Computational Environment Generation for Computer Algebra Systems

Svetlana Cojocaru

Institute of Mathematics and Computer Science, Academy of Sciences of Moldova

Str. Academiei, 5, Chişinău, MD 2028, Republic of Moldova

Abstract: This paper deals with the problem of user interface development for Computer Algebra Systems. An algorithm of computational environment generation based on inferences from users's actions is proposed.

Keywords: intelligent interface, computer algebra systems, adaptation

Svetlana Cojocaru – Dr. h., is the deputy director of the Institute of Mathematics and Computer Science, Academy of Science of Moldova. S. Cojocaru has published 3 books, 5 chapters in books, more than 100 papers. Her research interests include natural language processing, computer algebra, biological computing.

1. Introduction

The software complexity is permanently increasing and it requires from the user more vast and deep knowledge about systems functioning. Interfaces, which realize the human-computer interaction, take upon themselves to itself also a part of user's tasks. But in the process of interface developing one can observe the same tendency as in general software evolution – they also become more and more complicated and insufficiently flexible. A solution of this problem might be intelligent interfaces, which are able to adapt themselves on user's necessities, can learn, can take over the initiative in their interaction with user, are able to guide him in order to facilitate his objectives achieving in a fast and comfortable way.

Interface elaboration for Computer Algebra Systems (CAS) was and remains a topic to be investigated. There are a number of overviews treating this subject, among them we can mention (Kajler 1998, Kajler, Soiffer, 1998, Grabmeier, Kaltofen, Weispfenning, 2003). There is also a large list of concrete CAS descriptions, the corresponding interface (in a rudimentary or developed form) being one of the presented components.

Analysing the CAS interfaces one can conclude that they respect the requests specific for intelligent interfaces (Ross, 2000, Filip, 2007, Waern, 1997) in a reduced measure only:

- The user is guided in the process of his problem solution,
- It is not necessary to study preliminarily how the system works, often it is possible to learn the functioning mode directly in the process of interaction with the system,
- The system makes some errors prevention (partially) by checking the problem's formulation correctness and verification of initial data.

Less significant is the attention paid to user modelling. Because CAS are from the very beginning oriented on a pre-established users class, their model (in relation to the class of problems, data type, non-ambiguous interpreting) is beforehand included into the system.

Therefore in the case of CAS it would be more relevant to speak about user's personalisation. We should mention that the problem of personalisation, being realised, is not sufficiently examined. User can apply some possibilities of individualisation, but he should adapt the system to his needs or preferences by himself.

We will note also that the possibility to interact in natural language in the frame of CAS is rather limited.

Thus, in order to increase the intelligence level of CAS interfaces one can fix the necessity to implement some additional features, including user's personalisation, adaptation to his individual requests and preferences. One should note, that the user not always is able to formulate his preferences.

In the next sections the process of adaptation in CAS interfaces will be examined. We will respect the following principles:

- The goal of adaptive systems is to facilitate user's objectives achieving in the fastest and easiest way, assuring, as much as possible, a high degree of satisfaction (Ross, 2000).
- It is important that system shall not irritate the user, not reduce by its interventions the speed of his work, it shall not initiate unbidden operations in some inappropriate moments, creating in such a way obstacles to achieve the goal immediately.

2. Adaptation to the User

The more widely is used a software product the more difficult is to elaborate a comfortable environment, which will be able to assure a high performance for every user. A series of studies performed since 1980 had been finished in conviction that there is no possibility to specify all varieties of users, programs and equipment in order to work out interfaces ideally convenient for each case. The only resource comparable with the number of users are the users themselves, therefore namely they should be involved in the process of the personalised interfaces developing (Weld, Anderson, Domingos, 2003).

Analysis of many CAS interfaces convinced us that even in the case of the most developed ones users have some wishes, which demand an adaptation to their interests and individual habits (Cojocaru, Malahova, Colesnicov, 2006). As it was mentioned above, adaptation to user is one of the distinctive, indispensable features of intelligent interfaces. We will describe the process of its implementation in general case and the specific for CAS in particular.

Let us recall that the adaptation process might be realised in the following three modes:

- Adaptation is performed in advance, by a design oriented to a specific type of users.
- The system permits a dynamic adaptation, which is performed by user in the process of his interaction with the system.
- The adaptation is made by system in the process of its interaction with the user.

In the last case the adaptation is made through a dialogue initiated by the system or through "spying" user's actions and making adjustments to his preferences basing on inferences from the collected knowledge.

In some sense we can consider that all CAS are from the very beginning oriented on a fixed class of problems and a determined group of users, thus the last two methods of adaptation will be the subject of our interest. The case, when the system permits a dynamic adaptation realised according to user's request (and with his contribution) is frequently implemented in modern software and is called "customization". In (Weld, Anderson, Domingos, 2003) the following ways of implementing of that adaptation method are underlined:

- Visible menu selection,
- Adding buttons in menu bar,
- Definition of macros,
- Using specialised programming languages to write own scripts.

Of course, only a limited number of users poses the necessary abilities and have sufficient time (and patience) to apply this methods (especially, the last two), therefore, keeping in mind all these possibilities, we will deal with dynamic adaptation initiated by the system. Our approach will be explained basing on the example of Computer Algebra system Bergman (www.servus.math.su.se/bergman).

3. Interface of Computer Algebra System Bergman

Bergman is a computer algebra system calculating non-commutative and commutative Gröbner basis, Hilbert series, Anick's resolution, and Betti numbers for ideals over rings and for modules. Bergman allows a multitude of choices for various kinds of input, operation, output, etc. One may let predefined "top of the top'" procedures make the choices for him, or specify details by means of a large number of lower level mode setting procedures.

To perform some computations in Bergman you should set up the polynomial ring and its surrounding. This action includes selection of the algebra type (commutative or non-commutative), the variables, the ordering, the characteristic, type of coefficients, and variable weights. One can also select the strategy of computation, and input-output mode. There are some minor mode selections too. For a complete description see (Backelin, Cojocaru, Ufnarovski, 2005).

Note, that different kinds of calculations demand different parameters settings, they should be compatible with each other and compatible with the corresponding calculation procedure. Because Bergman has a high degree of flexibility in modes selection, sometimes it might be difficult for user to make a correct and efficient selection. These problems are avoided by developing of graphical user interface, which permits to operate with Bergman in a manner close to the usual mathematical surroundings.

The main principles respected in the process of interface design correspond both to common requests and to those specific for intelligent interfaces, namely:

- the interface reflects specific mathematical object and notions,
- it is possible to use the system without preliminary studying of its description,
- some tasks are taken over from the user and executed by the system,
- the interface simplifies access to different Bergman's functions,
- the interface operates the data control before calculations,
- input and output can be performed in bi-dimensional format,
- it is possible to save the environment in which the computations were performed and to repeat the calculations in the same conditions, with the same or modified data.

Moreover, the interface can serve also as a prospectus for Bergman giving the possibility to see on the graphical panel the most important facilities without reading a manual or applying the corresponding help function.

The shell consists of the main menu at its top, then the toolbar, and then three panels (top-left, bottom-left, and the right one). The main menu consists of items that are used to organize calculations, and the top-left panel contains drop-down menus and other elements that specify mathematical parameters of the problem to be solved. The bottom-left and right panels display information, or are used to enter some data (e.g., variables and relations are input from the right panel). Main menu items are those ordinary (*File, Tools, Options, Windows, Run* etc.), but there are two specific ones named *Sessions* and *Environment.* These items are less obvious and we shall explain them.

Our observations on CAS using permit to conclude that most of the mathematicians deal for a more or less long period with research of a certain class of problems, which need calculations in a specific mathematical environment, determined by a series of parameters. These parameters remain mostly unchanged during several computation sessions, only a few of them may be modified (for example, in the case of Bergman many calculations are performed only by variations related to input polynomials or modules). We introduced the notion of Environment just to establish this repeatable surrounding. It fixes the preferable setup immediately after starting Bergman. It is possible to have a number of environments for each user, depending on the class of problems one is interested to solve. Moreover, it is possible to make some changes in the environment, permanent or valid during the current session. The same environment might be used by different users if they solve the problems belonging to the same class.

A possibility with a higher degree of determinism is that realized by the mechanism of sessions. A session is a set of parameters that fully defines the problem to be solved. Session is implemented as a directory where all data are saved, Bergman input files are generated, and Bergman output files are produced. Sessions serve to return to previous Bergman calculations, modify them, and experiment with them. Informally, sessions give to a mathematician the possibility to use the previous experience of Bergman's users (own or others) and to save the current setup for the future calculations.

When the user wants to create a new session, he will be asked which environment should be used as the base to create this session. An environment is a partial set of data common for several sessions. It corresponds to the group of mathematical problems the user investigates during different sessions. E.g., after Bergman's installation the environment directory contains a profile called ``commutative" that fixes a single parameter, the commutativity. There is also a pre-defined environment containing the default settings. For example, by default, if no other settings are made, Bergman will calculate Gröbner basis in the commutative ring with characteristics equal to zero.

4. Environment Generation

The menu items *Environmen*t and *Session* serve to implement the mechanism of adaptation. There are three methods implementing environment creation in Bergman.

The first one is based on a dialogue operated by the system. It puts a number of questions, collecting and analyzing the answers so that it becomes possible to fix necessary data permitting to generate the computational environment. Each question is followed by a list of possible answers. A fragment of such a dialogue is presented below:

```
> (dialogue)
What kinds of objects do you plan to work with?
 - One of the following: grc, ngrc, grnc, ngrnc
> grc
```

The suggested answers mean correspondingly "gradded commutative", "non-gradded commutative" etc. Obviously, this method can be used only by persons having some experience in work with the system.

The second method is based on session editing; it might be the current session or any of the saved ones. The system displays the list of parameters with corresponding values and proposes to select the wanted ones. The selected values can be edited if necessary.

The third method is based on inferences from user's actions. The series of previously saved sessions are analyzed and on their base a new environment is created. In adaptation based on inferences, as a rule, one of two possible criteria is used: that of *recent* actions or that of *frequent* actions (Gajos et al., 2006). We propose an heuristic algorithm which will take in account both criteria. It is necessary to mention that solving of this problem for CAS has its peculiarities: the parameters' values, selected from different sessions according to the frequency criterion might be contradictory.

Let $A_1,\ldots,A_n$ be a sequence of attributes associated with Computer Algebra System S. Each attribute A_i has a corresponding set of values $V_i=\{v_{i1},\ldots,v_{ik}\}$ $(i=1,\ldots,n;k=1,\ldots,r)$. These values form a multidimensional array

$$F_S=[v_{11},v_{12},\ldots,v_{1n_1}]X[v_{21},v_{22},\ldots,v_{2n_2}]X\ldots[v_{m1},v_{m2},\ldots,v_{mn_m}].$$

The element $f_{i_1i_2\ldots i_m}$ has the value 1 if there exists a sequence of values $v_{i_1},v_{i_2},\ldots,v_{i_m}$ the attributes $A_1,A_2,\ldots,A_m$ can have in system S, otherwise $f_{i_1i_2\ldots i_m}=0$. This array will be called a *compatibility function*.

Basing on these attributes and values we construct the corteges

$$C_l=\{(A_jv_{jm_j}),\ldots,(A_{j+s}v_{j+s,m_s})\},l=1,\ldots,N.$$

Each cortege will have the corresponding weight t_l.

Our aim is to construct a cortege C following both criteria of the most frequent and most recent actions adding one by one attributes A_i with compatible values. Initially we will take into account the frequency criterion. Let φ be the initial threshold of frequency.

1. An attribute A is included into cortege C if it is present at least in φ of corteges C_l.
2. The value v of this attribute is established equal to the most frequent value of the attribute A occurred in corteges C_l. If there are two or more values with equal frequencies go to step 3, if not – go to step 4.
3. Use the recent action criterion: among the values with equal (maximal) frequencies select as a value of attribute A that, which belongs to the cortege with the greatest weight.
4. Verify if values from the obtained cortege satisfy the compatibility function. If yes, then C is the result, else go to step 5.
5. From the sequence of values select that component (or those components) which cause the incompatibility. Replace them with other values having lower frequencies. Repeat step 4 if such values exist, otherwise go to step 6.
6. It was not possible to construct a cortege with those parameters. Replace them, namely the sequence of initial corteges C_l, the threshold of frequency φ. Repeat the process starting with step 1.

The proposed algorithm does not produce always a resulting cortege. In an unsuccessful case on can vary the initial parameters (the sequence of corteges C_l, the threshold of frequency φ etc.) and repeat the process until a result satisfying the compatibility function will be obtained.

5. Conclusions

N. Kajler (Kajler, 1998) considers computer-human interaction in symbolic computation "a wide and interdisciplinary research area", including different aspects, which "current user interfaces either do not handle well or do not address at all". We tried to cover only a small part of them, namely the problem of adaptation to users using the mechanism of computational environments and sessions. Our experience shows that in many cases about 5 saved sessions are enough to be able to understand user's preferences and to generate for him a convenient environment reflecting his domain of interests.

REFERENCES

1. BACKELIN, J., S.COJOCARU, V.UFNAROVSKI, **Mathematical Computations Using Bergman**, Lund University, Centre for Mathematical Science, 2005, 206 p.

2. COJOCARU, S., L. MALAHOVA, A. COLESNICOV, **Providing Modern Software Environments to Computer Algebra Systems,** Computer Algebra in Scientific Computing, Proceedings of the 9th International Workshop, CASC 2006, Chişinău, Moldova, September 11-15, 2006. LNCS, Springer, pp. 120-140.

3. COJOCARU, S., A. COLESNICOV, L. MALAHOVA, **Interfaces to Symbolic Computation Systems: Reconsidering Experience of Bergman,** Computer Science Journal of Moldova, ISSN 1561-4042, Vol. 13, No. 2, 2005, pp. 232-244.

4. FILIP, F. G., **Decision Support Systems**, Editura Tehnică, Bucureşti, 2007, 364 p. (in Romanian).

5. GAJOS, K., M. CZERWINSKI, D. TAN, D. WELD, **Exploring the Design Space for Adaptive Graphical User Interfaces**, In: Proceedings of the working conference on Advanced visual interfaces - 2006, Venezia, Italy, May 23 - 26, 2006, pp. 201 – 208.

6. GRABMEIER, J., E. KALTOFEN, V. WEISPFENING (eds.) **Computer Algebra Handbook**, Springer, 2003, 637 p.

7. N. KAJLER (ed.). **Computer-Human Interaction in Symbolic Computation**, Springer-Verlag, Wien, 1998, 212 p.

8. KAJLER, N., N. SOIFFER, **A Survey of User Interfaces for Computer Algebra Systems**, Journal of Symbolic Computation, Vol. 25, issue 2, February 1998, pp. 127-159.

9. ROSS, E., **Intelligent User Interfaces: Survey and Research Directions**, Technical report, CSTR-00-004, March, 2000, Department of Computer Science, University of Bristol (www.cs.bris.ac.uk/Publications/Papers/1000447.pdf)

10. WAERN, A., What is an Intelligent Interface? Available at www.sics.se/˜annika/papers/intint.html

11. WELD, D., C. ANDERSON, P. DOMONGOS et al., **Automatically Personalizing User Interfaces**, In: Proceedings of IJCAI-03, 2003, Acapulco, Mexico, available at www.cs.washington.edu/homes/weld/papers/weld-ijcai03.pdf.