Joint NCC & IIS Bull., Comp. Science, 16 (2001), 141–152

Model vs. Algorithm: change of paradigm in information technology

A. S. Narin'yani

Algorithm is one of the most fundamental concepts in information technology. Its absolutely dominant position reminds that of programming in machine codes during the era of the first computers: it appeared to be the only conceivable method to communicate with a machine, but its fate disproved this axiom. The same future undoubtedly awaits Algorithms as well, and we want to prove the thesis in this present paper:

(i) Although this notion appears to be natural, Algorithms are natural only for computers (and for professional programmers, of course). This is result of a fundamental difference between the traditional algorithmic approach and that based on the notion of a Model, which is the most natural way to work for a problem-oriented expert (engineer, economist, etc.)

(ii) In the nearest 10–15 years the fate of Algorithm will repeat that of machine-code programming: it will preserve its place only in the lowest and relatively thin layer of software technology.

(iii) The technology that will replace the current Algorithm-based paradigm already exists. It provides the user with qualitatively new capabilities: now the user can interact with his Models directly, without any mediums from Methods, Algorithms and Programming. This technology is a part of the constraint programming approach to the formation of a new IT paradigm.

Introduction

Model and Algorithm are the most fundamental notions not only in mathematics but also in the entire modern information technology. Their positions, however, are quite different. This becomes especially clear if we look at the example of computational mathematics: while Model is used here only as a formal description of the object of computations, Algorithm is the very basis of the computer process.

This is why we have chosen computational mathematics as a basis for all of the discussions in this paper. We focus on two closely related topics:

- extreme inadequacy of the current, Algorithm-oriented technology, unshakable though it seems, and
- the prospect of a cardinal change of paradigms, which transposes Model and Algorithm in a new concept of information processing, developed in the *constraint programming* approach and *subdefinite mod-*

els as a branch of constraint programming that was originated by our team in the early 80s [8, 9].

This paper is a revised version of the publication under the same title in the "Information Technologies" magazine, Moscow, 4, 1997 (in Russian)

1. A strange world of computational mathematics

1.1. At the beginning of the computer era, before the first high-level languages appeared, programming in machine codes seemed to be the only one conceivable way to communicate with computers. But only 10–15 years later programming in machine codes and assembly language kept their place only in operating systems, a very thin layer that is closest to the hardware.

The notion of Algorithm is the most fundamental one in the computational mathematics, data processing and software technology. This is a "notion forever": to use a computer means to program, and a program is an Algorithm. All attempts to rise to a declarative specification of the original task make this situation even more contrast. A really declarative statement of a problem has been possible only for problem-oriented packages, which are libraries of specialized Algorithms, containing sometimes many dozens of them, like most universal math solvers.

1.2. Thus, the notion of an Algorithm is the *alpha* & *omega* of the contemporary information technology. There is, however, another notion that is no less fundamental for the mathematics and applied sciences — the notion of a formal Model, i.e., a pair of unordered sets: a set of variables-parameters and a set of relations linking these variables. It is a fundamental concept for any science using mathematical techniques, because any attempt to work with a real phenomenon in precise terms should begin with constructing its formal Model.

The Algorithm is just a tool that implements *how* when you know the Method and need to define it for the computer. In the general case, however, you must decide *what* before you define *how*, i.e. you need to build a Model. This simple truth is obvious for any science that uses mathematics, perhaps with the exception of computer science. For the latter, the Model is a poor relative that is almost invisible in the shadow of His Majesty the Algorithm.

It is so understandable: the Algorithm is a practical tool and a key to computer (maybe even *the* key). Computational mathematics offers a set of such tools, which are the Methods it has succeeded to develop, but not the Problems it should learn to solve. Thus, the Model remains practically useless in applications, and you cannot apply the computer to Models to obtain results, except for some very special cases. This situation has led

to the current state, when Models are met only in theoretical research, and then as illustrations to the subject at hand.

2. Model vs. Algorithm

2.1. To compare the two notions, we begin with listing the most obvious differences between them.

MODEL	ALGORITHM
Declarative by definition	In a sense, an Algorithm is anti-declarative
Symmetric with respect to its parameters, because each of its variables is implicitly expressed via the other ones.	Divides its parameters into input and output ones, and the latter are explicitly computed from the former.
Defines implicitly the solutions to all computational problems related to the object of the Model.	Defines explicitly the solution to one problem whose relation to the real Object is not clear enough (see below).
May be subdefinite in the general case.	The notion of "subdefinite" does not seem to be applicable to the classical Algorithm.
Determines a space of solutions rather than one point, as well as many (not unique) alternative values for each of its variables.	Only a special form (interval Algorithms) can work with interval values of numbers.

These five issues reflect only the external, visible distinction between the two notions, which goes much deeper.

2.2. First of all, a formal Model is a special case of the general notion of a Model. The very semantics of the latter incorporate the object of modeling. Without answering the question *"The Model of what?"* the notion in itself is simply meaningless.

A mathematical Model of some real phenomenon is its formal approximation and with certain restrictions can be used instead of the phenomenon itself, in a computer study or analysis of its nature and behavior. In this capacity, the Model may serve as a basis in the solution of common computational problems, for example:

⇒ *how do the given values of some subset of parameters affect the values of other parameters?*

⇒ *what values of the parameters satisfy a given set of constraints?*

⇒ *what combinations of the parameters' values are optimal under certain criteria and a given set of constraints?*

Any mathematical Model that is not a formalization of some real entity may only be interesting within mathematics itself, since it is not a tool for solution of practical problems.

2.3. Conversely, the notion of an Algorithm does not imply existence of some original; the sole exception is the description of a deterministic process or procedure. The question "Algorithm of what?" is meaningful only with respect to a function; moreover, the relation between the two concepts is rather non-trivial. Indeed,

A. An Algorithm may implement a particular function that maps tuples of input values into tuples of output values. In this case, the function is the primary entity and the Algorithm is an explication of its computational implementation.

B. An Algorithm may be constructed as a formal object that defines a computational procedure that, in turn, defines some Function. In this case, the Algorithm is the primary entity and the function is its abstract equivalent.

When comparing function and Algorithm, the following should be noted:

(i) every Algorithm defines only one function

(ii) each function may be implemented by with a number of different (but functionally equivalent) Algorithms

(iii) there is no method to construct an Algorithm for any function or, rather, this problem has been solved only for some special classes of functions.

3. From Model to Algorithm: six paradoxes

3.1. Thus, an Algorithm has no direct relation to a real entity or phenomenon. The connection between the object of any practical problem and the computer may be represented by the following scheme:

OBJECT — MODEL — FUNCTION — ALGORITHM — COMPUTATION.

In what follows, we discuss certain complex dependencies that link the components of this scheme and represent the current view of the transition from a Model to the process that solves a concrete problem on a computer. These dependencies will be reflected in the form of six paradoxes that illustrate the process; although it seems well-known and natural, this process borders on absurdity in many cases.

3.2. Let us assume that we have developed a formal Model of some object, for example, of an electric transformer or the budget of an organization, and proved that the Model is adequate by running a series of tests. As mentioned before, the Model is a pair of sets, a set of parameters of various types and a set of formal relations between these parameters. We have created the Model because we need to solve a number of problems related to the object, each problem based on a set of parameter values that satisfy a given collection of constraints, specific for the problem.

We recall that every set of k parameters defines a k-dimensional space which is the Cartesian product of the parameters' domains. Each point of the space corresponds to a k-tuple of the parameters' values. Each Model defines a body in the space of its parameters' values (let us call it the *solution body*), whose points satisfy all the relations of the Model.

This body may be a single point if the Model is definite. It may be a set of points, a hypersurface, a system of bodies, etc., if the Model is subdefinite. At last, it may be empty if the Model is inconsistent. If the body is not empty, its projections onto each parameter are the parameters' domains within the Model.

The term *constraint* is synonymous to *relation* in our context. Thus, adding constraints to a Model A produces a Model A' which obviously defines a body imbedded in the body of the original Model A. Moreover, the parameters' domains under constraints A' are generally narrower (or, at least, not wider) than in the Model A.

Suppose that our constraints are explicit restrictions on the domains of the parameters. That means that they cut the space (and the body) with a corresponding set of $(k-1)$-dimensional hyperplanes. This case is typical for many real applications: the constraints may be technical requirements for a transformer or financial restrictions for an investment project.

Thus, defining the initial conditions for a specific problem in the form of new constraints added to the original Model determines new domains for the parameters of the problem, acceptable under the new constraints.

3.3. In the traditional technology, commonly used today, solution of a problem for an object described by a formal Model should begin with a necessary "ritual" step: the problem, i.e., a Model with the original constraints, is pre-

sented to an expert on computational Methods. The relationship between the parties here is rather peculiar and can be described by the first paradox.

Paradox 1. *The more real and natural our Model is, the less natural it is for the expert.*

Indeed, the task of the expert is to find a matching computational Method, and almost all of the Methods can deal only with homogeneous problems, in which similar relations link parameters of one type. At the same time, any Model of a real object includes parameters of different types and a combination of various relations (linear and non-linear equations, inequalities, logic expressions, even tabular relations and nomograms). Such a system looks to the expert as a collage consisting of a chaotic conglomeration of formal components of various classes.

The expert's reaction is obvious: he will inform us that our masterpiece of abstract art cannot be used "as is"; for serious work it should be replaced with some appropriate normal (i.e. more regular) approximation and, as a rule, not with a Model but with an Algorithm.

3.4. Thus, the transition from the Model and our problem to an Algorithm is the next necessary "ritual" step, whose peculiarity is reflected in the second paradox.

Paradox 2. *In order to advance in the computational solution of your problem, you should divide your parameters into input and output ones.*

In many cases this subdivision is quite unnatural and nontrivial, since in most real problems the solution body is symmetric with respect to all or, at least, to most of the parameters. Nevertheless, you should divide them in order for the expert to reconstruct the function that maps the input domain into the output domain. A precise reconstruction is usually impossible, and so we have to use an approximation, i.e. a *similar* function for which a computational Method is known.

Here we come to the third paradox, which mirrors the close relationship between the technology discussed here and the good old "scientific" tradition: to look for a lost key under a street lamp not because it was lost there, but because one can see better.

Paradox 3. *While trying to solve a problem defined in the context of a Model of a real object, we have to replace it with some different problem, which has a Method for its solution, but no definite connection with the original problem.*

3.5. When the expert selects a method to solve your problem, he will advise you on what coefficient it is necessary to multiply the calculation result with to make it more reliable as a solution for *your* problem. Now we have the fourth paradox.

Paradox 4. *While the result is calculated with maximal possible precision (many orders), it is then multiplied by a rough coefficient to make the solution more reliable.*

However, this is not the last paradox. Recall that as a rule we do not know the values of the input parameters. We know only the constraints on the values of some of the parameters and sometimes their types: some of them are now input parameters and others are output parameters. This means that the solution of our problem with the help of the selected Algorithm requires exhaustive testing of all input values to find those that (i) satisfy the constraints of the problem and (ii) produce the output values that also satisfy these constraints. This leads to the next paradox.

Paradox 5. *Even if we have the Algorithm we cannot always apply it to the problem directly.*

3.6. The sequence of "ritual" steps discussed above may be generalized into the following concluding paradox:

Paradox 6. *Solution of different problems for the same Model employs generally different approximating functions, hence different solution Methods, different Algorithms and different correction coefficients.*

Of course, this connection to the Model or the object is not quite clear for any problem. Neither is the relationship with the other Methods and Algorithms that are chosen for other problems arising for the same Model and object. Thus, the traditional, Algorithm-based technology of computation turns out to be utterly inadequate and helpless when it is applied to a Model rather than to a well-defined and computable function or procedure.

4. New horizons

4.1. However, providing tools for construction of formal Models is the primary and most important task of mathematics, which makes it the foundation for all applied sciences.

Let us try to come out of the "strange world" of the contemporary information technologies and imagine an ideal picture. The computer can interpret Models directly. On receiving a formal Model, the machine automatically contracts its space to a k-dimensional parallelepiped that contains

the whole solution body. When the user adds more constraints or modifies the Model, the parallelepiped is generally shrunk further or changes its dimensions in accordance with the parameters' domains. If it contracts to an empty set, this means that the Model with the additional constraints is inconsistent.

Such a technology could work with subdefinite models as well as with subdefinite values of parameters. It would not require initial approximations. To find an optimal solution, you should only set the optimized parameter to its upper or lower limit. The relations of the Model could also be subdefinite, for example, include subdefinite coefficients or indices in equations and/or inequalities.

This technology could be based on a special, universal engine that would work with the Model as a whole, rather than with a composition of many components, which are autonomous and connected only via common variables used in calculations. Not being based on the imperative style and the algorithmic mentality, this new computational mechanism would be inherently parallel, decentralized, asynchronous, and hence would allow a natural implementation on parallel computer architectures.

4.2. To an expert in traditional computation techniques, it is obvious that such a technology is impossible in principle. However, such a technology does exist. It provides the user with almost all of the new possibilities described above: he can interact with the Model directly, without needing intermediaries to apply Methods, Algorithms and Programming. Moreover, it allows using different formal apparatuses within one Model, such as computational algebra, logic, set theory, etc., as well as with their combinations.

These new capabilities may be illustrated with a small Model containing two equations, one inequality and two logic expressions (the "$\longrightarrow$" designates the implication operator, k is integer, x and y are real):

$x^3 + 10 * x = y^x - 2^k$;
$k * x + 7.7 * y = 2.4$;
$(k-1)^{y+1} < 10$;
$\ln(y + 2.0 * x + 12.0) < k + 5$ or $y > k^2 \longrightarrow x < 0.0$ and $y < 1.0$;
$x < 0.0 \longrightarrow k > 3$;

The system without the last expression has several solutions.

$$k = [1, 4]$$
$$x = [-1.2850, -0.08932]$$
$$y = [0.32329, 0.97925]$$

Adding the last expression provides a unique solution that meets this restriction:

$$k = 4$$
$$x = [-1.285058, -1.2850575]$$
$$y = [0.9792507, 0.9792510]$$

Obviously, there is no general Method within the traditional paradigm to deal with problems similar to this simple example.

4.3. This approach is twenty years old now. At the very beginning of the 80s our group came to a qualitatively new approach that was named subdefinite mathematics [8, 9]. This philosophy very soon developed into a theory [10, 11] extending various classic formal apparatuses (algebra, logic, set theory, graphs, etc.) with the ability to apply the same operations and relations to variables with subdefinite values. The approach has been implemented in a number of software systems and technologies [12, 13, 14, 15, 16, 17, 18].

In the second half of 80s, a similar approach called constraint programming (or constraint satisfaction, constraint propagation, etc.) began to blossom into a promising and popular research field at the intersection of AI and interval mathematics. Independently of our work that was not known outside USSR at the time, several new lines of research combined constraints with subdefinite (interval) numbers [2, 3]. As a result, several new software systems and technologies were created which expanded the Model-oriented paradigm to wider applications [5, 6].

5. Conclusion: end of the Algorithm Era

5.1. The comparison of Model and Algorithm, the criticism of the algorithmic approach and a brief description of the new paradigm were presented above on the material of algebraic computation problems. To present a more general view within the framework of the whole software technology, it is necessary to mention briefly the following three directions:

A. *Differential equations.* I have no information about application of the new paradigm in this domain, except a few interesting experiments of small scale.

B. *Software technology.* Although from the very beginning the Algorithm was the foundation of the programming techniques for computers of von Neumann architecture, many attempts to develop alternate approaches have been made. First of all this was related with research on parallel programming for multiprocessor computers: for example, the concept of asynchronous programs [4] and many other efforts in the late 60s and early 70s (see survey [7]).

However, qualitative progress here was ensured by the apparatus of subdefinite models and the most recent research in constraint programming, because they are based on a decentralized, multi-agent data-driven computational process, which allows implementing any software system in the form of a structured Model that integrates an hierarchical complex of autonomous components.

Another important step in the radical transformation of the entire information technology is the development of the object-oriented philosophy, which started the evolution of the traditional program into a Model. Presently, however, this approach is only forming the basis of the future technology by bringing an object-oriented program closer to a structured Model although leaving unchanged the original algorithmic control of its realization. One of the research themes in our group is the project TAO (Technology of Active Objects) [17], which is aimed at integration of the object-oriented approach with multi-agent constraint programming and subdefinite models.

C. *Artificial Intelligence.* The focus of this discipline is in the formal means for knowledge representation and processing, which is very close to the central purpose of the mathematics itself. Their methodological positions differ, however, in the following very important points:

- ◇ While concentrating on creation and study of formal apparatuses, mathematics does not pay much attention to their application to problems in other disciplines.
- ◇ Artificial Intelligence has the opposite orientation: from studying various forms of knowledge to developing a complex of formal means that attempts to cover the whole spectrum of activities.

Thus, from the very beginning of Artificial Intelligence the Model and direct interaction to it had been the principal goal. And a natural result of that orientation was the need to break out of the Algorithm paradigm in all possible directions: Lisp, Prolog, frames, production rules, multi-agent systems and, finally, constraint programming.

It seems that the new paradigm should be formed through integration of mutually complimentary constituents of all of the above directions of the information technology development.

5.2. The number of researchers working on the new paradigm is insignificant compared to the immense professional manpower involved in further development of the traditional paradigm (in particular, in the computational mathematics) over many decades. In spite of that, however, the results obtained by now are obvious enough: in many applications the novel paradigm

is capable to demonstrate its advantages even now. Its fast progress and wide adoption as a general information technology are making its superiority over Algorithms more and more obvious, and so we take seriously the forecast that in the nearest 10–15 years Algorithm will repeat the fate of programming in machine codes and assembly languages: it will keep its place only in a relatively thin, lowest level in the software technology of the next generation.

References

[1] Borning A., The programming language aspects of ThikLab, a constraint-oriented simulation laboratory. ACM Trans.Progr.Lang.Syst., 1981, 3(4).

[2] Davis E., Constraint propagation with interval labels. Artificial Intelligence, 1987, V32.

[3] Hyvonen E., Constrain reasoning based on interval arithmetic: the tolerance propagation approach. Artificial Intelligence, 1992, V.58.

[4] Kotov V. E., Narin'yani A. S., Asynchronous computing processes over memory. Kibernetika, 1966, N 3 (in Russian)

[5] Kumar Vipin, Algorithms for constraint-satisfaction problems: a survey. AI Magazine.-1992, Spring.

[6] Mayoh B., Tyugu E., Uustalu T., Constraint Satisfaction and Constraint Programming: A Brief Lead-In. Constraint Programming. Berlin, Springer-Verlag, 1994.

[7] Narin'yani A. S., Looking for an approach to a theory of models of parallel computation. — In: International Symposium on Theoretical Programming, Springer Verlag, Berlin, 1974.

[8] Narin'yani, A. S., Subdefinite sets — new data type for knowledge representation. Preprint N 232 of Comp.Center of Siberian Div. of USSR Acad.Sci., 1980 (in Russian).

[9] Narin'yani, A. S., Sub-definite models and operations over subdefinite values. Preprint No. 400, Computer Center of the Siberian Div. of the USSR Acad. Sci., Novosibirsk, 1982 (in Russian).

[10] Narin'yani, A. S., Sub-definiteness, Over-definiteness and Absurdity in Knowledge Bases (some formal aspects). Computers and Artificial Intelligence. — Bratislawa, 1986.

[11] Narin'yani, A. S., Subdefiniteness in knowledge representation and processing. Technical Cybernetics, Moscow, No. 5, 1986 (in Russian).

[12] Narin'yani, A. S., Intelligent software technology for the new decade. Communications of the ACM, v. 34, No. 6, 1991.

[13] Narin'yani A. S., Borde S. B., Ivanov D. A., Subdefinite mathematics and novel scheduling technology. Artificial Intelligence in Engineering, v.11, N1, February 1997 (Published in 1996)

[14] Shvetsov I., Kornienko V., Preis S., Interval spreadsheet for problems of financial planning. Proc. of the 3rd Intern. Conf. on the Practical Application of Constraint Technology PACT'97, England, London, April 1997.

[15] Shvetsov I., Nesterenko T., Starovit S., Technology of Active Objects. Proc. of AAAI Workshop on Constraints and Agents, Providence, USA, July 1997.

[16] Shvetsov I. E., Telerman V. V., Ushakov D. M., NeMo+: Object-Oriented Constraint Programming Environment Based on Subdefinite Models. 3rd Intern. Conf. on Principles and Practice of Constraint Programming CP'97, Linz, Austria, October 1997. — Lecture Notes in Computer Science, No. 1330.

[17] Sussman G. J., Steele G. L. Jr., CONSTRAINTS — a language for expressing almost-hierarchical descriptions. Artificial Intelligence, v. 14, 1980.

[18] Tyugu E. H., Conceptual Programming. Moscow, Nauka, 1984 (in Russian).