RELVIEW – A System for Calculating With Relations and Relational Programming

Ralf Behnke, Rudolf Berghammer, Erich Meyer, and Peter Schneider

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
Preusserstraße 1–9, D–24105 Kiel, Germany

The calculus of relational algebra has its roots in the second half of the last century with the pioneering work on binary relations of G. Boole, A. de Morgan, C.S. Peirce, and E. Schröder. The modern axiomatic development is due to A. Tarski and his co-workers. In the last two decades this formalization has been accepted by many mathematicians and computer scientists as a fruitful base for describing fundamental concepts like graphs, combinatorics, orders and lattices, games in mathematics and like relational data bases and correctness as well as verification of programs in computer science. A lot of examples and references to relevant literature can be found in [6, 2].

Relational algebra has a fixed and surprisingly small set of operations. On finite carrier sets these operations and many others which are based on them can easily be implemented using, for example, two-dimensional Boolean arrays or successor lists. Moreover, concrete relations over finite sets can graphically be represented in a very elegant way as Boolean matrices and directed graphs. Therefore, a computer system for calculating with relations and relational programming can easily be implemented using modern user interface techniques.

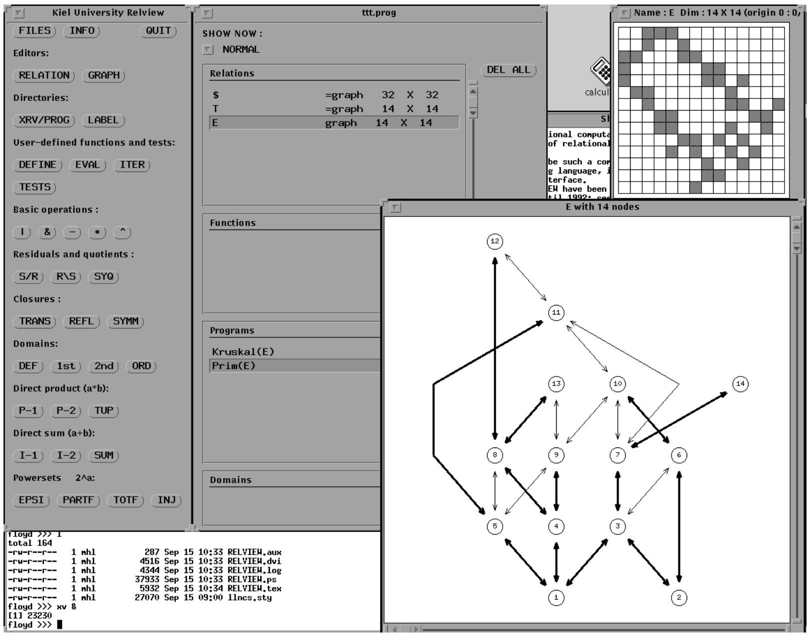
In the following, we give an impression of such a computer system, called RELVIEW. Written in the C programming language, it runs under X windows and makes full use of the graphical user interface. The first versions of RELVIEW have been written at the University of the German Forces Munich from 1988 until 1992; see [5]. Based on the experiences with the Munich system, in the last four years RELVIEW was redesigned and extended at Kiel University. It is available free of charge by ftp from host ftp.informatik.uni-kiel.de, where it is located in the directory pub/kiel/relview, and runs on Sun SPARC workstations and INTEL-based Linux systems. Additional information (e.g., a user's and programmer's guide with many examples [4]) and latest news can be found on the World-Wide-Web page http://www.informatik.uni-kiel.de/~progsys/relview.html.

RELVIEW can be used to solve many different tasks while working with relational algebra, concrete relations, relation-based discrete structures, and relational programs. First, it assists the formulation and the proof of relational theorems. In this field, the system can help to construct examples which support the validity of a theorem or to find counter examples to disprove the considered relation-algebraic property. For these activities, "playing" and "experimenting" with relational terms is essential and this is one of the purpose the RELVIEW system has been designed for. In particular, the interactive nature of the system allows to add, change and remove relations and directed graphs, and makes it possible to apply functions and to invoke relational programs at every time within a working session.

Relational program development is a second very important area for the use of RELVIEW. Whereas the relational algebra and especially the relational calculus forms the formal basis for ensuring correctness of the derived programs, the RELVIEW system supports the main validation tasks within nearly all stages of a development of a relational algorithm. In a first step of such a development, a relational specification is constructed, for example, from a logical problem description fixing a desired result of a computation. The resulting specification consists of a relational term which can be prototyped, i.e. evaluated, with RELVIEW. The evaluation not only allows to check the derived specification against the logical description, but also against the requirements of the given problem. Although the relational specification can be considered as an algorithm for solving the problem, it usually has to be improved in efficiency. Therefore, in a second step the specification can be used as a starting point for a formal program development. In spite of the correctness of a derived program established by relational proofs, it normally makes sense to prototype, i.e. execute or test, the resulting program to obtain additional safety.

Altogether, one can say that in the area of relational program development the RELVIEW system is a good means to bridge the gap between classical software engineering tasks like prototyping and testing and objektives of formal methods like assuring correctness.

RELVIEW is a totally interactive and completely graphic-oriented computer system. Although the detailed appeareance of its windows depends on resources defined in a configuration file, typically the four main windows (viz. the menu window, the directory window, the window of the relation editor, and the window of the graph editor) look as in the following screen snapshot:



After starting the system, the menu window is presented to the user and the command buttons are available. Conceptually they can be devided into three parts, viz. buttons for system administration (like loading from and saving into files), for function declaration, evaluation and iteration, and for the fundamental operations on relations and relational domain definitions.

The directory window presents the actual state of the system's workspace to the user. It contains four scroll lists showing the names and dimensions of the present relations and possibly existing graphs, globally declared relational functions, loaded relational programs, and, finally, globally declared relational domains. In the above picture, the workspace consists of two relations `E` and `T` and two relational programs `Prim` and `Kruskal`; there are no globally defined relational functions and no relational domains.

Using the window of the relation editor (resp. the graph editor), the user is able to edit a (homogeneous) relation as a Boolean matrix (resp. a directed graph) using the mouse pointer. In these windows also the results of the computations are shown, where the system provides some additional visualization means like nice drawings of graphs, a labeling mechanism for rows and columns of matrices resp. nodes of graphs, and facilities to emphasize nodes or edges of a graph. As an example, in the above picture the spanning tree with relation `T` of the depicted undirected graph with relation `E` is indicated by the boldface edges. The graph itself is drawn with the system's layer graph drawing algorithm.

The main purpose of RELVIEW is the evaluation of relational terms which are constructed from the relations of the workspace using predefined operations (like `-`, `^`, `&`, `|`, and `*` for complement, transposition, intersection, union, and multiplication) and tests (like `empty` for testing emptiness) on them, user-defined relational functions, and user-defined relational programs.

A declaration of a relational function is of the form $F(X_1, \ldots, X_n) = t$, where F is the function name, the X_i, $1 \leq i \leq n$, are the formal parameters (standing for relations), and t is a relational term over the relations of the workspace that can additionally contain the formal parameters X_i. As an example, an unary relational function `Hasse` which computes the Hasse diagram $R \cap \overline{RR^+}$ of a partial order R in RELVIEW is

$$\texttt{Hasse(R) = R \& -(R * trans(R))},$$

where a call of the predefined operation `trans` computes the transitive closure of its argument.

A relational program in RELVIEW essentially is a while-program based on the datatype of binary relations. Such a program – which is stored in a human-readable text file and can be loaded into the system with the help of an administration command – has many similarities with a function procedure in the programming languages Pascal or Modula-2. It starts with a head line containing the program's name and a list of formal parameters. Then the declaration part follows, which consists of the declarations of local relational domains, local relational functions, and local variables. The third part of a relational program is its body, a sequence of statements which are separated by semicolons and terminated by the return-clause. As an example, the following relational program uses Prim's method to compute the relation of a spanning tree for a nonempty, undirected and connected graph with (symmetric and irreflexive) relation `E`.

```
Prim(E)
  DECL T, v
  BEG  T = atom(E);
       v = dom(T) | ran(T);
       WHILE -empty(-v) DO
         T = T | atom(v * -v^ & E);
         v = dom(T) | ran(T)
         OD
       RETURN T | T^
  END.
```

In this program, the predefined operations `dom` and `ran` compute the domain resp. range of a relation. Furthermore, the base operation `atom` yields for a nonempty relation a subrelation which contains exactly one ordered pair.

In the last years a lot of case studies have been performed with RELVIEW. These include graph-theoretic questions and algorithms, Petri-net problems, computations on orders and lattices, data flow analysis, computations on finite automata, and relational semantics. See [3, 4, 1]. At Kiel University, RELVIEW was and is also applied in education, i.e., in lectures and seminars. These applications have shown that RELVIEW is a helpful tool for the interactive manipulation of relations and supports many different prototyping tasks within nearly all stages of a development of a relational program. Its main attraction is its flexibility since this property allows to experiment with new relational concepts as well as relational specifications and programs while avoiding unnecessary overhead.

References

1. Behnke R.: Extending relational specifications by sequential algebras – Prototyping with RELVIEW. In: Berghammer R., Simon F. (eds.): Programming languages and fundamentals of programming. Report 9717, Institut für Informatik und Praktische Mathematik, Universität Kiel, 12-22 (1997)
2. Brink C., Kahl W., Schmidt G. (eds.): Relational methods in Computer Science. Advances in Computing Science, Springer (1997)
3. Berghammer R., von Karger B., Ulke C.: Relation-algebraic analysis of Petri nets with RELVIEW. In: Margaria T., Steffen B. (eds.): Proc. TACAS '96, LNCS 1055, Springer, 49-69 (1996)
4. Behnke R., Berghammer R., Schneider P.: Machine support of relational computations. The Kiel RELVIEW system. Report 9711, Institut für Informatik und Praktische Mathematik, Universität Kiel (1997)
5. Berghammer R., Schmidt G.: RELVIEW – A computer system for the manipulation of relations. In: Nivat M., et al. (eds.): Proc. AMAST '93, Workshops in Computing, Springer, 405-406 (1993)
6. Schmidt G., Ströhlein T.: Relations and graphs. Discrete Mathematics for Computer Scientists, EATCS Monographs on Theoret. Comput. Sci., Springer (1993)

This article was processed using the LaTeX macro package with LLNCS style