Joint NCC & IIS Bull., Comp. Science, 16 (2001), 129–139

Representation of interval models and interval data in the constraint programming system

A. A. Lipatov, A. V. Poltaratskii

Introduction

One of the most advanced and practically significant approaches in constraint programming [1] is the method of subdefinite models [2, 3].

This method is the basis for a multilevel constraint programming technology that has been developed to solve various problems in economics, engineering, calendar planning, etc.

A subdefinite model is a pair (X, C), where X is a set of variables and C is a set of constraints on them. In the subdefinite model, the values of variables can be both precise and subdefinite. For variables of numerical data types subdefinite values can be defined as intervals or sets of possible values of variables.

We call a subdefinite model, where subdefinite values of variables may be only numeric intervals, an interval model.

This paper deals with some problems related to the representation of interval models and interval data in the UniCalc solver [4] intended to solve the systems of algebraic equations and ineualities. A number of methods developed by the authors for the solution of these problems in the UniCalc environment are considered.

1. Some issues of data representation in the UniCalc solver

The UniCalc solver supports the interactive mode of problem solving. If a system has several solutions, then each of them can be found by entering additional constraints on the values of the variables. These constraints can be in the form of equations, inequalities or logical expressions.

Based on the analysis of calculation results, we can change the source model, for example, by modifying the original constraints, adding new constraints or refining the values of variables.

Effectiveness of the UniCalc solver can be increased by adding the following new capabilities to the UniCalc shell.

1. Displaying and editing the interval model using a traditional mathematical notation. In the current implementation of the solver shell, the model is created and presented in the UniCalc language.
2. A high-level model description language that facilitates creation and modification of large complex mathematical models. Such a language is not available in the current implementation of the solver.
3. Visualization of the interval data used in models. The current version of the UniCalc shell has a tool for visualization of user-defined functions. It allows one to display one or several functions of one argument (possibly with subdefinite values).

The problem of visualization of interval data arises in connection with the development of various methods of interval calculations, as well as within the framework of subdefinite models.

It is necessary to note that in UniCalc the function of graphical tools is not only to allow a visual presentation of the interval data produced by calculation, but also to facilitate direct control over the calculation process.

One of the most important problems of interval data visualization is displaying arrays of interval values and subdefinite functions.

The following characteristics of arrays of interval values are relevant for their visualization.

1. The values of the elements of arrays are subdefinite.
2. The amount of elements in array is large.
3. The range of values of the elements may be very large, covering several orders of magnitude.
4. We need to support interactive manipulation with the graphical representation of interval data.
5. It is necessary to compare several arrays or a sequence of states of one array obtained after execution of several calculation steps.

The following problems arise in connection with visualization of subdefinite functions.

1. The values of functions and their arguments are subdefinite.
2. The number of arguments of a function is arbitrary.

3. It is necessary to afford an opportunity to preset the invariable values for some arguments of function. These values may be either exact or subdefinite. Such arguments will be called "fixed."
4. Interactive manipulation with the graphical representation of a function should be supported, when the user is capable to add and change additional constraints on the values of the function for some or all values of its arguments.
5. It is necessary to compare the results produced by different calculation steps for the model.

2. Solutions for some problems

The authors are developing a new version of the integrated shell for the UniCalc solver. An outline of the solution for some of the problems listed in section 1 is presented. The new shell of the solver includes advanced tools for symbolic representation of interval models, as well as a subsystem for graphical representation of interval data (called iG). In particular, iG implements visualization of one-dimensional and two-dimensional arrays of interval values. New methods of visualization of subdefinite functions of one argument have been also developed.

2.1. Enhancing the UniCalc language

The main reason for extending the language is that in its current implementation it is difficult to develop complex models with many variables and expressions; in addition, reusing models or their parts in other problems is impossible. The key technologies used in the new language are generalization, inheritance, aggregation and polymorphism of models.

Generalization and inheritance of models

Generalization and inheritance make possible to use local similarity of different parts of models, as well as multilevel classification of models. For implementation of complex models, it allows moving from simple to complex:

– first, we describe the base model with a minimal set of expressions;
– next, we describe a model that inherits all the expressions of the base model and then add new expressions.

Aggregation of models

Aggregation is establishing dependencies between a compound model and the submodels representing its components (the "whole"–"part" relation).

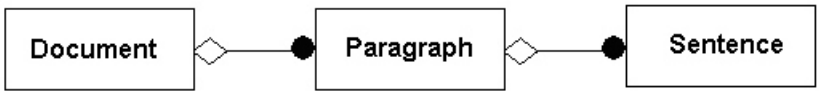

Figure 1. The document consists of several paragraphs

The most important property of the relation of aggregation is its transitivity (if A is part of B, and B is part of C, then A is part of C): thus, from the above figure it is possible to conclude that the document consists of several (zero or more) sentences.

It is easy to see that the relation of aggregation is not symmetric (if A is part of B, then B is not part of A).

Polymorphism of models

In the current version of the UniCalc shell, we are implementing two forms of polymorphism.

1. At the level of inheritance of models, there are two base models M_i and M_j that are different descriptions of the same object. There is also a model M whose parent can be either of the base models, depending on the context. Thus, in compliance with more and more definite conditions, M may inherit either from M_i or from M_j.
2. At the level of modifying structure of the task, two variants of the base model M_0 are defined in blocks B_i and B_j, correspondingly. The model M_1 is derived from M_0 by adding to M_0 some constraints which are kept in the block B. In the structure of the task we describe the condition that decides which blocks would be used to implement M_1: B_i and B_1, or B_j and B_1.

2.2. Visualization of a UniCalc model

The following problems relate to the display of complex mathematical models:

– difficulty in writing and perceiving complex mathematical expressions;

– large number of relations within unstructured models.

The editor of the new environment satisfies modern general requirements to the user interface. It provides the following features:

– highlighting comments and standard functions;

– construction of the model navigation tree;

– calculation stack;

– change of the text font size.

An intuitively clear interface is provided for model development. The toolbar provides buttons that help to write formulas of arbitrary complexity, such as: multi-level fractions, square roots, exponential expressions etc. Besides, the developer can write the formulas in a usual linear manner, and they can be automatically transformed to a multi-level (multi-line) form. The editor provides automatic single-line to multi-line conversion, i.e. a model developed in an older version of the environment can be viewed in the new, multi-line notation.

Development of models often implies writing documentation. The support of the Windows object embedding mechanism allows one to insert expressions and formulas built by the shell into documents created by other programs.

2.3. Visualization of one-dimensional arrays of interval data

The basic image is a one-dimensional array placed on the screen horizontally (one element on a line) over an optimal coordinate grid. Each line of the image corresponds to an element of the array. The interval value of an element is represented by a coloured band and, possibly, in the text form.

The purpose of the iG subsystem is to support visualization of arrays with the number of elements from several units to several hundreds. For this purpose, it has a system of scales that allows one to represent arrays of different size with a different level of detail. The system includes the following scales.

Basic scale: each element is represented by a line containing information about its value and graphical representation of the inerval, see Figure 2.

Minimal interactive scale: there is no text information in the line and the size of the graphical representation is the minimum necessary to support the interactive mode, see Figure 3.

Minimal graphical scale: one line of the minimal visible width is provided for each element of the array. In this case, the interactive mode is impossible.

For any scale, information about the value of any element of the array may be displayed in the status bar.

It is possible to change the scale during a session.

For visualization of arrays with wide spread in values of their elements over the intervals and their position in the numerical axis, iG implements *linear* and *logarithm* grids. It is also possible to choose the scale for the element value axis.

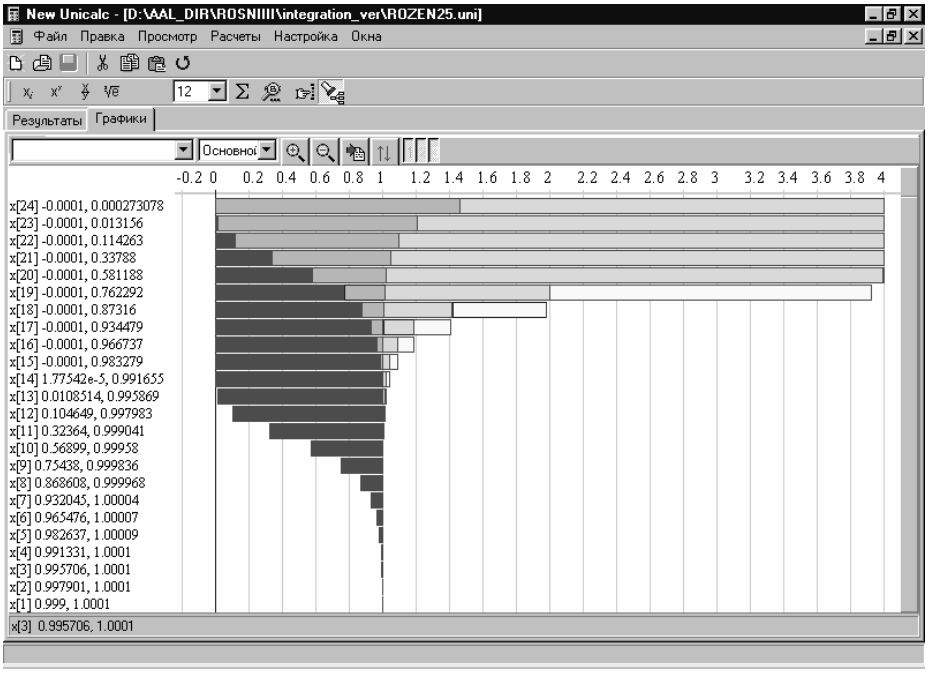

Figure 2. The iG system is used to display the array X containing 24 elements. The current values are between -0,00001 and 1. The values of elements for the previous states are between -0.00001 and 1E19. The linear scale is used

The interval may be chosen by explicit or implicit marking of its bounds. In the latter case, the range is set so as to display the value of the present element at full width of the screen. The sequence of displayed intervals is stored in a special stack, and so the user can return to any previously displayed interval.

The *logarithmic* grid may be used in the case when the range of values of the array elements will not allow comparing some of their values using the *linear* grid. Thus, the values of an array are displayed against a special coordinate grid. This grid reduces the spread in values of the elements: emphasizes small differences and reduces great ones.

The values of elements can be edited by entering new numerical values or by direct manipulating the graphical representation. In the former case, the editing is more precise, while in the latter it is more intuitive. The values entered by the user are added to the model as additional constraints. The UniCalc solver takes into account these constraints at the following step of calculations.

The iG subsystem provides the capability to compare the results obtained

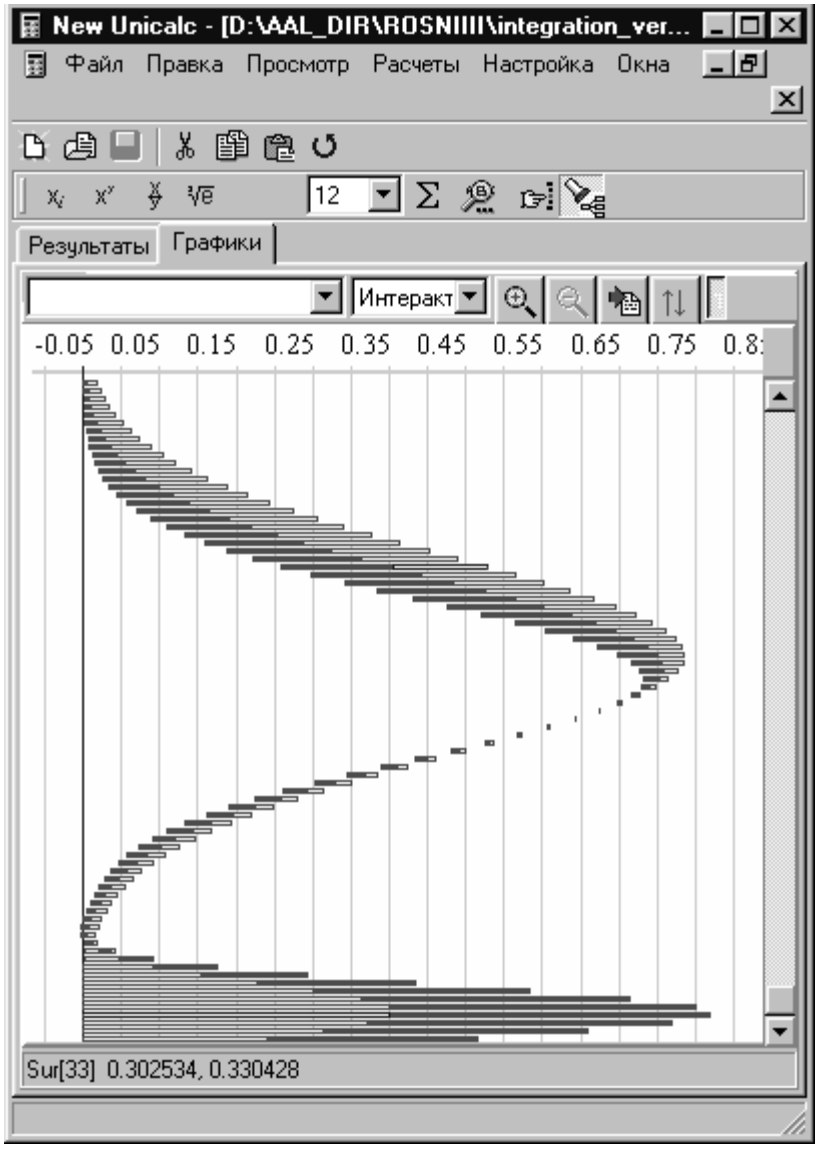

Figure 3. Array Sur with 91 element. The array is shown in the minimal interactive scale. The values from the last and the two preceding calculation steps are displayed

at different calculation steps of the model. The results of each calculation step of this model are stored in a stack. At any time, the results of several (upto six) calculation steps can be displayed simultaneously, see Figure 2, Figure 3.

The values obtained in different steps are displayed by shades of the chosen colour, with earlier results shown in paler shades. In addition to the elements of the array, the system displays the constraints added by the user with the help of the iG subsystem, see Figure 2.

2.4. Visualization of two-dimensional arrays of interval data

Visualisation of two-dimensional arrays of interval data is based on a generalized representation that allows us to estimate the values of the elements and their distribution in the array, as well as to choose cross-sections that will be displayed in greater detail.

A two-dimensional array is represented by a matrix. Each element of the matrix corresponds to an element of the array. The shade of colour of an element of the matrix reflects the position of the corresponding interval value on the numerical axis (i.e. the average value of the interval), see Figure 5, or the degree of subdefiniteness of its value (i.e. the width of an interval), see Figure 4. The user selects the characteristic to be displayed. The correspondence between the shades and the values of the characteristic is determined by a colour scale. Positive values correspond to shades of red colour, and negative values correspond to shades of dark blue colour. In addition, the user can get information about the value of an element of the array and its indices in the text form.

The display allows adjustable scales for rows and columns. The user can view arrays of various sizes and their fragments with a different level of detail.

It is possible to choose a numerical interval that will be displayed using the larger part of the colour scale. This allows us to show arrays with wide spread in the values of elements, both in the width of intervals and their position on the numerical axis. The distribution of colour shades on the numerical interval containing the values of elements can be *linear* or *logarithmic* chosen by the user.

A two-dimensional array can be displayed in a more evident manner by choosing vertical (column) or horizontal (row) sections and displaying them in a window for one-dimensional arrays. Even in this mode, it is still possible to view the two-dimensional display at the same time. The display of the array section supports all of the capabilities described above for one-dimensional arrays.

In order to support working with multiple sections, it is proposed to

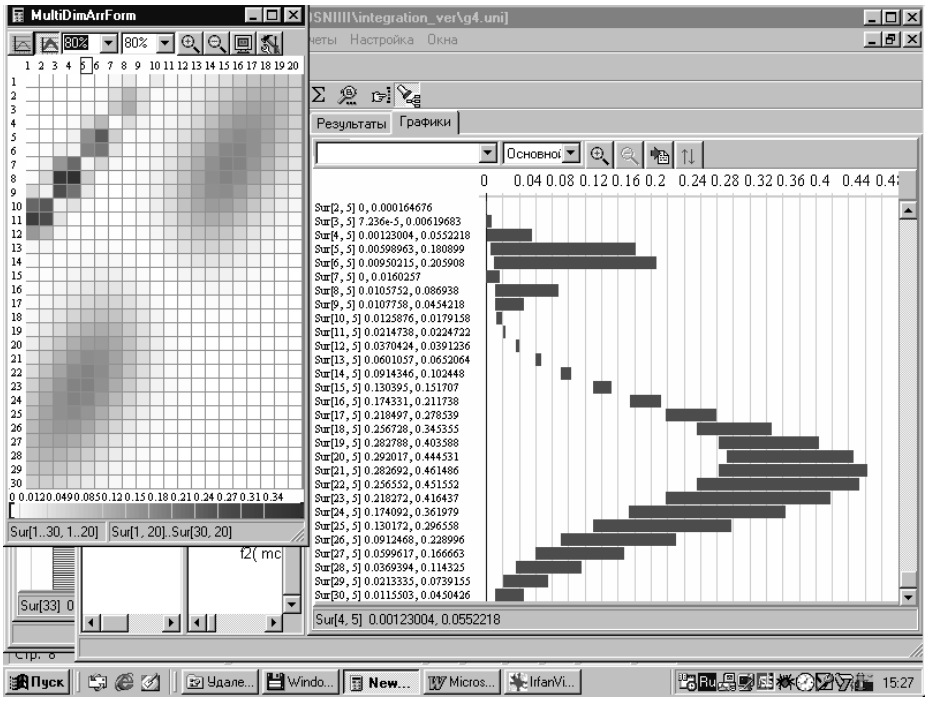

Figure 4. Array Sur (30×20 elements) is shown in the mode of displaying subdefiniteness of its elements. The values of subdefiniteness lying in the ranges from 0 to 0.37 is displayed by shades of red colour. The window displaying one-dimensional arrays shows the 5-th column of this array

develop the means for navigating through the previously displayed sections. The sequence of displayed sections should be stored in a stack and presented to the user as a list in which he (she) can choose the necessary section.

Interactive manipulation with a two-dimensional array is supported in the one-dimensional section display windows as described above in 2.3.

Comparison of the states of the array at several computation steps will be implemented in a manner similar to comparison of one-dimensional arrays.

2.5. Visualization of subdefinite functions

The visual representation of a function is based on the one-dimensional array of its values. The values of the function, which are stored in the array, correspond to the values of the variable argument that are uniformly distributed on the interval defined by the user. The number of values of the varying argument and values of the fixed argument are set by the user.

Graphically a function is represented by a two-dimensional diagram. The

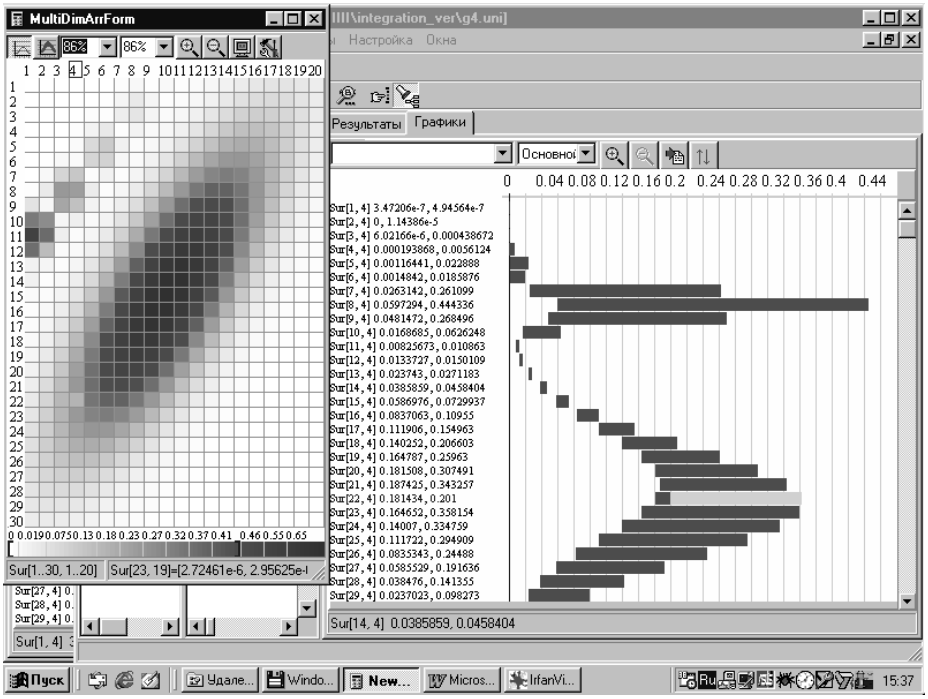

Figure 5. The array Sur (30×20 elements) is shown in a mode of display of the position of values of its elements on a numerical axis. The values of this characteristic lying in the ranges from 0 to 0.8 is displayed by shades of red colour. In a window of display of one-dimensional arrays, the 4-th column of this array is shown. The additional consraint Sur $[22, 4] < 0.201$ is imposed on one of its elements

X-axis in the diagram shows the values of the variable argument, and the Y-axis represents the values of the function. The function is represented by two curves for the lower and upper bounds of its subdefinite values. The value of a function corresponding to the value of its argument not reflected in the array is determined by interpolation.

The user has a possibility to enter additional constraints on the function values by manipulation with its graphical representation. These new constraints can be used at the next step of the model calculation. In order to compare the results of several calculation steps, several function diagrams for these calculation steps can be simultaneously displayed over the same coordinate grid.

Conclusion

In this paper, we discussed some problems relevant to representation of data in constraint programming systems, in particular, in the UniCalc solver. The following problems have been considered:

- development of means that facilitate creation and editing of interval mathematical models;
- graphical representation of interval data.

The paper describes the methods for solution of some of these problems developed by the authors for application in the UniCalc solver. The following methods were considered:

- editing of interval models in the text form;
- graphical display of one-dimensional and two-dimensional arrays of interval values;
- graphical display of subdefinite functions.

The application of the proposed methods of representation of the interval data makes the user interaction with the solver more efficient. The graphical representation of the interval data allows one to perceive the results of calculations and to control the calculation process in a more intuitive manner.

The methods can be used in various packages of interval calculations for solving the abovementioned problems.

References

[1] Montanari U., *Networks of Constraints: Fundamental Properties and Applications to Picture Processing*, *Inform. Sci.* — Vol.7, 1974. — P. 95–132.

[2] Narin'yani A. S., *Model vs. Algorithm: Change of Paradigm in Information Technology*, In present one shot.

[3] Telerman V. V., Ushakov D. M., *Subdefinite Models as a Variety of Constraint Programming*, *Proc. of the Intern. Conf. Tools of Artificial Intelligence (ICTAI'96),* Toulouse, 1996.

[4] Narin'yani A. S., Semenov A. L., Babichev A. B., Kashevarova T. P. and Leshchenko A. S., *A New Approach to Solving Algebraic Systems by Means of Sub-Definite Models*, In: Proc. of the 16-th IFIP Conference on System Modelling and Optimization. Compiegne, France. July, 1993. J.Henry and J.-P. Yvon (Eds.), Lecture Notes in Control and Information Sciences. — Vol. 197, Springer Verlag, 1994.