# Increasing Explorativity by Generation

Stephan Diehl, Andreas Kerren

University of Saarland, FR 6.2 Informatik
PO Box 15 11 50, D-66041 Saarbrücken
{diehl,kerren}@cs.uni-sb.de

**Abstract.** Visualization of computational models is at the heart of educational software for computer science and related fields. In this paper we look at how generation of such visualizations and the visualization of the generation process itself increase explorativity. Four approaches of increased explorativity are introduced.

## 1 Introduction

Educational software mostly aims at width of knowledge (facts) and is not able to teach complex processes. The majority of systems are intrinsic electronic books, encyclopaedias and dictionaries. Process-oriented systems exist mostly for physics. In the area of computer science tutorials of programming languages prevail. Particularly in computer science and compiler design the theory and algorithms are very abstract and usually complex. Therefore visualizations are appropriate for computer science instruction. Although compiler design is often considered a practical field within computer science, most of its techniques are based on work in theoretical computer science, e.g. formal languages, automata theory and formal semantics. In recent years we have developed several educational software systems for topics in compiler design and theoretical computer science. These systems have in common that they teach computational models by animating computations of instances of these models with example inputs. But they differ in the level of explorativity.

| **Approach** | **Input** | **Computational Model** | **Generator** |
|---|---|---|---|
| Static | fixed | fixed | none |
| Interactive | user | fixed | none |
| First-order generative | user | user | yes |
| Second-order generative | user | user | yes/visualized |

**Table 1.** Levels of explorativity

Table 1 not only reflects the increased flexibility of the software developed, but also the chronological development of software in our group. Higher levels of explorativity demand more prerequisites and self-control by the learner. Thus, in the educational software the learner should start with static examples and as the user advances the level of explorativity should be increased. Exercises and textual hints in the educational software should guide the learner, to make sure he/she doesn't miss the important issues.

## 2 Approaches

*Static Approach* In the static approach the execution of an instance of a computational model is animated for a given, fixed input. The educational software "Animation of Lexical Analysis" [Braune et al. 99] is an example for this approach (http://www.cs.uni-sb.de/RW/anim/animcomp_e.html).

*Interactive Approach* In the interactive approach an instance of a computational model is animated for an example entered by the user/learner. An example for this approach is our application "Animation of Semantical Analysis" [Kerren 99] (http://www.cs.uni-sb.de/RW/anim/animcomp_e.html).

*First-Order Generative Approach* In the first-order generative approach the user enters the specification of an instance of a given computational model. Then an interactive visualization of this instance is generated and the user can enter an example input as in the interactive approach. As an example of the first-order generative approach we present GANIMAM [Diehl et al. 99], our web-based generator for interactive animations of abstract machines (Fig. 1 and http://www.cs.uni-sb.de/RW/users/ganimal/GANIMAM/).

Proceedings of the AACE World Conference on Educational Multimedia,
Hypermedia and Telecommunications (EDMEDIA '00), Montreal, Kanada, 2000.

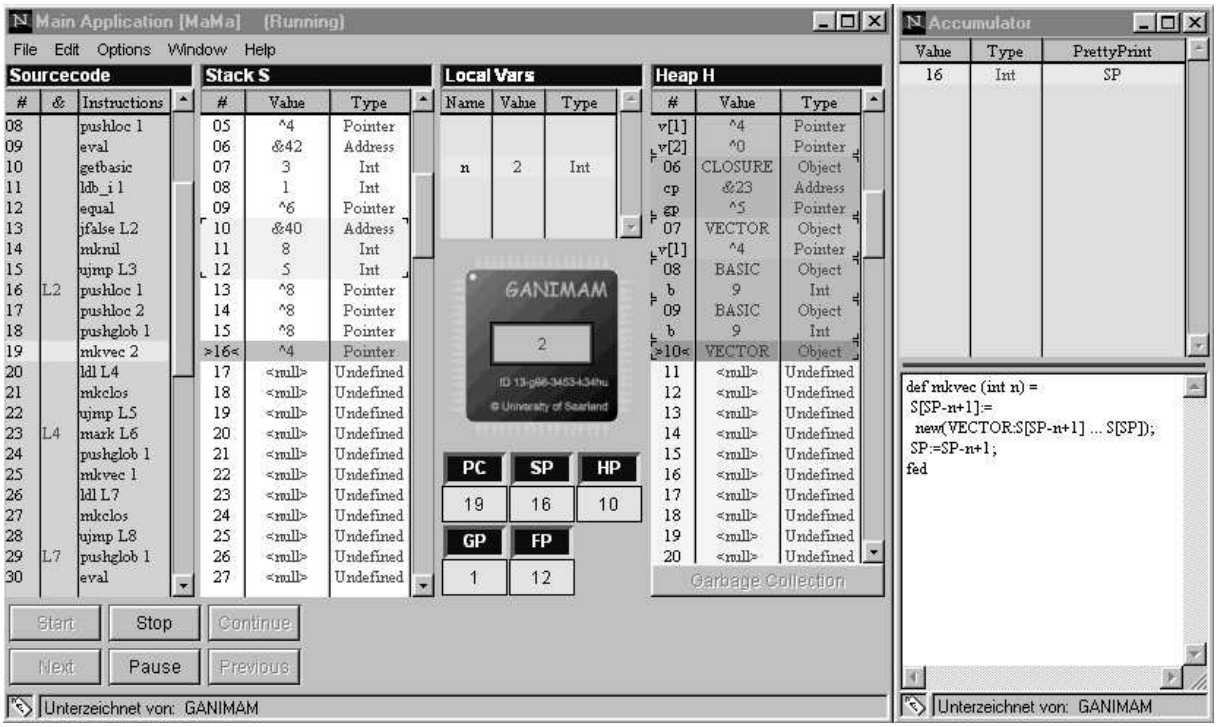


**Fig. 1.** Screenshot of an animated abstract machine

*Second-Order Generative Approach* As in the first-order generative approach the user enters a specification of an instance of a given computational model. But in the second-order generative approach in addition to visualizing the computation also the generation process is shown as an interactive visualization. An example of this approach is our application GANIFA, a web-based generator for animated finite automata (http://www.cs.uni-sb.de/RW/users/ganimal/GANIFA/).

## 3 Explorativity and Learner Control

In conventional educational software answers of an exercise are checked for correctness. Unfortunately such correctness checking is not possible in most interesting cases. In computer science, many properties of computational models can not be checked because of the halting problem. As a consequence we need alternative ways to provide feedback for the learner.

In the generative approaches an interactive animation is produced from the response (specification) of the learner. Then the learner can test it on the basis of own examples. In this way he/she can detect errors. The generative approach offers a way for explorative, self-controlled learning. The learner can focus on certain aspects in the generated, interactive animation and see what effects small modifications in the specification have. With the help of such observations he/she formulates hypotheses and checks these empirically. In the interactive approach such checkable hypotheses are restricted to the instance of the computational model. In the first-order generative approach also hypotheses about the computational model and in the second-order generative approach about the generation process itself can be checked.

## References

[Braune et al. 99] B. Braune, S. Diehl, A. Kerren, R. Wilhelm. (1999). Animation of the Generation and Computation of Finite Automata for Learning Software. To appear in *Proceedings of Workshop of Implementing Automata WIA'99*, Potsdam, Germany.

[Diehl et al. 99] S. Diehl, T. Kunze. (1999). Visualizing Principles of Abstract Machines by Generating Interactive Animations. To appear in *Future Generation Computing Systems*, Elsevier.

[Kerren 99] A. Kerren. (1999). Animation of the Semantical Analysis. In Proceedings of "8. GI-Fachtagung Informatik und Schule INFOS99" (in German), Informatik aktuell, Springer, pp. 108-120.