Dr. Jacques Pasquier,

Institut für Informatik, Universität Freiburg, 1700 Freiburg, Schweiz

Zweiter Gutachter: Prof. Dr. Johannes J. Bisschop,

Faculteit der Toegepaste Wiskunde, Universiteit Twente, 7500 AE Enschede, The Nederlands

Habilitationsabgabe: 21. Juli 1997

Preface

Computer-based mathematical modeling – the technique of representing and managing models in machine-readable form – is still in its infancy despite the many powerful mathematical software packages already available which can solve astonishingly complex and large models. On the one hand, using mathematical and logical notation, we can formulate models which cannot be solved by any computer in reasonable time – or which cannot even be solved by *any* method. On the other hand, we can *solve* certain classes of much larger models than we can practically *handle* and manipulate without heavy programming. This is especially true in operations research where it is common to solve models with many thousands of variables. Even today, there are no general modeling tools that accompany the whole modeling process from start to finish, that is to say, from model creation to report writing.

This book proposes a framework for computer-based modeling. More precisely, it puts forward a modeling language as a kernel representation for mathematical models. It presents a general specification for modeling tools. The book does *not* expose any solution methods or algorithms which may be useful in solving models, neither is it a treatise on how to build them. No help is intended here for the modeler by giving practical modeling exercises, although several models will be presented in order to illustrate the framework. Nevertheless, a short introduction to the modeling process is given in order to expound the necessary background for the proposed modeling framework.

Therefore, this work is not primarily intended for the model user or the modeler – the mathematician, physicist, or operation research model builder who use software to solve a given problem. Ultimately, this research is done for *them*, of course, and I hope I have created and implemented a useful software tool (LPL) to this end – at least for educational purposes.

Nor is this book about implementation topics. I do not describe here how I have concretely implemented my own modeling tools. This would have considerably augmented the number of pages and it would have been of scarce interest, since there are many excellent books on compiler construction already available.

This work offers *concepts* and a *general framework for modeling* and is mainly intended to help other designers of modeling tools. It details my views on 'why and how' we should create such tools and which utilities and functionalities they should contain in order to be useful for "practitioners". In that sense, this work is more a research programme than the presentation of an already established domain. I think that we are only at the beginning of a new development in "modeling with the aid of computers" and that I am able to barely scratch the surface of this new and exciting research field. I am, however, firmly convinced that there is a bright future for this topic – just as there was for word processors and spreadsheets fifteen years ago; or just as there was forty years ago when the first high level programming language was developed.

To grasp the main ideas, the reader should have some background in applied mathematics, logic, operations research, and computer science, as I rarely explain basic notions from these disciplines (like, for example, the concepts of "computational complexity" from the theoretical computer science, "cutting planes" for the polyhedral theory, and others), although these notions are fundamental to this book. I assume that the reader is familiar with these concepts. However, further references are always given.

My own background is, in order of importance, computer science, operations research and economics. This ordering also corresponds to the weight the three fields take in this book. Developing *ideas* merely on how modeling tools should be implemented is either easy or too far away from what is really needed; at least it is only a small part of the task as a whole to elaborate on all kinds of neat concepts without actually going through the thorny work of implementation. Only the process of implementation teaches you what works and what does not work.

My motivation for this research came initially from practical modeling: a research group under the direction of Dr. Hättenschwiler and Prof. Kohlas at the Institute of Informatics of the University of Fribourg (IIUF) had to build and maintain – at that time – large LP models. Although I was never directly engaged in the management of these models, I followed closely what they did and what they needed, and I created a first version of LPL (Linear Programming Language) which enabled them to formulate their models better.

That was 1987 and LPL was in its infancy. Nevertheless I got a Ph.D. for it and it encouraged me to extend and generalize it in several directions. Since then I have worked more or less intensely on the language. This book is the result of this “project LPL”.

The book is made up of three parts. In Part I, the concept of *model* and related notions are defined, the modeling life cycle is outlined, and different model types and paradigms are presented. Part II gives an overview of what actually exists in the line of computer-based modeling tools, explains why we need more, and presents a general modeling framework. Part III describes my own concrete contribution to this field: the modeling language LPL. A more complete survey of this book is given at the end of the *Introduction*.

Acknowledgements

There are many people who have directly or indirectly participated in the successful completion of this book. I particularly wish to thank the individuals who were invaluable in the process of making this work a reality:

Prof. Dr. Jürg Kohlas was the initiator of LPL. He proposed to me to develop a declarative language which would allow us to formulate concisely large LP models. Figure 0-1 is a sketch he made during a conference in the Swiss Alps (3ème Cycle d'Informatique, 1985). He has also supported and still supports actively my research since then.

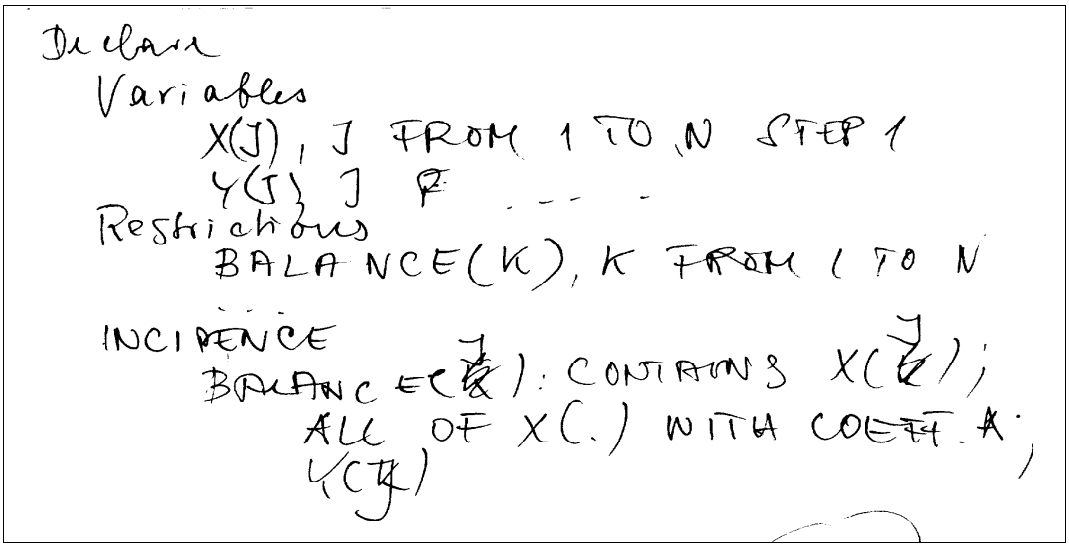

Figure 0-1: The Birth of LPL (Kohlas J.)

Prof. Dr. Ambros Lüthi was always around in the most critical moments of my life as a researcher. He gave me invaluable advice and supported my research project actively. Without him my career as a researcher would never have began and would have ended quite some time ago.

Prof. Dr. Jacques Pasquier is not only the supervisor of my habilitation, he was actively engaged in the modeling management from the hypertext and software engineering point of view. As a supervisor, he did a great job and read earlier drafts of this manuscript several times and gave me many invaluable tips. He also eagerly supported my project and has thus allowed me to pursuit my

research an additional two years.

Prof. Dr. Pius Hättenschwiler, the project manager of several large LPs, was and is still an intense interlocutor when it come to the practical use of LPL. Without him, LPL would be quite different or would probably even vegetate as a theoretical tool without much practical use. Marco Moresino, working under the direction of Pius, was and is probably the most intense and fiercest user of LPL. He found many subtle bugs.

Rare are the people in one's life who – when you meet them for the first time – seem to be like old friends who you have known years. Prof. Dr. Johannes Bisschop from the University Twente of Enschede is one of thee people for me. This is probably because we share a similar dream of a modeling system. Not only is he the second referee for my habilitation, but he also encouraged me to continue with my small (compared to the titans available on the markets) LPL implementation project when I was at the point of abandoning the whole thing (in Budapest 1994). A seemingly minor event might be of special interest in this context. When I first met Johannes, I asked him what he thought about the expressive power of modeling languages, whether they should include the whole power of a Turing Machine or not. His answer was a simple yes, and had an inestimable influence on what I now think should be a modeling language (the answer is in Chapter 7).

Prof. Dr. Arthur Geoffrion of UCLA allowed me to stay with him for one year. I owe him a great deal of credit when it comes to many of my ideas in modeling management systems.

Daniel Raemi, my old friend, has given extravagantly of his time to read earlier drafts of this book. His many corrections and suggestions have been very valuable. I also wish to thank Simon Lacey for his help in proof-reading the finished manuscript. Mitra Packham did a excellent job in reading the finished manuscript and greatly improved my English.

I also owe a depth of gratitude to many other colleagues and teachers from whom I have learned so much over the years. I thank them all.

None of the mentioned (or not mentioned) persons, however, is responsible for the defects in what follows. I alone am to blame for all and any errors which may remain or for the type setting of this document.

Finally, I wish to thank the following institutions for their financial support and the use of their infrastructure. The first is the Swiss National Science Foundation who financed my stay at UCLA in Los Angeles with Prof. Arthur Geoffrion for one year in 1989, and still supports my research (under Project No. 1217-45922.95). The second is the Institute of Informatics at the University of Fribourg in Switzerland (IIUF) where I developed and implemented most of my ideas in a favourable and agreeable atmosphere of collaboration. The Institute also let me use all the necessary equipment over many years to accomplish this work.

Tony Hürlimann
Freiburg, Switzerland
July 1997

Availability of the LPL System

Part III of this book presents software called **LPL.**[1] LPL is my own contribution to the field of computer-based mathematical modeling and is available free of charge from the Institute of Informatics at the University of Fribourg (IIUF), Switzerland. Currently, there are implementations on MS/DOS, Windows 95 and NT and Macintosh PowerPC. The version used in this book is 4.20.

[[Note at 2009: This old version 4.20 is no longer available on the net. The newest version always can be downloaded from.]]

www.virtual-optima.com

---

[1] Initially, **LPL** was an abreviation of ***L**inear **P**rogramming **L**anguage*, since it was designed exclusively for Linear Programs. During the intervening years, LPL's capability in logical modeling has become so important that one could also call it ***L**ogical **P**rogramming **L**anguage*. My intention, however, is to offer a tool for full model documentation too. Therefore, in the futur it may be called ***L**iterate **P**rogramming **L**anguage* (see [Knuth 1984]).

Table of Contents

1. INTRODUCTION

"Dass alle unsere Erkenntnis mit der Erfahrung anfange, daran ist gar kein Zweifel.
... Wenn aber gleich alle unsere Erkenntnis *mit* der Erfahrung anhebt, so entspringt sie darum doch nicht eben alle *aus* der Erfahrung."
— Kant I., Kritik der reinen Vernunft, 1877.

"Model-building is the essence of the operations research approach."
— Wagner H.M., Principles of Operations Research, 1975.

"Today mathematicians are generally part of a project team – as such they develop expertise about the process of system being analysed rather than act merely as solvers of mathematical equations."
— Cross/Moscardini, 1985.

Observation is the ultimate basis for our understanding of the world around us. But observation alone only gives information about particular events; it provides little help for dealing with new situations. Our ability and aptitude to recognize similarities in different events, to distil the important factors for a specific purpose, and to generalize our experience enables us to operate effectively in new environments. The result of this skill is *knowledge*, an essential resource for any intelligent agent.

Knowledge varies in sophistication from simple classification to understanding and comes in the form of principles and models. A *principle* is simply a general assertion and is expressed in a variety of ways ranging from saws to equations. , and are examples of principles. They can vary in their validity and their precision. A *model* is, roughly speaking, an analogy for a certain object, process, or phenomenon of interest. It is used to explain, to predict, or to control an event or a process. For example, a miniature replica of a car, placed in a wind tunnel, allows us to predict the air resistance or air eddy of a real car;

a globe of the world allows us to estimate distances between locations; a graph consisting of nodes (corresponding to locations) and edges (corresponding to streets between the locations) enables us to find the shortest path between any two locations without actually driving between them; and a differential equation system enables us to balance an inverted pendulum by calculating at short intervals the speed and direction of the car on which the pendulum is fixed.

A model is a powerful means of structuring knowledge, of presenting information in an easily assimilated and concise form, of providing a convenient method for performing certain computations, of investigating and predicting new events. The ultimate goal is to make decisions, to control our environment, to predict events, or just to explain a phenomenon.

1.1. Models and their Functions

Models can be classified in several ways. Their characteristics vary according to different dimensions: function, explicitness, relevance, formalization. They are used in scientific theories or in a more pragmatic context. Here are some examples classified by function, but also varying in other aspects. They illustrate the countless multitude of models and their importance in our life.

Models can *explain phenomena*. Einstein's special relativity explains the Michelson–Morley experiment of 1887 in a marvellously simple way and overruled the ether model in physics. Economists introduced the IS-LM or rational expectation models to describe a macroeconomical equilibrium. Biologists build mathematical growth models to explain and describe the development of populations. Modern cosmologists use the big-bang model to explain the origin of our world, etc.

There are also models *to control our environment*. A human operator, e.g., controls the heat process in a kiln by opening and closing several valves. He or she knows how to do this thanks to a learned pattern (model); this pattern could be formulated as a list of instructions as follows: “IF the flame is bluish at the entry port, THEN open valve 34 slightly”. The model is not normally explicitly

described, but it was learnt implicitly from another operator and maybe improved, through trial and error, by the operator herself. The resulting experience and know-how is sometimes difficult to put into words; it is a kind of *tacit* knowledge. Nevertheless one could say that the operator acts on the basis of a model she has in mind.

On the other hand, the procedure for aeroplane maintenance, according to a detailed checklist, is thoroughly *explicit*. The model is possibly a huge guide that instructs the maintenance staff on how to proceed in each and every situation.

Chemical processes can be controlled and explained using complex mathematical models. They often contain a set of differential equations which are difficult to solve (see [Rabinovich 1992]). These models are also explicit and written in a *formalized* language.

Other models are used to control a social environment and often contain *normative* components. Brokers often try, with more or less success, to use guidelines and principles such as the FED publicizes a high government deficit provision, the dollar will come under pressure, so sell immediately. Such guidelines often don't have their roots in a sophisticated economic theory; they just prove true because many follow them. Many models in social processes are of that type. We all follow certain principles, rules, standards, or maxims which control or influence our behaviour.

Still other models constitute the basis *for making decisions*. The famous waterfall model in the software development cycle says that the implementation of a new software has to proceed in stages: analysis, specification, implementation, installation and maintenance. It gives software developers a general idea of how to proceed when writing complex software and offers a rudimentary tool to help them decide in which order the tasks should be done. It does not say anything about how long the software team have to remain at any given stage, nor what they shoud do if earlier tasks have to be revised: It represents a *rule-of-thumb*.

An example of a more *formal* and complex decision-making-model would be a mathematical production model consisting typically of thousands of constraints and variables as used in the petroleum industry to decide how to transform crude oil into petrol and fuel. The constraints – written as mathematical

equations (or inequalities) – are the capacity limitations, the availability of raw materials etc. The variables are the unknown quantities of the various intermediate and end products to be produced. The goal is to assign numerical values to the variables so that the profit is maximized or some other goal is attained.

Both models are tools in the hand of an intelligent agent and guide her in her activities and support her in her decisions. The two models are very different in form and expression; the waterfall model contains only an informal list of actions to be taken, the production model, on the other hand, is a highly sophisticated mathematical model with thousands of variables which needs to be solved by a computer. But the degree of formality or complexity is not necessarily an indication of the "usefulness" of the model, although a more formal model is normally more precise, more concise, and more consistent. Verbal and pictorial models, on the other hand, give only a crude view of the real situation.

Models may or may not be *pertinent* for some aspects of reality; they may or may not *correspond* to reality, which means that models can be misleading. The medieval model of human reproduction suggesting that babies develop from homunculi – fully developed bodies within the woman's womb – leads to the absurd conclusion that the human race would become extinct after a finite number of generations (unless there is an infinit number of homunculi nested within each other). The model of a flat, disk-shaped earth prevented many navigators from exploring the oceans beyond the nearby coastal regions because they were afraid of "falling off" at the edge of the earth disk. The model of the falling profit rate in Marx's economic theory predicted the self-destruction of capitalism, since the progress of productivity is reflected in a decreasing number of labour hours relative to the capital. According to this theory, labour is the only factor that adds plus-value to the products. Schumpeter agreed on Marx's prediction, but based his theory on a very different model: Capitalism will produce less and less innovative entrepreneurs who create profits! The last two examples show that very different sophisticated models can sometimes lead to the same conclusions.

In neurology, artificial neural networks, consisting of a connection weight matrix, could be used as models for the functioning of the brain. Of course,

such a model abstracts from all aspects except the connectivity that takes place within the brain. However, some neurologists believe that only 5%(!) of information passes through the synapses. If this turned out to be true, artificial neural nets would indeed be inappropriate models for the functioning of the brain.

One can see from these examples that models are ubiquitous and omnipresent in our lives. “The whole history of man, even in his most non-scientific activities, shows that he is essentially a model-building animal” [Rivett 1980, p. 1]. We live with “good” and “bad”, with “correct” and “incorrect” models. They govern our behaviour, our beliefs, and our understanding of the world around us. Essentially, we see the world by means of the models we have in mind. The value of a model can be measured by the degree to which it enables us to answer questions, to solve problems, and to make correct predictions. Better models allow us to make better decisions, and better decisions lead us to better adaptation – the ultimate “goal” of every being.

1.2. The Advent of the Computer

This work is not about models in general, their variety of functions and characteristics. It is about a special class thereof: mathematical models. Mathematics has always played a fundamental role in representing and formulating our knowledge. As sciences advance they become increasingly mathematical. This tendency can be observed in all scientific areas irrespective of whether they are application- or theory-oriented. But it was not until this century that formal models were used in a systematic way to solve practical problems. Many problems were formulated mathematically long ago, of course. But often they failed to be solved because of the amount of calculation involved. The analysis of the problem – from a practical point of view at least – was usually limited to considering small and simple instances only.

The computer has radically changed this. Since a computer can calculate extremely rapidly, we are spurred on to cast problems in a form which they can manipulate and solve. This has led to a continuous and accelerated pressure to formalize our problems. The rapid and still ongoing development of computer technologies, the emergence of powerful user environment software for geometric modeling and other visualizations, and the development of

numerical and algebraic manipulation on computers are the main factors in making modeling – and especially mathematical modeling – an accessible tool not only for the sciences but for industry and commerce as well.

Of course, this does not mean that by using the computer we can solve every problem – the computer has only pushed the limit between practically solvable and practically unsolvable ones a little bit further. The bulk of practical problems which we still cannot, and never will be able, to solve efficiently, even by using the most powerful parallel machine, is overwhelming. Nevertheless, one can say that many of the models solved routinely today would not be thinkable without the computer. Almost all of the manipulation techniques in mathematics, currently taught in high-schools and universities, can now be executed both more quickly and more accurately on even cheap machines – this is true not only for arithmetic calculations, but also for algebraic manipulations, statistics and graphics. It is fairly clear that all of these manipulations will become standard tools on every desktop machine in the very near future. Twenty years ago, the hand calculator replaced the slide rule and the logarithm tables, now the computer replaces most of those mathematical manipulations which we learnt in high-school and even at university.

1.3. New Scientific Branches Emerge

Some research communities such as *operations research* (OR) owe their very existence to the development of the computer. Their history is intrinsically linked to the development of methods applicable on a computer. The driving force behind this development in the late 1940s was the Air Force and their Project SCOOP. Numerical methods for linear programming (LP) were stimulated by two problems they had to solve: one was a diet problem. Its objective is to find the minimum cost of providing the daily requirement of nine nutrients from a selection of seventy-seven different foods. Initially, the calculations were carried out by five computers[2] in 21 days using electromechanical desk calculators. The simplex method for linear

[2] Human calculators carrying out extended reckoning were called “computers” until the end of the 1940's. Many research laboratories utilized often poorly paid human calculators – most of them were women.

programming (LP), discovered and developed by Dantzig in 1947[3], was and still is one of the greatest successes in OR. Together with good presolve techniques we can now solve almost any LP problems up to 250,000 variables and 300,000 constraints (or alternatively 1,000,000 variables and 100,000 constraints) by a direct method. Problems with 800 constraints and 12 million variables have been solved using column generation methods. Problem belonging to the particular class of transportation problems and consisting of 10's of thousands of constraints and 20 million variables are routinely solved today. LPs containing special structures with 10^{50} variables are solved frequently.[4] Unfortunately, this success story does not prove true for most other interesting problems. On the contrary, it was discovered that almost all integer and combinatorial problems are algorithmically hard to solve. No efficient procedure is yet known despite intensive research over the last 40 years. There seems to be little hope of ever finding an efficient one. There are even problems that cannot be solved, neither efficiently nor inefficiently, but that is another story...

Artificial intelligence (AI) is also thoroughly dependent on the progress in computer science. Initially, many outstanding researchers in this domain believed that it would only be a question of decades before the intelligence of a human being could be matched by machines. Even Turing was confident that it would be possible for a machine to pass the Turing Test by the end of the century. Some scientists believe that we are not far from that point. Even though most problems in AI turned out to be algorithmically hard, since they are closely related to combinatorial problems. This led to an intensive research of heuristics and "soft" procedures – methods we humans use daily to solve complex problems. The combination of these methods and the computer's extraordinary speed in symbolic manipulation produces a powerful means to

---

3 See: Dantzig G.B. [1991], Linear Programming, in: History of Mathematical Programming, A Collection of Personal Reminiscences, edited by Lenstra J.K., Rinnooy Kann A.H.G., Schrijver A., CWI, North-Holland, Amsterdam, 1991.

4 See: Infanger G., [1992], Planning under Uncertainty: solving large-scale stochastic linear programs, Techn. Univ. Wien. See also: Hansen P., [1991], Column Generation Methods for Probabilistic Logic, in: ORSA Journal on Computing, Vol. 3, No. 2, pp. 135–148.

implement complex problems in AI.

Other scientific communities had already developed highly efficient procedures for solving sophisticated numerical problems before the first computer was built. This is especially true in physics and engineering. For example, Eugène Delaunay (1816–1872) made a heroic effort to calculate the moon's orbit. He dedicated 20 years to this pursuit, starting in 1847. During the first ten years he carried out the hand calculation by expressing the differential system as a lengthy algebraic expression in power series, then during the second ten years he checked the calculations. His work made it possible to predict the moon's position at any given time with greater precision than ever before, but it still failed to match the accuracy of observational data from ancient Greece. A hundred year later, in 1970, André Deprit, Jacques Henrard and Arnold Rom revised Delaunay's calculation using a computer-algebra system. It took 20 hours of computer time to duplicate Delaunay's effort. Surprisingly, they found only 3 minor errors in his entire work. Calculations in celestial mechanics was the most challenging task in the 19th century and no engineer's education was complete without a course in it.[5]

Many numerical algorithms, such as the Runge-Kutta algorithm and the Fast Fourier Transform (FFT), were already known – the later even by Gauss[6] – before the invention of the computer and they were much used by human “computers”. But for many problems these efforts were hopeless, for the simple reason that the human computer was too slow to execute the simple but lengthy arithmetics.

An interesting illustration of this point is the origin of numerical meteorology.[7] Prior to World War II, weather forecasting was more of an art, depending on

---

5 See: Peterson I, [1993], Newton's Clock, Chaos in the Solar System, W.H. Freeman, New York, pp. 215, see also: Pavelle R., Rothstein M., Fitch J., [1981], Computer Algebra, in: Scientific American, Dec. 1981, p. 151.

6 See: Cooley J.W., [1990], How the FFT Gained Acceptance, in: Nash S.G. (ed.), A History of Scientific Computing, ACM Press, New York, pp. 133–140. See also [Heideman al. 1985].

7 An excellent account of its origins is given in Chapter 6 of: Aspray W., [1990], John von Neumann and the Origins of Modern Computing, The MIT Press, Cambridge.

subjective judgement, than a science. Although, in 1904, Vilhelm Bjerknes had already elaborated a system of 6 nonlinear partial differential equations, based on hydro- and thermodynamic laws, to describe the behaviour of the atmosphere, he recognized that it would take at least three months to calculate three hours of weather. He hoped that methods would be found to speed up this calculation. The state of the art did not fundamentally change until 1949 when a team of meteorologists – encouraged by John von Neumann, who regarded their work as a crucial test of the usefulness of computers – fed the ENIAC[8] with a model and got a 24-hour "forecast", after 36 hours of calculations, which turned out to be surprisingly good. Four years later, the Joint Numerical Weather Prediction Unit (JNWPU) was officially established; they bought the most powerful computer available at that time, the IBM 701, to calculate their weather predictions. Since then, many more complex models have been introduced. The computer has transformed meteorology into a mathematical science.[9]

In still other scientific communities, until very recently, mathematical modeling was not even a topic or was used in a purely academic manner. Economics is a good example of this. Economists produced many nice models without practical implications. Sometimes, such models have even been developed just to give more credence to policy proposals. But mathematical models are not more credible simply because they are expressed in a mathematical way. On the other hand, important theoretical frameworks – such as game theory going

---

8 ENIAC (Electronic Numerical Integrator and Computer) was one of the first electronic general-purpose computers built at the Moore School of Engineering, University of Pennsylvania, in 1946.

9 Meterology is only one representative of a large problem class that can be formulated as differential systems, fluid dynamics being another. These applications are still regarded as "grand challenge" problems in comtemporary supercomputing. It is estimated that a direct simulation of air flow past a complete aircraft will require at least an exaflop (10^{18}) computer. The supercomputer Cray YMP is running at 200 megaflops (10^6). See: COVENEY P., HIGHFIELD R., [1995], Frontiers of Complexity, Fawcett Columbine, New York, p. 68-69. However, Intel has announced (December 1996) the first teraflop (10^{12}) computer consisting of more than 7000 Pentium Pro processors.

back to the late twenties when John von Neumann published his first article on this topic – have been developed. Realistic *n*-person games of this theory cannot be solved analytically. They need to be simulated on computers. So the attitude towards these formal methods is also gradually changing in these other sciences as well. Today, no portfolio manager works without optimizing software. Branches such as evolutionary biology which have been more philosophical, are also increasingly penetrated by mathematical models and methods.

1.4. Mathematical Modeling – the Consequences

We should not underestimate the significance of the development of computers for mathematical modeling. Their capacity to solve mathematical problems has already changed the way in which we deal with and teach applied mathematics. The relative importance of skills for arithmetic and symbolic manipulation will further decrease. This does not mean that we will no longer need applied mathematicians. On the contrary, we will need even more people qualified to translate real-life problems into formal language, and what activity is more rewarding and intellectually more challenging in applied mathematics than – *modeling*? While relatively fewer mathematicians are needed to solve a system of differential equations or to manipulate certain mathematical objects, more and more are needed who are skilled and expert in modeling.

This development is by no means confined to science. Since World War II, a growing interest has been shown in formulating mathematical models representing physical processes. These kinds of models are also beginning to pervade our industrial and economical processes. In several key industries, such as chip production or flight traffic, optimizing software is an integral part of their daily activities. In many other industrial sectors, companies are beginning to formalize their operations. *GeoRoute*, for instance, a transportation company in the United States who delivers more than 10 millions items a year all over the USA using 500 trucks, is literally built on a computer program, called NETOPT, which optimizes the deliveries in real-time. Mathematical models are beginning to be an essential part of our highly developed society.

But the use of models in an industrial context still demands a highly specialized team of experts; experts in operations research, in computer science as well as

in management. These teams are costly. Is it imaginable that for many modeling tasks these teams could one day be replaced by a small group or even a single person who uses a highly sophisticated computer-based modeling tool? The question seems to be a little bit naive! But one has to look to the problem in another way: In practice, many operations are *not* formalized, and highly developed optimizing software is *not* used, *because the set-up of the mathematical model is too costly*. So a large amount of the precious time of these highly qualified people is wasted to express the model in a way that the solver can use.

The dream of such a miraculous software has already got a name: “Expert Systems” or “Decision Support Systems (DSS)”. According to the philosophy of these new technologies, the idea that only experts can perform complex work is outdated. Certainly, experts are needed to assure progress in their special fields. But in order to solve more and more complex problems efficiently, experts can be replaced by generalists. The argument is: “… the real value of expert systems technology lies in its allowing relatively unskilled people to operate at nearly the level of highly trained experts. … Generalists supported by integrated systems can do the work of many specialists, and this fact has profound implications for the ways in which we can structure work.” [Hammer/Champy 1994, p. 93]. While the treatise of Hammer/Champy is not about DSS, the quotation expresses very well the underlying philosophy. There is a vast amount of literature about DSS. But, as is often the case, quantity is not necessarily a sign of quality. Too much expectation has been sown, and too few results have been harvested. The main limitation of most DSS – as built and used today – is that they are *not general* tools for modeling, but can only be applied in a restrictive way only and for narrowly specified problems. Nonetheless, the need for such tools is evident.

An important prerequisite for the widespread use of DSS tools is a change in the mathematical curriculum in school: A greater part of the manipulation of mathematical structures should be left to the machine, but more has to be learnt about how to recognize a mathematical structure when analyzing a particular problem. It should be an important goal in applied mathematics to foster creative attitudes towards solving problem and to encourage the students' acquisition and understanding of mathematical concepts rather than drumming purely mechanical calculation into their heads. Only in this way can the student

be prepared for practical applications and modeling.[10]

But how can modeling be learnt? Problems, in practice, do not come neatly packaged and expressed in mathematical notation; they turn up in messy, confused ways, often expressed, if at all, in somebody else's terminology. “Whereas problem solving can generally be approached by more or less well-defined techniques, there is seldom such order in the problem posing mode” [Thompton 1989, p. 2]. Therefore, a modeler needs to learn a number of skills. She must have a good grasp of the system or the situation which she is trying to model; she has to choose the appropriate mathematical tools to represent the problem formally; she must use software tools to solve the models; and finally, she should be able to communicate the solutions and the results to an audience, who is not necessarily skilled in mathematics.

It is often said that modeling skills can only be acquired in a process of learning-by-doing; like learning to ride a bike can only be achieved by getting on the saddle. It is true that the case study approach is most helpful, and many university courses in (mathematical) modeling use this approach [Clements 1989], [Ersoy 1994]. But it is also true – once some basic skills have been acquired – that theoretical knowledge about the mechanics of bicycles can deepen our understanding and enlarge our faculty to ride it. This is even more important in modeling industrial processes. It is not enough to exercise these skills, one should also acquire methodologies and the theoretical background to modeling. In applied mathematics, more time than is currently spent, should be given to the study of discovery, expression and formulation of the problem, initially in non-mathematical terms.

So, the novice needs first to be an observer and then, very quickly, a do-er. Modeling is not learnt only by watching others build models, but also by being actively and personally involved in the modeling process (case study approach).

---

[10] A desperate teacher of applied mathematics wrote: “If one fights for applied mathematics teaching, one can actually promise only ’blood, sweat and tears’ and, moreover, must already fear one more wave (the computer wave) which will be supported as widely and extensively as that first” [Schupp H., in: Blum al., 1989, p. 36].

As a consequence of these needs, which became evident in the seventies, a "movement of model-building" in applied mathematics education has emerged, marked by an increasing number of publications, see [Clements 1989, p. 10]. The journal *Teaching Mathematics and Its Applications* was founded in 1982 (as a successor of the *UMAP Journal*), and regular conferences have been organized in this field since 1983.

The use of computer modeling tools to simulate, visualize, and analyze mathematical structures has spread steadily. In the eighties, the "micro-computer" – a word which disappeared as quickly as it emerged – was the archetype of a whole generation of self-made quick and dirty models. Everyone produced their own simulation tool, mathematical toolbox, etc. This phenomena has now almost vanished. We have powerful packages such as Mathematica [Wolfram 1991], Maple [Char et al., 1991], MatLab [MatLab 1993], Axiom [Jenks, 1992], the NAG-library [NAG], to mention but a few, for solving complex problems. But the modeling process still needs more than easy-to-use solving tools: It also needs tools to *integrate different tasks*: such as data manipulation tools, representation tools for the model structure, viewing tools to represent the data and structure in different ways, reporting tools, etc. Some tasks can be done by different software packages: data is best manipulated in databases and spreadsheets, and is best viewed by different graphic tools.

1.5. Computer-Based Modeling Management

This brings us to the heart of this work: Modeling is still an art with tools and methods built in an ad hoc way for a specific context and for a particular type of model. Whilst I believe that modeling will always be an art in the sense of a creative process, I also firmly believe that we can build *general* computer-based modeling tools that can be used not only to help find the solution to a mathematical model, but also *to support the modeling process itself*. The extraordinary advances in science, on one hand, and computing performance, on the other hand, leave a cruel gap that can be bridged by general and efficient *modeling management systems*. Only when we reach this point, will the new technologies be fully exploited.

There are three main reasons, in my opinion, why DSS have not fulfilled their

promise.

- The first reason has to do with the modeling process itself. Every problem needs its own model specification, and making good models *is* difficult. No modeling management system however sophisticated can do the job on its own. A major difficulty in developing such a system is that the modeling of knowledge representation involves a wide range of activities. Modeling is done by people with very different backgrounds and in various contexts, and it is difficult to develop modeling tools which can be used by almost everyone.

  Therefore, most modelers still develop and use their own ad hoc tools to manage their models. There are important disadvantages in doing so. Models are difficult to maintain with a changing crew. Model transparency may suffer, and portability to different environments is limited. Often a model, or parts of it, could also be used in another context, but reusability is almost impossible. Operations research and artificial intelligence journals are full of articles describing a special implementation of a model and its environment. The cost of developing such ad hoc tools should not be underestimated. This is one of the main reasons why decision makers still tend to use rules-of-thumb rather than the troublesome path of model building.

- The second reason has to do with the solution complexity of many problems. While the specification of the problem is sometimes straightforward, the solution is not. This is especially true for discrete problems. In this case, we need tools which assist the modeler in reformulating and translating the model in such a way that different methods or heuristics can be used to solve it. The model structure must be easy to manipulate. We need flexibility in manipulating model structures in the same way as we are able to manipulate data – using databases for instance. None of the tools available today are suited for these tasks.

- The third reason concerns coping with *uncertainty* and *inconsistency*. In real problems, uncertainty is ubiquitous, and eliminating inconsistencies from a model is one of the modeler's main task. Both aspects have not received the attention they deserve, at least in the field of mathematical modeling. Chapter 4 explains, to some extent, why I think that these aspects are important.

It would be of great use, if decision makers owned some *universally usable* modeling tools and methods to do their job. Not only are we sorely in need of them, but – I am convinced – they are also feasible. Actually, there *are* important research activities going on in the realm of modeling management systems.

1.6. About this Book

This book does *not* propose (computer-based) solution methods for mathematical models, *nor* does it give suggestions on how to build mathematical models. It proposes a framework for representing and specifying mathematical models on computers in machine-readable form. It suggests a *modeling language* in contrast to a *programming language*.

The “ultimate” modeling language is in many senses a superset of a programming language: it specifies *what* the model is in a declarative way *and* it encodes *how* the model has to be solved in a procedural (algorithmic) way. A programming language only represents the second, algorithmic part. I shall concentrate on the first, declarative part of how mathematical models can be specified.

Part I explores fundamental features of models. It consists of three chapters. Chapter 2 defines the notion of “mathematical model” and other related concepts. It begins with general considerations about *models* and ends with a few thoughts about declarative and procedural knowledge. Chapter 3 gives a comprehensive overview of the modeling life cycle, wherein the different steps from building to solving models are traced. Chapter 4 summarizes different paradigms of models, different model types and their purposes. It also suggests a short introduction to modeling uncertainty – an important and all too often neglected topic in real life modeling.

Part II presents more thorough justifications for the need for modeling management systems, and especially for a modeling language. It also contains three chapters. Chapter 5 presents the current situation in model management systems as well as the problems linked to it. Different approaches and systems

in computer-based modeling are exposed in Chapter 6. The five most frequently used methodologies are reviewed. Finally, Chapter 7 proposes my own view of a general framework in computer-based modeling. This Chapter is the core of this book.

Part III displays my own concrete contribution to this field of research: the modeling language LPL. Chapter 8 shows how the general framework exposed in the previous chapter is implemented. An exact syntax notation of the language is given in extended Backus-Naur form. Chapter 9 exposes in which modeling environment the language has been embedded. Model browser and different graphical tools are presented. Finally, Chapter 10 gives different applications of the framework. Several examples used in previous chapters are restated in LPL, others explain various aspects such as units, modeling of logical constraints, and others.

The glossary summarizes the specific vocabulary used in this book.

Many examples are used in this text to illustrate different aspects of modeling. They are all clearly numbered and listed after the table of contents. The symbol character ¤ is used to indicate in the text where the example ends.

[[….You would like to read the whole book? Please contact the author at tony.huerlimann@unifr.ch.....]]

11. CONCLUSION

> “The best is yet to come. We've only scratched the surface...”
> — Shannon C.E.

Computer-based modeling is not an “ad hoc science” where we could apply the slogan “everything goes”; on the contrary, it is, like software engineering, a discipline which must follow certain criteria of quality. These criteria are *reliability* to ensure correctness and robustness of models, and *transparency* to enhance model extendibility, reusability and compatibility. Reliability can be achieved by a unique notation, in which models are coded, by a formal specification of its semantics, and by introducing types, units and other checking mechanisms, in order to enforce domain consistencies. Transparency can be obtained by flexible decomposition techniques which fractionates the whole model into easily maintainable software modules and components.

Models are different from programs in the sense that models often contain declarative *and* procedural elements of knowledge, while programs only describe algorithms. The declarative part in a model represents the state space – the “*what-is*”, while the procedural part covers *how* the model is to be transformed and, ultimately, solved. This combination implies that the notation, i.e. the computer-readable language, in which the model is expressed, must embrace both paradigms, the declarative and the procedural one.

One of the main goal of this book was to propose a modeling language, which allows the representation of both of these characteristics of models, and to

emphasize that modeling language design is submitted to the same criteria as programming language design in general.

With respect to this goal, several new concepts have been introduced:

- the *model specification as an encapsulated software module* containing a well defined interface (exports and imports) in order to communicate with other modules,
- the *coexistence (not the mixing) of declarative and procedural style* within the same language to allow the concise expression of a much broader number of models, than would be possible otherwise,
- *hierarchical index-sets* to allow the presentation and manipulation of hierarchical multi-dimensional data tables and sets in general.
- *graphical and textual views* which allow the modeler and user to switch in a bi-directional manner between different visual representations of the model, in order to browse, to edit, to modify, and to maintain it.

Specifying models as encapsulated modules or decomposing them into well defined, encapsulated software components have the advantage to reuse and to maintain the parts independently from the whole. Modifying a module is confined to its boundary, since the communication to other modules is explicit and transparent, and its inner working is hidden from the outside. Complexity can only be controlled by breaking the whole into autonomous, self-contained pieces. With this respect, no difference exists between models and programs, between “model engineering” and “software engineering”; all techniques and principles used in software engineering (information hiding, structured programming, object orientation, modular design, etc.) are equally applicable for model building.

Combining the declarative and procedural style, however, are necessary in coding models. A striking example, of why declarative and procedural knowledge must coexist for building models, was given in the cutting stock problem (Example 7-2). This model consists of two purely declarative models (an LP and a knapsack model) and a simple executable part, in which both (sub)models are solved several times. Many other models have a similar “solution structure”. The only two alternatives to approach such problems are

(1) to code them as very large (declarative) models or (2) to program the column generation method – or whatever method is used – explicitly in a procedural way. The first approach yields elegant and compact models which are unsolvable because of their size. The second obscures the model structure as well as the solution process. Combining the two paradigms not only makes the code much more readable and transparent but also a lot shorter.

An important constituent of every modeling language is its indexing mechanism in order to formulate large models. Like the mathematical notation itself, the modeling language should use indexed notation which allows the formulation of sparse tables and many indexed operations in a very convenient way. The concept of hierarchical index-set, as proposed in this book, is a generalization of this indexing mechanism and unifies different well-known notions already used in existing modeling languages (such as “compound index-sets”, “indexed index-sets”, “set of index-sets” and others). This fundamental part of the language also allows the manipulation of hierarchically organized data tables and other entities.

Of course, the modeling language is not the whole. It must be embedded in a modeling framework with model browsers and editors. Viewing models from different angles and through particular filters is an important ingredient to every modeling management system. However, this presupposes a unique “kernel form” in which a model is coded, otherwise it would not be possible to switch between the views in a bi-directional way. Several views – graphical and textual – have been presented and exemplified.

This book has taken a somewhat biased approach of *declarative* modeling, since it favoured that part of model specification, and most model examples are purely declarative. This bias is only natural in view of the overwhelming literature about procedural and algorithmic languages. The declarative style has received surprisingly little attention. I think, this is due to the prevailing opinion that “real” modeling cannot be done (that is, solved) using a declarative style. Only toy examples are possible. At best, a declarative representation can be used for model documentation, so goes the argument. I recall here that *solving a model* is *only one* aspect, documenting being another, and a model has many other functions. Having a language to automatically pretty-print a model specification and to compute the documentation *is* useful;

in fact that's all what TEX [Knuth 1986] is about for type-setting.

Furthermore, declarative modeling is powerful and generates compact models which are easy to maintain. Therefore, the modeler should favour this paradigm. On the other hand, many problems can easily be expressed as a mathematical constraint which is unsolvable in any reasonable time – or worse, of which we do not know *when* the solution is returned. If the model is to be solved, it is important to find an efficient formulation. This is, in general, easier said than done! Finding efficient formulation is a creative work. In any case, *the modeling language does not help in finding efficient formulations, it only provides syntactical elements in order to code them*. We do not expect from a programming language to assist us in finding an efficient sorting algorithm, we are already happy if we can code it without too much compromises to the language syntax! Certainly, it is so simple to formulate a model declaratively and then to blame the solver of not being able to solve it. In a world, where most “real” problems are (at least) NP-complete, there is little hope for having efficiency and beauty at the same time. Nevertheless, in declarative modeling – sometimes– we get a maximum of both.

The opinion may be expressed that I have proposed but a different programming language and that we do not need the term “modeling language” in any way. I would not mind to renounce this term, but would then insist that there are two particular elements in this new “programming language”, namely *mathematical variables and constraints*, not present in any other language (with exception of the logical programming languages). So, then my proposition is simply another logical programming language? Not quite! The difference to logical programming languages is that the proposal here clearly separates declarative and procedural knowledge and that the search procedures are made explicit which are coded in a procedural style or are encapsulated in an external solver. In view of the many very different methods that we need for solving a wide range of models, it is my impression that such an approach is more powerful. The modeler has the most flexible way in formulating the model: mathematically well-understood components can be done declaratively in a straightforward way, while the other parts are coded in a procedural way. In the logical programming paradigm the modeler is confined to the search and solution procedures which are hard-coded in the engine machine itself.

Another powerful concept is integrated into the modeling language which distinguish it from most programming languages: the indexing mechanism. This introduces *set manipulation* which is often hard to code in other languages, because they appear in so many forms, and dealing with them in an efficient way is not straightforward. The indexing mechanism also predisposes the modeling language to tackle with a mass of data and to communicate with databases in a natural way.

The propositions suggested in this book are far from being a well-settled theory about computer-based modeling. The research topic is simply too new. Many modeling languages exist in a pre-prototypical stage, but only few are really useful modeling languages in a general sense. Many syntax-concepts in existing languages are ad hoc constructs and should be embedded in a more consistent framework of language design. This does not mean, that I want to blame existing languages – modeling is hard and modeling language design is even harder – it only means that we should begin to look at modeling languages as fully-fletched programming languages in which we code the models, in the same way as we code algorithms as programs.

Using modeling languages is a new paradigm of mathematical modeling; and I am convinced that they will have a bright future if they follow certain criteria in language design coupled with simplicity and elegance.

> “Language – notation – shapes thought and mind. Our languages, and how they are defined, have a tremendous influence on what we can express and how simple. Often, a concept becomes clear only when a suitable notation for it has been defined and when rules for manipulating the notation have been developed: a concept and its expression are inextricably intertwined.”
>
> — Gries David in: [Hoare 1989, Preface].

In this sense, we would prefer to live with unsolved problems than accept a model and its solution we cannot understand.

APPENDICES

REFERENCES

ADAMO J.M., [1980], L.P.L. – A Fuzzy Programming Language, 1. Sytactic Aspects, 2. Semantic Aspects, in: Fuzzy Sets and Systems Vol. 3, p. 151–179 and pp 261–289.

AGGOUN A. & BELDICEANU N., [1993], Extending CHIP in Order to Solve Complex Scheduling and Placement Problems, in: Math. Comput. Modelling, Vol. 17, No. 7, p. 57–73.

AHO A.V. & SETHI R. & ULLMAN J.D., [1986], Compilers, Principles, Techniques, and Tools, Addison-Wesley Publ., Reading, Massachusetts.

AIBA A. & SAKAI K. & SATO Y. & HAWLEY D.J. & HASEGAWA R., [1988], Constraint Logic Programming Language CAL, in: FGCS'88, p. 263–276.

AIGNER M., [1988], Combinatorial Search, Wiley-Teubner Series in Computer Science, Chichester/Stuttgart.

ANDERSEN K.A. & HOOKER J.N., [1994], Bayesian Logic, in: Decision Support Systems 11, p 191–210.

ANGEHRN A.A. & LÜTHI H-J., [1990], Intelligent Decision Support Systems: A Visual Interactive Approach, in. Interfaces, Vol. 20:6, pp. 17–28.

BACKUS J., [1978], Can Programming be liberated from the von Neumann style? in: Comm. of the ACM 21 (1978), pp. 613–641.

BAECKER R & DIGIANO C. & MARCUS A., [1997], Software Visualization for Debugging, in: Comm of the ACM, Vol. 40, 4 (April), pp. 44–54.

BALCI O. & SHARDA R. & ZENIOS S.A. (eds), [1992], Computer Science and Operations Research, New Developments in their Interfaces, Pergamon Press, Oxford.

BALDWIN G., [1987], Implementation of Physical Units, Sigplan Notices, Vol.22, No.8, August 87, pp. 45–50

BELDICEANU N. & CONTEJEAN E., [1994], Introducing Global Constraints in CHIP, in: Math. Comput. Modelling, Vol. 20, No. 12, p. 97–123.

Bell Northern Research, [1988], BNR-Prolog, Reference Manual, Ottawa, Ontario, Canada.

BELLOMO N. & PREZIOSI L., [1995], Modelling Mathematical Methods and Scientific Computation, CRC Press, Boca Raton.

BENHAMOU F. & COLMERAUER A. (eds.), [1993], Constraint Logic Programming, Selected Research, The MIT Press, Cambridge, Mass.

BERRY J.S. & BURGHES D.N. & HUNTLEY I.D. & JAMES D.J.G. & MOSCARDINI A.O. (eds.), [1986], Mathematical Modelling Metholology, Models and Micros, Ellis Horwood, Chichester.

BEILBY M.H. & MOTT T.H., [1983], Academic Library Acquisitions, Allocation based on Multiple Collection Development Goals, in: Comput. & Ops. Res. Vol. 10, No. 4, pp 335–343.

BHARGAVA H.K. & KIMBROUGH S.O., [1993], Model Management, An embedded languages approach, in: Decision Support Systems, Vol. 10, pp. 277–299.

BHARGAVA H.K. & KIMBROUGH S.O., [1995], On Embedded Languages, Meta-Level Reasoning, and Computer-Aided Modeling, in: [Nash al. 1995], pp. 27–44.

BIRGE J.R. & WETS R. J-B. (eds), [1991], Stochastic Programming – Part I, Annals of Operations Research (ed-in-chief: Hammer P.L.), Vol 30, J.C. Baltzer AG, Basel.

BIRGE J.R. & WETS R. J-B. (eds), [1991], Stochastic Programming – Part II, Annals of Operations Research (ed-in-chief: Hammer P.L.), Vol 31, J.C. Baltzer AG, Basel.

BISSCHOP J. & ENTRIKEN R., [1993], AIMMS, The Modeling System, Paragon Decision Technology B.V.

BISSCHOP J.J., [1988], Language Requirements for a Priori Error Checking and Model Reduction in Large-Scale Programming, in: Mathematical Models for Decision Support, ed. G. Mitra, NATO ASI Series F. Vol:48, pp. 170–181.

BISSCHOP J. & MEERAUS A., [1982], On the Development of a General Algebraic Modeling System in a Strategic Planning Environment, in: Math. Programming Studies, Vol. 20, pp. 1–29.

BLANNING R.W. & HOLSAPPLE C.W. & WHINSTON A.B. (eds.), [1993], Special Issue on Model Management Systems, in: Decision Support Systems, Vol. 9, No. 1 (Jan.).

BLUM W. & BERRY J.S. & BIELER R. & HUNTLEY D. & KAISER-MESSMER G. & PROFKE L. (eds.), [1989], Applications and Modelling in Learning and Teaching Mathematics, Ellis Horwood, Chichester.

BOOK R.V., [1991], Rewriting Techniques and Applications, 4th International Conference, RTA–91, Como, Italy, April 1991, Proceedings, Springer, Berlin.

BOWMAN J.S. & EMERSON S.L. & DARNOVSKY M., [1996], The Practical SQL Handbook, 3th edition, Addison-Wesley Developers Press, Reading, Massachusetts.

BOYCE W.E. (ed.), [1980], Case Studies in Mathematical Modeling, Pitman Advanced Publ. Program, Boston.

BRADLEY G.H. & CLEMENCE R.D., [1987], A Type Calculus for Executable Modeling Languages, IMA Journal of Mathematics in Management, Vol 1(1987) pp. 177–191.

BRAMS S.J. & LUCAS W.F. & STRAFFIN P.D. Jr. (eds.), [1983], Political and Related Models, (Modules in Applied Mathematics, Vol 2), Springer, New York.

BRAUN M. & COLEMAN C.S. & DREW D.A. (eds.), [1983], Differential Equation Models, (Modules in Applied Mathematics, Vol 1), Springer, New York.

BROOKE A. & KENDRICK D. & MEERAUS A., [1988], GAMS, A User's Guide, The Scientific Press.

BROWN R.G. & CHINNECK J.W. & KARAM G.M., [1989], Optimization with Constraint Programming Systems, in: Sharda al. (eds), p. 463– 473.

BUCHBERGER B., [1985], Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, in: Bose N. (ed), Multidimensional Systems Theory, D. Reidel Publ. Comp., Dordrecht, p. 184–232.

BUDD T.A., [1994], Multiparadigm Programming in LEDA, Addison-Wesley Publ., Reading, Massachusetts.

CALVERT J.E. & VOXMAN W.L., [1989], Linear Programming, Harcourt Brace Jovanovich, Publishers, Academic Press, San Diego.

CASTI J., [1996], Die grossen Fünf. Mathematische Theorien, die unser Jahrhundert prägten, Birkhäuser, Basel.

CHANDRASEKHAR S., [1987], Truth and Beauty, Aesthetics and Motivations in Science, The University of Chicago Press, Chicago.

CHAR B.W. et al., [1991], Maple V Language/Reference Manual, Springer Verlag, New York (see also the WWW page: http://www.maplesoft.com/).

CHOKSI A.M. & MEERAUS A. & STOUTJESDIJK, The Planning of Investment Programs in the Fertilizer Industry, Johns Hopkins University Press, Batimore, 1980.

CHVATAL V., [1973], Linear Programming, W.H. Freeman Company, New York.

CLEMENTS R.R., [1989], Mathematical Modelling, A case study approach, Cambridge University Press, Cambridge.

CHINNECK J. & DRAVNIEKS E, [1991], Locating Minimal Infeasible Constraint Sets in Linear Programs, in: ORSA Journal on Computing, Vol 3, Nr 2, pp 157–168.

COHEN J., [1994], Automatic Identification and Data Collection Systems, McGraw-Hill, London.

COHEN J., [1990], Constraint logic programming languages, in: Communications of the ACM, 33(7), pp 52–68.

COLMENAUER A., [1990], An Introduction to PROLOG-III, in: Communications of the ACM, 33(7), pp 69–90.

COLLAUD G., [1993], Modélisation et optimation linéaire, un système graphique de création et de gestion des modèles (gLPS), Thèse, Université de Fribourg, Suisse.

COLLAUD G. & PASQUIER J., [1996], gW: un environnement hypertexte programmable comme support de cours pour l'optimisation linéaire, in: Revue Informatique et Statistique dans les Sciences Humaines, No. 4 (décembre), p. 225–241.

COLLAUD G. & PASQUIER J., [1994], gLPS: a Graphical Tool for the Definition and Manipulation of Linear Problems, in: European Journal of Operations Research, Vol. 72:2, p. 277–286.

COULLARD C. & FOURER R., [1995], Interdependence of Methods and Representations in Design of Software for Combinatorial Optimization, Technical Report 95-67, Industrial Engineering and Management Sciences, Northwestern University.

CPLEX, CPLEX Optimization, Inc., (version 4, 1996), see at info@cplex.com or http://www.cplex.com/.

CROSS M. & MOSCARDINI A.O., [1985], Learning the Art of Mathematical Modelling, Ellis Horwood Lim., Chichester.

CUNNINGHAM K. & SCHRAGE L, [1989], The LINGO Modeling Language, University of Chicago, Preliminary.

CZYZAK P. & SLOWINSKI R., [1989], Multiobjective Diet Optimization Problem under Fuzziness, in: The Interface between Artificial Intelligence and OR in Fuzzy Environment, Verdegay/Delgado (eds.), TÜV Rheinland, Köln.

DAVIS M.D. & WEYUKER E. J., [1983], Computability, Complexity, and Languages, Fundamentals of Theoretical Comptuer Science, Academic Press, New York.

DAVIS L. (ed.), 1991], Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York.

De FINETTI B., [1981], Wahrscheinlichkeitstheorie, Oldenbourg, München.

De KLEER J., [1986], An Assumption-based TMS. in: Artificial Intelligence, Elsevier Science Publ. B.V., Amsterdam, Vol. 28, pp. 127–162.

DENNIS S. & CATHY L., [1995], Out of their Minds, The Lives and Discoveries of 15 Great Computer Scientists, Copernicus (Spinger), New York.

DINCBAS M. & SIMONIS H. & Van HENTENRYCK P., [1990], Solving Large Combinatorial Problems in Logic Programming, in: J. Logic Programming, Vol. 8, p. 75–93.

DINCBAS M. & Van HENTENRYCK P. & SIMONIS H. & AGGOUN A. & GRAF T. & BERTHIER F., [1988], The constraint programming language CHIP, in FGCS'88, p. 693–710.

DOLK D., [1988], Model Management Systems for Operations Research: A Prospectus, in: [Mitra 1988, pp. 347–373].

DOLK D.R. & KONSYNSKI B.R., [1984], Knbowledge Representation for Model Management, in: IEEE Transaction Software Eng., SE-10.

DOUANYA NGUETSE G-B. & HANSEN P. & JAUMARD B., [1994], Probabilistic Satisfiability and Decomposition, Working Paper no 94-26, December, Institute of Informatics, University of Fribourg (Switzerland).

DREIHELLER A. & MOERSCHBACHER M. & MOHR B., [1986], Programming Pascal with Physical Units, Sigplan Notices, Vol. 21, No.12, December 1986, pp. 114–123.

DYM C.L. & IVEY E.S., [1980], Priciples of Mathematical Modeling, Academic Press, New York.

ELLISON E.F.D. & MITRA G., [1982], UIMP: User Interface for Mathematical Programming, ACM Transactions on Mathematical Software, Vol.8, No.3, September 1982, pp. 229–255.

ERMOLIEV Y. & WETS R. J-B., [1988], Numerical Techniques for Stochastic Optimization, Springer, Berlin.

ERSOY Y. & MOSCARDINI A.O. (eds.), [1994], Mathematical Modelling Courses for Engineering Education, Springer, Berlin.

EVES H., [1992], An Introduction to the History of Mathematics, sixth edition, The Saunders Series, Fort Worth.

FEIGENBAUM E.A., [1996], How the “What” Becomes the “How”, in: Communications of the ACM, Vol. 39, No. 5, pp 97–104.

FORSTER M. & MEVERT P., [1994], A Tool for Network Modeling, in: EJOR, Vol. 72(2), pp. 287–299.

FOURER R., [1995], What's new in standard AMPL?, in: Compass News, Issue No. 1, Fall 1995, 1005 Terminal Way, Suite 100, Reno, NV 89502.

FOURER R., [1993], Database Structure for Mathematical Programming Models, Technical Report 90-06, (revised November 1993), Industrial Engineering and Management Sciences, Northwestern University.

FOURER R. & GAY D.M. & KERNIGHAN B.W., [1990], A Modeling Language For Mathematical Programming, in: Management Science, Vol. 36, p. 519–554.

FOURER R. & GAY D.M. & KERNIGHAN B.W., [1993], AMPL, A Modeling Language For Mathematical Programming, The Scientific Press, San Francisco.

FOURER R., [1983], Modeling Languages Versus Matrix Generators for Linear Programming, in: ACM Transactions on Mathematical Software, Vol. 9, No. 2, June 1983, pp. 143–183.

FRÜHWIRTH T. & HEROLD A. & KÜCHENHOFF V. & LE PROVOST T. & LIM P. & MONFROY E. & WALLACE M., [1992], Constraint Logic Programming, An Informal Introduction, in: Comyn G., Fuchs N.E., Ratcliffe M.J., (eds), Logic Programming in Action, Proceedings of the Second International Logic Programming Summer School, LPSS '92, Zürich, Springer Lecture Notes in AI No 636, pp 3–35.

GARDNER M., [1991], The Unexpected Hanging and Other Mathematical Diversions, The University of Chicago Press, Chicago.

GARDNER M., [1988], Time Travel and other Mathematical Bewilderments, W.H. Freeman and Company, New York.

GEIGER K., [1995], Inside ODBC, Microsoft Press, Redmond.

GEOFFRION A.M., [1987], An Introduction to Structured Modeling, in: Management Science, Vol. 33, No. 5 (May), pp. 547–588.

GEOFFRION A.M., [1988], The Formal Aspects of Structured Modeling, in: Operations Research, Vol. 37, No. 1, pp. 30–51.

GEOFFRION A.M., [1989], Computer-based Modeling Environments, in: European Journal of Operational Research, 41, pp. 33–45.

GEOFFRION A., [1989a], SML: A Model Definition Language for Structured Modeling, Western Management Science Institute, University of California, Los Angeles, Working Paper No.#360, revised Nov. 1989.

GEOFFRION A.M., [1994], Structured Modeling: Survey and Future Research Directions, in: ORSA CSTS Newsletter, Vol. 15, No. 1, Spring 1994.

GEOFFRION A.M., [1995], An Informal Annotated Bibliography on Structured Modeling, Working Paper No. 390, Western Management Science Institute, University of California, LA, Revised October 1995.

GERTEN R. & JACOB U., [1989], Ein Report-Generator für eine Softwareentwicklungsumgebung, Interner Bericht 191, Fachbereich Informatik, Universität Kaiserslautern.

GLASS R.L., [1996], The Relationship Between Theory and Practice in Software Engineering, in: Communication of the ACM, Vol. 39,11, November, 1996.

GLOVER F. & KLINGMAN D. & McMILLAN C., [1977], The NETFORM concept: A More Effective Model Form and Solution Procedure for Large Scale Nonlinear Problems, in: Annual Proceedings of the ACM, Oct. 16–19, 1977, pp. 283–289.

GLOVER F., [1993], Tabu Search, Annals of Operations Research, Vol. 41, Baltzer, Basel.

GOLDBERG D.E., [1989], Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley Publ., Mass.

GÖRZ G. (Hrsg.), [1993], Einführung in die künstliche Intelligenz, Addison-Wesley, Bonn.

GREGORY J. W., (jwg@cray.com), Linear Programming FAQ, Usenet sci.answers. Available via anonymous FTP from rtfm.mit.edu in /pub/usenet/sci.answers/linear-programming-faq

GREENBERG H.J., [1981], Graph-Theoretic Foundations of Computer-Assisted Analysis, in: Greenberg H.J. & Maybee J.S., (eds.), Computer Assisted Analysis and Model Simplification, Academic Press, New York, 1981, p. 481–495.

GREENBERG H.J., [1995], A Bibliography for the Development of An Intelligent Mathematical Programming System, Working Paper, University of Colorado, Center for Computational Mathematics, UCD/CCM Report No. 59, July 1995. (see latest version in url http://www-math.cudenver.edu/~hgreenbe/impsbib.html).

GREENBERG H.J., [1995a], A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User's Guide for ANALYZE, November 1995, Mathematics Department, University of Colorado at Denver.

GREENBERG H.J., [1994], Syntax-directed Report Writing in Linear Programming using ANALYZE, in: EJOR Vol. 72(2), pp. 300–311.

GREENBERG H.J. & MURPHY F.H., [1995], Views of Mathematical Programming Models and their Instances, in: Desicion Support Systems, Vol. 3, No. 1, pp. 3–34.

HAILPERIN T., [1986], Boole's Logic and Probability, 2nd ed., North-Holland, Amsterdam.

HAMMER M. & CHAMPY J., [1994], Reengineering the Corporation, HarperBusiness.

HAMMER R. & HOCKS M. & KULISCH U. & RATZ D., [1993], Numerical Toolbox for Verified Computing I, Basic Numerical Problems, Springer, Berlin.

HANSEN P. & JAUMARD B. & POGGI M. de ARAGÃO P., [1991], Mixed-Integer Column Generation Algorithms and the Probabilistic Maximum Satisfiability Problem, Working Paper no G–91-53, December, GERAD, École des Hautes Études Commerciales, École Polytechnique, Université McGill, Montréal (Québec), Canada.

HANSEN P. & JAUMARD B. & DOUANYA NGUETSE G-B-. & de ARAGÃO P., [1995], Models and Algorithms for Probabilistic and Bayesian Logic, Working Paper no 95-01, January, Institute of Informatics, University of Fribourg (Switzerland).

HÄTTENSCHWILER P., [1988], Goal Programming becomes most useful using L_1–smoothing functions, in: Computational Statistics & Data Analysis 6, pp. 369–383.

HEIDEMAN M.T. & JOHNSON D.H. & BURRUS C.S., [1985], Gauss and the History of the Fast Fourier Transform, in: Archive for History of Exact Sciences, Vol 34, pp. 265–277.

HERFEL W.E. & KRAJEWSKI W. & NIINILUOTO I. & WOJCICKI R. (eds.), [1995], Theories and Models in Scientific Processes, (Proceedings of the AFOS '94 Workshop, August 15–26, Madralin and IUHPS '94 Conference, August 27–29, Warszawa), Rodopi, Amsterdam.

HOARE C.A.R., [1974], Hints on Programming Language Design, in: Bunyan C.J., (ed.), State of the Art Reprt 20: Computer Systems Reliability, Pergamon/Infotech, p. 505–534, reprinted also in: [Hoare/Jones 1989, Chapter 13].

HOARE C.A.R. & Jones C.B., [1989], Essays in Computing Science, Prentice Hall, New York.

HODGES J.S. & DEWAR J.A., [1992], Is it You or Your Model Talking? A Framework for Model Validation, RAND Publication Series: R-4114-AF/A/OSD, 1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138.

HOFSTADTER D.R., [1988], Metamagicum, Fragen nach der Essenz von Geist und Struktur, Klett-Cotta, Stuttgart.

HOOKER J.N., [1988], A Quantitative Approach to Logical Inference, in: Workshop "Mathematics and AI", Vol II, FAW Ulm, 19th–22nd December 1988, p.289–314. (reprinted in: DSS 4 (1988), p.45–69).

HOUSE R.T., [1983], A Proposal for an Extended Form of Type Checking of Expressions, The Computer Journal, Vol. 26, No. 4, 1983, pp. 366–374.

HÜRLIMANN T., [1997a], Reference Manual for the LPL Modeling Language, Working Paper, Version 4.01, December 1996, Institute of Informatics, Universit of Fribourg, (newest version is always on: ftp://ftp-iiuf.unifr.ch/pub/lpl/Manuals, file Manual.ps).

HÜRLIMANN T., [1997b], Index-sets in Modeling Languages, Working Paper, forthcoming, April 1997, Institute of Informatics, University of Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/Manuals, file Indexset.ps) (forthcoming).

HÜRLIMANN T., [1997c], IP, MIP and Logical Modeling Using LPL, Working Paper, April 1997, Institute of Informatics, University of Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/Manuals, file Mip.ps).

HÜRLIMANN T., [1997d], LPL DLL Guide, Working Paper, January 1997, Institute of Informatics, University of Fribourg, ftp://ftp-iiuf.unifr.ch/pub/lpl/Manuals, file LPLDLL4.ps).

HÜRLIMANN T. (guest editor), [1994], Software Tools for Mathematical Programming, EJOR, feature issue, Vol 72 (1994), No 2, North-Holland.

HÜRLIMANN T., [1991], Units in LPL, Institute of Informatics (formerly: Institute for Automation and Operations Research), Working Paper No. 182, April 1991, (last update August 95), Fribourg.

HÜRLIMANN T., [1987], LPL: A Structured Language for Modeling Linear Programs, Disseration, Peter Lang, Bern.

IBM, [1988], Mathematical Programming System Extended/370 (MPSX/370), Version 2, Program Reference Manual.

IGNIZIO J.P. & CAVALIER T.M., [1994], Linear Programming, Prentice Hall, Englewood Cliffs, NJ.

IGNIZIO J.P., [1976], Goal Programming and Extensions, Lexington Bokks, D.C. Health and Comp., Toronto.

JACOBY S.L.S. & KOWALIK J.S., [1980], Mathematical Modeling with Computers, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

JAFFAR J. & MAHER M.J., [1996], Constraint Logic Programming: A Survey, (to appear), (a final draft can be downloaded from *pop.cs.cmu.edu* in the directory: */usr/joxan/public*).

JAFFAR J. & MICHAYLOV S. & STUCKEY P.J., [1992], The CLP(R) Language and System, in: ACM Transactions on Programming Languages and Systems, Vol. 14, No. 3, July, p. 339–395.

JENKS R.D. & SUTOR R.S., [1992], AXIOM, The Scientific Computation System, NAG and Springer, New York.

JEROSLOW R. G. (ed.), [1987], Approaches to Intelligent Decision Support, Annals of Operations Research Vol. 12, Baltzer; Basel.

JEROSLOW R.G., [1989], Logic-Based Decision Support, Mixed Integer Model Formulation, Annals of Discrete Mathematics Vol 40, North-Holland, Amsterdam.

JOHNSON E.L., [1989], Modeling and Strong Linear Programs for Mixed Integer Programming, in: Algorithms and Model Formulations in Mathematical Programming, ed. Stein W.W., NATO ASI Series F, Vol. 51, Springer, pp. 1–43.

JONES C.V., [1990], Anintroduction to Graph-Based Modeling Systems, Part I: Overview, in: ORSA Journal of Computing, Vol. 2, No. 2, pp. 136–151.

JONES C.V., [1991], Anintroduction to Graph-Based Modeling Systems, Part II: Graph-Grammars and the Implementation, in: ORSA Journal of Computing, Vol. 3, No. 3, pp. 180–207.

JONES C.V., [1996], Visualization and Optimization, Kluwer Academic Publishers, Boston.

KALL P., [1976], Stochastic Linear Programming, Springer, Berlin.

KALL P. & MAYER J., [1993], SLP-IOR: An Interactive Model Management System for Stochastic Linear Programs, IOR, University of Zurich, Switzerland, (submitted to Math Programming Ser. B).

KALL P., [1992], A Model Management System for Stochastic Linear Programming, in: Kall P. (ed), System Modelling and Optimization, Springer, 1992, pp 580–587.

KALL P. & WALLACE S.W., [1994], Stochastic Programming, John Wiley & Sons, Chichster.

KAPUR J.N., [1979], Mathematical Modelling, Its Philosophy, Scope, Powers and Limitations, in: Bulletin of the Mathematical Association of India, Vol. XI, No 3, 1979, pp. 69–112.

KARR M. & LOVEMAN D.B., [1978], Incorporation of Units into Programming Languages, Comm. of the ACM, May 1978, Vol. 21, No.5, pp. 385–391.

KENDRICK D.A. & LASDON L.S. & RUEFLI T.W. (eds.), [1989], Integrated Modeling Systems: AI in the Business and Economic Context, in: Computer Science in Economics and Management, Vol. 2 No. 1, 1989,

KING A.J., [1988], Stochastic Programming Problems: Examples from the Literature, in: Ermoliev/Wets, pp 543–567.

KNUTH D.E., [1996], Selected Papers on Computer Science, CSLI Publications, Cambridge University Press.

KNUTH D.E., [1986], The Texbokk, Addison Wesley, Reading, Massachusetts.

KNUTH D.E., [1984], Literate Programming, in: Computer Journal, Vol. 27, 2 (May), pp. 97–111.

KNUTH D.E., [1976], Ancient Babylonian Algorithms, in: [Knuth 1996, Chapter 11].

KNUTH D.E. & LEVY S., [1994], The CWEB System of Structured Documentation, Addison-Wesley Publ. Comp., Reading, Massachusetts.

KOHLAS J. & MONNEY P.-A., [1994], Theory of Evidence – A Survey of its Mathematical Foundations, Applications and Computational Aspects, in: ZOR – Mathematical Methods of Operational Research, (1994) Vol 39, p. 35–68.

KOHLAS J. & MONNEY J-P., [1995], A Mathematical Theory of Hints, An Approach to the Dempster-Shafer Theory of Evidence, Springer, Lecture Notes in Economics and Mathematical Systems, Vol. 425, Berlin.

KUHN T.S., [1962], The Structure of Scientific Revolutions, University of Chigago Press, Chigago.

KUIP C.A.C., [1992], Index Sets in Mathematical Programming Modeling Languages, Proefschrift, Universiteit Twente, Netherland.

KUMAR V., [1992], Algorithms for Constraint-Satisfaction Problems: A Survey, in: AI Magazine, Spring 1992, p. 32–44.

LAHDELMA R. & RUUTH S., [1995], An Object-Oriented Mathematical Modelling Language in C++ — Proposal for a New Standard, Talk at:the Applied Mathematical Programming and Modelling Conference, Brunel University, 3–5 April 1995.

LELER W., [1988], Constraint Programming Languages: Their Specification and Generation, Addison-Wesley, Readings, Mass.

LENARD M.L., [1988], Structured Model Management, in: [Mitra 1988, pp. 375–391].

Lindo Systems Inc., [1995], Solver Suite, Lindo, Lingo, What's Best!, 1415 North Dayton Street, Chicago, Illinois 60622, email: tech@lindo.com.

LOECKX J. & EHRLICH H.D. & WOLF M., [1996], Specification of Abstract Data Types, Wiley/Teubner, Chichester.

LUCAS W.F. & ROBERTS F.S. & THRALL R.M. (eds.), [1983], Discrete and System Models, (Modules in Applied Mathematics, Vol 3), Springer, New York.

MANKIN R., [1987], (Letter), Sigplan Notices, Vol. 22, No. 3, March, p. 13.

MÄNNER R., [1986], Strong Typing and Physical Units, Sigplan Notices, Vol. 21, No. 3, March 1986, pp. 11–20.

MANNA Z., [1974], Mathematical Theory of Computation, McGraw-Hill Book Company, New York.

MARCUS-ROBERTS H. & THOMPSON M. (eds.), [1983], Life Science Models, (Modules in Applied Mathematics, Vol 4), Springer, New York.

MatLab, [1993], User's Guide, The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760. (see also the page: http://www.mathworks.com).

MAYOH B. & TYUGU E. & PENJAM J. (eds), [1994], Constraint Programming, NATO ASI Series F: Computer and Systems Sciences, Vol. 131, Springer, Berlin.

McALOON K. & TRETKOFF C., [1995], 2LP: Linear Programming and Logic Programming, in: Saraswat/van Hentenryck 1995, pp. 101–116.

MESTERTON-GIBBONS M., [1989], A Concrete Approach to Mathematical Modelling, Addison-Wesley Publ., Redwood City.

MEYER B., [1997], Object-oriented Software Construction, 2nd edition, Prentice Hall, New York.

MEYER B., [1992], Eiffel: The Language, Prentice Hall, Object-Oriented Series, New York.

MITCHELL G., [1993], The Practice of Operational Research, John Wiley] Sons, Chichester.

MITRA G. (ed.), [1988], Mathematical Models for Decision Support, NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 48, Springer, Berlin.

MITRA G. & LUCAS C. & MOODY S., [1995], Sets and Indices in Linear Programming Modelling and their Integration with Relational Data Models, in: Computational Optimization and Applications, Vol. 4, p. 263–283.

MITRA G. & LUCAS C. & MOODY S., [1994], Tools for reformulation logical forms into zero–one mixed integer programs, in: European Journal of Operational Research, Vol. 72,2, North-Holland.

MÖSSENBÖCK H., [1994], Objektorientierte Programmierung in Oberon-2, 2. Auflage, Springer, Berlin.

MOUSAVI H. & MITRA G. & LUCAS C., [1995], Data and Optimisation Modelling: A Tool for Elicitation and Browsing (DOME), Working Paper TR/11/95, Brunel University, Uxbridge, Middlesex, UK, June, 1995.

MÜLLER-MERBACH H., [1990], Database-Oriented Design of Planning Models, in: IMA Journal of Mathematics Applied in Business & Industry, Vol. 2, pp. 141–155.

NAG library, [1996], see the WWW page: http://www.nag.co.uk/.

NASH S.G. & SOFER A., [1996], Linear and Nonlinear Programming, The McGraw-Hill Companies, Inc., New York.

NASH S.G. & SOFER A. (eds.), [1995], The Impact of Emerging Technologies on Computer Science and Operations Research, Kluwer Acad. Publ., Boston.

NAUR P., [1981], The European Side of the Last Phase of the Development of ALGOL 60, in: History of Programming Languages, Wexelblat R.L. (ed.), (from the ACM SIGPLAN History of Programming Languages Conference, June 1–3, 1978), Academic Press.

NEELAMKAVIL F., [1987], Computer Simulation and Modelling, John Wiley & Sons, Chichester.

NEMHAUSER G.L. & WOLSEY L.A., [1988], Integer and Combinatorial Optimization, Wiley.

NEWELL A. & SIMON H.A., [1972], Human Problem Solving, Prentice-Hall Inc., Englewood Cliffs.

NIELSEN S.S., [1995], A C++ Class Library for Mathematical Programming, in: [Nash al. 1995], pp. 221–243.

NILSSON N.J., [1986], Probabilistic Logic, Artificial Intelligence, 28 (1986), pp.71–87.

OLDER W.J. & BENHAMOU F., [1996], Programming in CLP(BNR), papar available from benham@gia.univ-mrs.fr.

OTTEN R.H.J.M. & Van GINNEKEN L.P.P.P., [1989], The Annealing Algorithm, Kluwer Academic Publishers, Boston.

PALMER K.H., [1984], A Model-Management Framework for Mathematical Programming, An Exxon Monograph, John Wiley & Sons, New York.

PARNAS D.L., [1972], On the Criteria to Be Used in Decomposing Systems into Modules, in: Communications of the ACM, Vol. 15, 12, pp. 1059–1062.

PARRELLO B.D. & KABAT W.C. & WOS L., [1986], Job-Shop Scheduling Using Automated Reasoning: A Case Study of the Car-Sequencing Problem, in: Journal of Automated Reasoning, Vol. 2, p. 1–42.

PASQUIER J. & MONNARD J., [1995], Livres électronique, de l'utopie à la réalisation, Presses Polytechniques et Universitaires Romandes, Lausanne.

PASQUIER J. & HÄTTENSCHWILER P. & HÜRLIMANN T. & SUDAN B., [1986], A Convenient Technique for Constructing Your Own MPSX Generator Using dBASEII, in: Angewandte Informatik, Vol. 7, pp. 295–300.

PAWLAK Z., [1982], Rough Sets, in: International Journal of computer and Information Sciences, 11 (1982), pp. 341–356.

PEARL J., [1988], Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA.

POLYA G., [1962], Mathematical Discovery, On understanding, learning, and teaching problem solving, Vol I+II, John Wiley & Sons, New York.

POLYA G., [1954], Mathematics and Plausible Reasoning, Volume I: Induction and Analogy in Mathematics, Princeton University Press, New Jersey, 12th printing, 1990.

POLYA G., [1954a], Mathematics and Plausible Reasoning, Volume II: Patterns of Plausible Inference, Princeton University Press, New Jersey, 12th printing, 1990.

POLYA G, [1945], How To Solve It, (new edition with a preface by I. Stewart), Penguin Books 1990, London.

POLYA G. & TARJAN R.E. & WOODS D.R., [1983], Notes on Introductory Combinatorics, Birkhäuser, Boston.

POPPER K., [1976], Logik der Forschung, sechste, verb. Auflage, J.C.B. Mohr (Paul Siebeck) Tübingen.

POSWIG J., [1996], Visuelle Programmierung, Carl Hanser Verlag, München.

RABINOVICH L.M., [1992], Mathematical Modelling of Chemical Processes, CRC Press, Boca Raton.

RAVINDRAN A. & PHILLIPS D.T. & SOLBERG J.J., [1987], Operations Research, Principles and Practice, 2ed, John Wiley] Sons, New York.

REEVES C.R. (ed), [1993], Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons, New York.

REISER M., [1992], Programming in Oberon, Steps Beyond Pascal and Modula, Addison-Wesley, New York.

RICH E. & KNIGHT K., [1991], Artificial Intelligence (2nd edition), International edition, McGraw-Hill, Inc., New York.

RIVETT P., [1980], Model Building for Decision Analysis, Wiley, Chichester.

ROBERTS F.S., [1976], Discrete Mathematical Models, With applications to social, biological, and environmental problems, Prentice Hall, Englewood Cliffs, New Jersey.

ROBINSON J.A., [1965], A Machine-Oriented Logic Based on the Resolution Principle, in: Journal of the ACM, Vol 12 pp. 163–174.

ROMMELFANGER H., [1993], Fuzzy Decision Support-Systeme, Entscheiden bei Unschärfe, (2. Auflage), Springer-Lehrbuch, Berlin.

ROZENBERG G. & SALOMAA A., [1994], Cornerstones of Undecidability, Prentice Hall, New York.

SARASWAT V. & Van HENTENRYCK P. (eds.), [1995], Principles and Practice of Constraint Programming, The MIT Press, Cambridge.

SHAFER G., [1976], A mathematical theory of evidence, Princeton University Press, Princeton.

SHAFER G., [1978], Non-Additive Probabilities in the Work of Bernoulli and Lambert, in: Archives for the History of Exact Sciences, 19, p309–370.

SCHIFFER M.M. & BOWDEN L., [1984], The Role of Mathematics in Science, The Mathematical Association of America.

SCHNIEDERJANS M.J., [1995], Goal Programming, Metholology and Applications, Kluwer Acad. Publ., Boston.

SENGUPTA J.K., [1972], Stochastic Programming, Methods and Applications, North-Holland, Amsterdam.

SHARDA R. & GOLDEN B.L. & WASIL E. & BALCI O. & STEWART W. (eds.), [1989], Impacts of Recent Computer Advances on Operations Research, North-Holland, New York.

SHARDA R. & RAMPAL G., [1995], Algebraic Modeling Languages on PC's, OR/MS Today, June 1995.

SHETTY B. & BHARAGAVA H.K. & KRISHNAN R. (eds.), [1992], Model Management in Operations Research, in: Annals of Operations Research, Vol. 38 (1992), No. 1–4, J.C. Baltzer AG, Basel.

SOBER E., [1975], Simplicity, Clarendon Press, Oxford.

STACHOWIAK H. (Hrsg.), 1983], Modelle – Konstruktion der Wirklichkeit, Wilhelm Fink Verlag, München.

STEIGER D. & SHARDA R. & LeCLAIRE B., [1992], Functional Description of a Graph-Based Interface for Network Modeling (GIN), in: Balci 1992 pp. 213–229.

STERLING L. & SHAPIRO E., [1986], The Art of Prolog: Advanced Programming Techniques, The MIT Press, Cambridge, Mass.

STÖCKLE D. & POLSTER F.J., [1980], Die Report-Definitionssprache RDL und ihre Implementierung im Report-Generator FAREG, KfK 2910, Institut für Datenverarbeitung in der Technik, Kernforschungszentrum Karlsruhe.

STRANG G., [1988], Linear Albegra and its Applications, Third Edition, Harcourt Brace Jovanovich, Publ. San Diego.

SUNSET SOFTWARE, XA, A Professional Linear and Mixed 0/1 Integer Programming System, 1613 Chelsea Road, Suite 153, San Marino, California 91108, phone: 818-441-1565, fax: 818-441-1567.

TAMIZ M. & MARDLE S.J. & JONES D.F., [1995], Detecting IIS in Infeasible Linear Programmes using Techniques from Goal Programming, Talk presented at the APMOD 95 conference, 3–5 April 1995, at the Brunel University, West London

TARSKI A., [1994], Introduction to Logic and to the Methodology of the Deductive Sciences, 4rd edition, Oxford University Press.

THOMPSON J.R., [1989], Empirical Model Building, Wiley, New York.

TOGAI InfraLogic Inc., [1991], Fuzzy-C Development System, User's Manual, 30 Corporate Park, Suite 107, Irvine, CA 92714.

TSANG E., [1993], Foundations of Constraint Satisfaction, Academic Press, London.

ULLMAN J.D., [1988], Principles of Database and Knowledge-Base Systems, Volume I: Classical Database Systems, Computer Science Press, Rockville.

Van HENTENRYCK P., [1989], Constraint satisfaction in logic programming, Logic Programming Series, MIT Press.

Van LAARHOVEN P. J. M. & AARTS E. H. L., [1987], Simulated Annealing: Theory and Applications, Dordrecht, Boston.

YAGLOM I.M., [1986], Mathematical Structures and Mathematical Modelling, Gordon and Breach Science Publ., New York, (translated from the Russian by D. Hance, original edition 1980).

YAMAKAWA T., [1989], Stabilization of an Inverted Pendulum by a High-speed Fuzzy Logic Controller Hardware System, in: Fuzzy Sets and Systems Vol 32, p.161ff.

WEGNER P., [1997], Why Interaction Is More Powerful Than Algorithms, in: Communications of the ACM, Vol. 40, No. 5, p. 80–91.

WELSH J.S. Jr., [1987], PAM – a Practitionar's Approach to Modeling, in: Management Science, Vol. 33, p. 610–625.

WETS R.J-B., [1991], Stochastic Programming, in: Handbooks in Operations Research and Management Science, Vol 1, Optimization (eds. Nemhauser G.L., Rinnooy Kan A.H.G., Todd M.J.), North-Holland, Amsterdam.

WIDMER M. & BUGNON B. & VARONE S. & HERTZ A., [1997], Rhythmed Flow-Shop: How to Balance the Daily Workload, in: the Proceedings of the IFAC/IFIP Conference on Management and Control of Production and Logistics, 31 August – 3 September 1997, Campinas, Brazil (to appear).

WIGNER E.P., [1960], The Unreasonable Effectiveness of Mathematics in the Natural Sciences, in: Communications on Pure and Applied Mathematics, Vol. XIII (1960), pp. 1–14.

WILLIAMS H.P., [1993], Model building in mathematical programming, 3rd edition, Wiley, Chichester.

WILLIAMS H.P. [1977], Logical problems and integer programming, Bull. Inst. Math. Appl., 13 (1977), pp.18–20.

WINSTON P.H., [1992], Artificial Intelligence (3rd edition), Addison-Wesley Publ., Reading, Mass.

WIRTH N., [1996], Grundlagen und Techniken de Compilerbaus, Addison-Wesley, Bonn.

WIRTH N, [1993], Recollactions about the Development of Pascal, in: ACM SIGPlan Notices, Vol 28,3, March 1993, pp. 333–342.

WIRTH N. & GUTKNECHT J., [1992], Project OBERON, The Design of an Operating System and Compiler, ACM Press, New York, Addison-Wesley Publ., Workingham.

WIRTH N, [1984], Compilerbau, Teubner Studienbücher, Informatik, Stuttgart.

WIRTH N., [1977], Modula – A Programming Language for Modular Multiprogramming, in: Software – Practice and Experience, Vol. 7(1977), p. 3–35.

WOLFRAM S., [1991], Mathematica, A System for Doing Mathematics by Computer, Second Edition, Addison-Wesley Publ. (see also in the WWW page: http://www.mathematica.com/).

ZADEH L.A., [1965], Fuzzy Sets, in: Information and Control Vol 8 (1965), pp. 338–353.

ZIMMERMANN H.J. & ZADEH L.A. & GAINES B.R. (eds.), [1984], Fuzzy Sets and Decision Analysis, in: Studies in the Management Sciences, Vol 20, North Holland, Amsterdam.

ZIMMERMANN H.J., [1987], Fuzzy Sets, Decision Making, and Expert Systems, Kluwer Acad. Press, Boston.

ZIMMERMANN H.J., [1991], Fuzzy Set Theory and its Applications, Second Edition, Kluwer Academic Publ., Boston.

ZIONTS S., [1988], Multiple Criteria Mathematical Programming: An Updated Overview and Several Approaches, in: Mathematical Models for Decision Support, Mitra G. (ed.), NATO ASI Series F, Vol 48, Springer Verlag.

GLOSSARY

The objective of this glossary is to summarize the vocabulary used in this book. It is confined to the "modeling language design community". The standard vocabulary of other scientific branches such as operations research, computer science, discrete mathematics, etc. has not be considered here. However, certain terms used also in computer science are so fundamental in the proposed framework that they have been entered the following list (example *algorithm*).

Several terms collide with well known concepts in mathematics or computer science. A prominent example is the term *variable* which is an unknown quantity in mathematics, while it is a place-holder of a memory location in computer science. (I use *memory variable* for the latter case.) Another example is the term *parameter*, which is a symbolic name for a known quantity in mathematics while in computer science it is a place-holder for passing values from and to procedures and functions. (I use *formal* and *actual parameter* in the latter cases.)

actual parameter

an argument in a calling procedure in a programming or modeling language (see also *formal parameter*; do not confound with *parameter*)

algorithm

a process, a procedure, a method or a recipe that expresses how a problem can be solved in a finite number of steps. Any algorithm can be represented by a Turing Machine (Church/Turing thesis)

algorithmic knowledge

the amount of information for a problem to represent its solution as an *algorithm* (see also *declarative knowledge*)

algorithmic part

the coded part in a modeling language which expresses the *algorithmic knowledge* of a model (see also *declarative part*)

alias (attribute)

an optional *attribute* defining additional *name attributes* for an *entity*

atom

an undecomposable item used as element which can be an identifier, a number, a string (or an expression returning an identifier, a number or a string)

attribute

a property of an *entity*. Each entity consists of a set of attributes; some are mandatory others are optional

arity (of a tuple)

the number of components forming a tuple

browser (of a model)

a tool that allows the *modeler* or the *model user* to skim through a particular *view* of the *model* (see also *editor* and *view*)

comment (attribute)

a text explaining a particular *entity*

component (in a tuple)

a part of a *tuple* representing an *element*

compound index-set

an *index-set* where all *elements* are *tuples* of the same *arity*

constraint

an Boolean expression containing variables that restricts the *state space* for a problem

data

see *model data*

declarative knowledge

the amount of information for a problem to represent it as a the subset of the *state space*. The subset is normally expressed by a set of *constraints*

declarative part

the coded part in a modeling language which expresses the *declarative knowledge* of a *model* (see also *algorithmic* or *procedural part*)

editor (of a model)

a tool that allows the *modeler* or the *model user* to skim through a particular *view* of the *model* and to modify, update or extend the model (see also *browser* and *view*)

element

an item in an *index-set* which can be an *atom* or a *tuple*

entity

an indivisible fragment of the *declarative part* of a *model*

formal parameter

an argument in the heading of a procedure that acts as a place-holder for passing values to and from the procedure (see also *actual parameter*; do not confound with *parameter*)

genus

the class to which an *entity* belongs (set, parameter, variable, etc.); it is a mandatory *attribute* for every entity

index

a symbolic name (an identifier) in an expression referring to an arbitrary element of an *index-set*

hierarchical index-set

a countable, normally finite collection of *elements*, where the elements are *atoms* or *tuples* of arbitrary *arity*

index-set

a countable, normally finite collection of *elements* (see also *simple*, *compound* and *hierarchical index-set*

index-tree

a graphical picture to represent a *(hierarchical) index-set*

instruction entity

one of the five following *entities*: Check, Read, Write, Transform, and Solve entity

label

the stamp (a *tuple*) of a node in the *index-tree*

memory variable

a symbolic place-holder for a memory location in a program (do not confound with *variable*)

modeling management system (MMS)

a (computer based) framework which allows the *modeler* to create and the *model user* to maintain and solve a *model*

modeling language

a machine readable language which allows a modeler to express *declarative* as well as *algorithmic knowledge* (see also *programming language*)

model (mathematical)

a formal representation expressing the *declarative* as well as the *algorithmic knowledge* of a problem, (see also *program*). From a purely mathematical point of view, a model only expresses the declarative knowledge (Chapter 2). In logic and model theory, a model is an valid interpretation (Chapter 2).

model data

the values (numerical or alphanumerical) assigned to parameters, index-sets and eventually to variables

model instance

a model where all parameters and index-sets are assigned by concrete *data* (see also *model structure*)

model structure

a model where the parameters and/or the index-sets have not yet assigned *data* (see also *model instance*)

model user

the “end-user” who uses and maintains the model in order to solve problems (do not confound with *modeler*)

modeler

the creator of a model. The modeler is for modeling what the programmer is for programming (do not confound with *model user*)

modeling

the process of creating a model which maps a certain problem

name (attribute)

an identifier naming an *entity*; it is an mandatory *attribute*

parameter

an *entity* representing a known quantity (see also *variable*; do not confound with *formal parameter* and *actual parameter*)

procedural part

see *algorithmic part*

program

a formal representation expressing an *algorithm* (see also *model*)

programming language

a machine readable language which allows a programmer to express the *algorithmic knowledge* of a problem (see also *modeling language*)

simple index-set

an *index-set* where the *elements* are *atoms*

solution (of a model)

an assignment of values to all *variables* such that the *constraints* are fulfilled. Note that this term normally expresses both things: the process of finding an assignment and the assignment itself.

solver

a program or an algorithm which finds a *solution* of a *model*

state space

the set of all values assignable to the *variables*

tag (of a component)

an identifier representing a *component*

tuple

an ordered sequence of *components* which is used as *elements* in *index-sets*

type (attribute)

an expression or an identifier specifying the domain of the value of an *entity*

unit (attribute)

an expression or an identifier specifying the measurement of the value of an *entity*

variable

an *entity* representing an unknown quantity (see also *parameter*; do not confound with *memory variable*)

view

an appearance, a showing or a presentation of certain aspects of a *model*. A model can have a multitude of views. They can be in textual, graphical, pictorial or any other form (see also *browser* and *editor*)

INDEX