Diss. ETH No. 17479

Interactive Learning Environments for Mathematical Topics

A dissertation submitted to

ETH ZURICH

for the degree of

Doctor of Sciences

presented by

Ruedi Arnold
Dipl. Informatik-Ing. ETH

born June 23, 1976
citizen of Bürglen UR

accepted on the recommendation of

Prof. Dr. Friedemann Mattern, examiner
Prof. Dr. Werner Hartmann, co-examiner
Prof. Dr. Carl August Zehnder, co-examiner

2007

Abstract

In the information society an efficient and effective use of computers and their capabilities play a central role. An understanding of important computer science concepts should therefore be part of general education today. But concepts of computer science are often abstract and thus inherently difficult to understand and teach. Fortunately, it is precisely the computer itself that, due to dynamic visualizations, interactive simulations, and immediate feedback, allows an easier access to abstract topics.

As part of this dissertation the system "InfoTraffic" was developed which contains three new interactive learning environments targeted at propositional logic, queuing theory, and dynamic systems. The three programs are embedded into the common scenario of "traffic control" and consistently use a virtual-enactive and everyday-life-based approach to introduce abstract topics in education. InfoTraffic thus contributes among other things to an increase of the significance of logic in general education.

InfoTraffic was developed according to engineering science methods including insights from teaching practice and educational science. The experience gained lead to pragmatic recommendations for the development of interactive learning environments. Along with teaching materials the system is freely available online and it is already in use in high schools, universities, and teacher education.

Kurzfassung

In der Informationsgesellschaft kommt der effizienten und effektiven Nutzung des Computers und seiner Möglichkeiten eine zentrale Rolle zu. Ein Verständnis für wichtige informatische Konzepte gehört deshalb heute zur Allgemeinbildung. Informatische Konzepte sind allerdings oft abstrakt und daher inhärent schwierig zu verstehen und zu unterrichten. Es ist aber gerade der Computer selbst, der dank dynamischen Visualisierungen, interaktiven Simulationen und unmittelbaren Rückmeldungen einen leichteren Zugang zu abstrakten Themen ermöglicht.

Das im Rahmen dieser Dissertation entwickelte System "InfoTraffic" umfasst drei originäre interaktive Lernumgebungen zu Aussagenlogik, Warteschlangentheorie und dynamischen Systemen. Die drei Programme sind eingebettet in das gemeinsame Szenario "Verkehrssteuerung" und nutzen konsequent einen virtuell-enaktiven, das heisst handlungsorientierten, und alltagsbezogenen Ansatz zur Einführung abstrakter Themen im Unterricht. InfoTraffic leistet damit unter anderem einen Beitrag zur Erhöhung des Stellenwerts von Logik in der Allgemeinbildung.

InfoTraffic wurde nach ingenieurswissenschaftlichen Methoden unter Einbezug von Erkenntnissen aus der Schulpraxis und der Lehr- und Lernforschung entwickelt. Die gemachten Erfahrungen führen zu pragmatischen Empfehlungen zur Entwicklung von interaktiven Lernumgebungen. Das System ist samt Unterrichtsmaterialien online frei verfügbar und wird bereits an Gymnasien und Hochschulen sowie in der Lehrerausbildung eingesetzt.

Acknowledgements

Many people contributed to this work. My first and main thank belongs of course to Prof. Dr. Werner Hartmann – InfoTraffic and this dissertation would not exist without him. I was happy when I could join his group for teacher education and computer science at ETH Zurich, I felt sad when he left the department due to unfortunate circumstances, and since then I even more appreciated his continuous premium support of my PhD project. Werni, I am deeply grateful for all the e-mails, meetings, conversations, proof-readings, idea-sharing, visions, advice, and everything else! Thanks for letting me learn a lot from you – you simply are a great guy.

I am also indebted to Prof. Dr. Friedemann Mattern for having me in his research group for distributed systems. Thank you very much for all the support and assistance during the past two and a half years.

A big thanks goes to Prof. Dr. Carl August Zehnder for accepting to be co-examiner of my dissertation.

Thanks to the members of the distributed systems group for all the support and good times. I especially thank "my" post-doc Dr. Marc Langheinrich for proof-reading my PhD thesis as well as all the valuable hints, remarks and discussions over the years.

I am very thankful to all other proof-readers of this document, especially to Hans Fischer for (linguistically) reviewing the whole document and to Michela Pedroni and Christof Roduner for carefully proof-reading selected chapters.

I am grateful to all the students contributing to InfoTraffic, especially to the two master students Anna-Nina Simonetto and Nicolas Born, and also to the semester thesis students Anna-Nina Simonetto, Hasan Karahan, Marc Bühler, and Xiaoping Yin.

After a little more that five years at ETH Zurich, three professors and accordingly as many research groups, and about ten offices, I am finally done with the Swiss Federal Institute of Technology Zurich. I leave it with both a smiling and a crying eye. With the following list of friends and mates I met during my time here at ETH I would like

to express my gratefulness for having shared time and thoughts with me. Sorry for those I forgot, just contact me and I will add your name manually to your personal copy. – Is this ok with you?

Here we go: Thanks to "Das" Keno aka Dr. Albrecht for being a great first office mate and good friend through all these years. As one of the last free-living individuals of your species, you almost made spam have-been and were a great indoor climbing mate. Best wishes for Lisa's and your new collaborative project "Emma". Thanks to Judith Zimmermann for the friendship and the many fascinating discussions through all these times at ETH – good luck for your further career at this institution. Thanks to Hermann "Aquila Chrysaetos" Lehner, my time-wise best-organized diploma student ever, for all the coffee conversations about life, people, politics, tech-stuff, fun and everything else that matters. Thanks to Marc Langheinrich, Judith Zimmermann, and Eva Schuberth for co-leading the forum for women in computer science with me, the friendship, and all the experience I could gain in this position together with you. I further appreciated very much all the coffee break discussions and e-mails with Dr. Floris Tschurr – thank you very much Floris for all the insights into life, organizations, and humans and their behavior you offered me.

Additional thank for support, feedback, hints, and whatever else goes in alphabetical order to: Christian Sigg, Diana Jurjevic, Prof. Dr. Jürg Nievergelt, Lea Simonetto, Dr. Markus Brändle, Matthias Dreier, Michela Pedroni, Myke Näf, Nando Stöcklin, Nina Huggenberger, Dr. Raimond Reichert, Samuel Zürcher, and Prof. Dr. Stefan Wolf. Further I am grateful for all the feedback I received at the two international doctoral consortia in Paderborn and Zurich as well as at the ACM SIGCSE doctoral consortium in Covington. Many thanks as well for all the valuable feedback of teachers and students using InfoTraffic.

My final thank goes on the one hand to my parents and my family for making it possible for me to enter ETH Zurich and the academic world in the first place. On the other hand I thank my friends for sharing time with me, cheering me up, and distracting me from time to time (especially on weekends and holidays) with (outdoor) action, fascinating conversations, fun, going out, traveling, music, and so on... – you know who you are!

Zurich, November 2007

rarnold@wherever.ch

Contents

1 Introduction and Contributions

Over the last decades, computers and their communication capabilities have become increasingly important in our lives. Most people in developed countries have access to the Internet and use its services such as e-mail and web search almost daily. Even children use computers not just for playing, but also to write and print texts, to store, modify and share digital photographs, or to download music and organize a personal music collection. These activities mostly do not require an explicit knowledge of the underlying abstract concepts such as analog vs. digital, hierarchical organization, volatile vs. persistent memory, logical operators, divide-and-conquer, or synchronous vs. asynchronous communication, to name some. But in order to make use of the full potential, a deeper and explicit understanding of the underlying mathematical and computer science concepts is required. Everybody should therefore have a certain comprehension of these concepts. It is thus beneficial to include computer science education in general education and teach topics such as programming as a formal notation of algorithms, databases as a systematic and efficient way to organize information, or logic as the foundation of rational thinking and argumentation.

Most topics in computer science are abstract however, and thus in general difficult to understand. Fortunately, computers and their multimodal and communicative capabilities are well suited to support teaching and learning of abstract concepts by offering visualizations, interactive simulations, and immediate feedback.

We contribute educational software that makes use of these capabilities. This thesis presents InfoTraffic, a collection of three novel interactive learning environments targeted at introductory teaching of propositional logic, queuing theory, and Markov chains, respectively. These three teaching tools make use of *real-world-based* and *virtual-enactive* approaches to teaching abstract topics as they share the common scenario of traffic control and offer different interactive representations.

In this introductory chapter we briefly discuss the roles of computers in education. We summarize the motivation and the goals of our work

and give a short overview of the InfoTraffic project with its components LogicTraffic, QueueTraffic, and DynaTraffic. Finally we present the four main contributions of this thesis in condensed form.

1.1 ICT and Education

In our information society it has become increasingly important to master *information and communication technologies* (ICT), which is largely equivalent to mastering the "use of computers". Hartmann et al. [42] observe three different roles of computers in education:

ICT as a Tool: Tools such as word processors, spreadsheets or Internet services are used by most professionals today. In general this use does not require specific knowledge of computer science such as programming. Yet for an efficient use they require knowledge of some of its basic concepts.

ICT as a Medium: Computers and their multimodal and communicative capabilities can support teaching and learning. Here we distinguish two main modes: humans either use topic-specific educational software or they employ computers mainly to communicate with other humans. Well-known examples of the first use, the human-computer-interaction mode, include vocabulary training programs or interactive learning environments, e.g., to simulate the behavior of a complex ecosystem.

Furthermore, Internet-based communication opportunities offer anywhere and anytime learning. Common keywords are online learning, e-learning or blended learning. The focus of existing e-learning systems typically lies on the second mode, human-human-interaction, i.e., people discuss questions online or collaboratively edit texts.

ICT as a Subject: Important concepts of computer science like programming, algorithms, logical circuits, databases or networks are the content of computer science education. There is an ongoing debate on what concepts should be taught and how and where they should be taught, including for example arguments about the best programming language to start with.

As noted in [42], these three roles are generally not kept well apart of each other in the context of education. One often sloppily talks of

computer science education (CSE) and does not differentiate between the different roles, which potentially leads to unnecessary confusion und misconceptions. Therefore we first clarify the relation between our InfoTraffic system and the three roles of computers in education.

InfoTraffic addresses both *ICT as a medium* and *ICT as a subject*. On the one hand, InfoTraffic makes use of the advantages of the medium computer by offering *interactive* simulations and *dynamic* visualizations. On the other hand, InfoTraffic deals with important concepts of computer science and mathematics such as propositional logic or Markov chains.

1.2 Motivation and Goals

Although logic is of fundamental importance to everybody, we observe that logic is rarely a topic in today's curricula of general education, e.g., in Swiss or German high schools. As will be elaborated in chapter 3, the absence of logic education is partly due to the way logic has been taught, namely in a formal and abstract manner. Starting from this observation we investigate on how logic should be taught and introduced in general education. This inquiry leads to the main ideas for LogicTraffic, and we identify some other important concepts that can be nicely illustrated within the scenario of traffic control.

It is generally accepted today that teaching is particularly efficient whenever students can establish a connection between the subject to learn and their own experiences in everyday life. The key idea behind InfoTraffic was to facilitate learning by using scenarios from everyday life experience. We all understand safety at road intersections and realize that crashes might occur if two intersecting lanes both show green traffic lights simultaneously. It is also obvious to us that traffic jams occur if there are too many cars on the street. We understand as well that we can predict the distribution of cars if we have a simple model of probabilities for their distribution. InfoTraffic uses these examples to introduce propositional logic, queuing theory, and Markov chains. With the presented scenario of traffic control, these three rather abstract topics can be perfectly accommodated.

Providing interactive software, readily available for teachers, along with teaching materials (e.g., introductory presentation, problems, and solutions to the problems), InfoTraffic supports the goal of bringing logic back to schools and general education. InfoTraffic is available at

www.swisseduc.ch/compscience/infotraffic/.

As another result of the InfoTraffic project we have gained insights into how to teach abstract topics, by using our approach with sample applications based on real-world examples. This approach can be generalized and used in other areas and for other topics. It leads to shareable experience in the development of state-of-the-art interactive learning environments for computer science.

1.3 The InfoTraffic Interactive Learning Environments

InfoTraffic [2, 4, 5, 7] is a collection of new *interactive learning environments* (ILE). An ILE is interactive educational software with the purpose to facilitate teaching and support learning by taking advantage of the capabilities of computers such as simulation, visualization or giving feedback; see for example [29, 35, 69].

The ILEs of InfoTraffic share the common scenario of controlling traffic systems and allow a gentle introduction to the following abstract topics of computer science and mathematics:

Propositional Logic: The LogicTraffic ILE uses formulas in propositional logic to describe safe traffic light settings at an intersection. The control of traffic lights serves as a well-suited example for the use of propositional logic. Truth tables and formulas receive an evident everyday-life meaning, and further concepts such as equivalence or normal forms of formulas follow naturally. LogicTraffic is covered in detail in chapter 3.

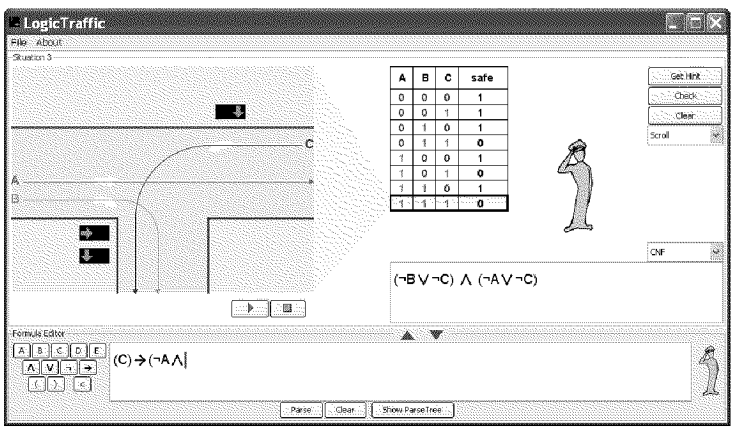

Figure 1.1: Screenshot of LogicTraffic

Queuing Theory: The QueueTraffic ILE is an easy to use yet powerful tool to simulate and analyze traffic jams at an intersection. Traffic jam is a well-known phenomenon. Important parameters of queuing theory such as the arrival rate of cars or the duration of green traffic lights for individual lanes can easily be modified. Through simulation QueueTraffic allows a visual impression of what is going on and an in-depth analysis by providing relevant information like the current utilization of a lane or the number of cars waiting in the system. QueueTraffic is outlined in chapter 4.

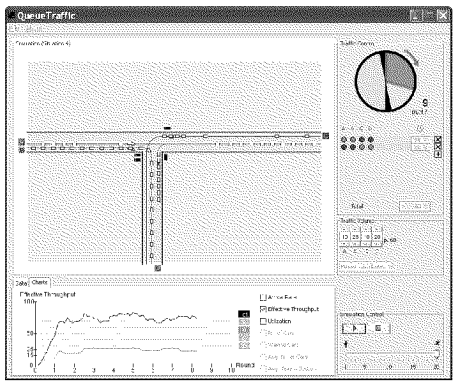

Figure 1.2: Screenshot of QueueTraffic

Markov Chains: The DynaTraffic ILE uses Markov chains to model the distribution of traffic in a system with several intersections, predicting the number of cars on certain streets. Important parameters such as the transition probabilities and the state vector can easily be modified. Through different but corresponding views, the program allows a step-wise analysis of the behavior of the underlying Markov model. Chapter 5 is dedicated to DynaTraffic.

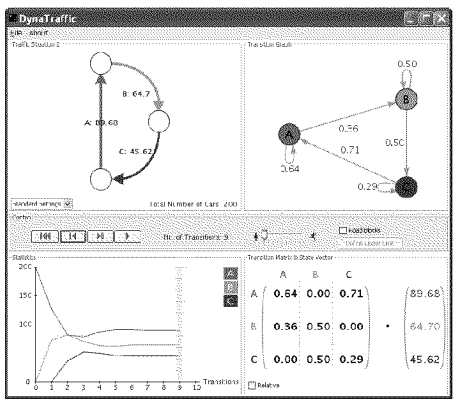

Figure 1.3: Screenshot of DynaTraffic

As stated above, all three ILEs are about control of traffic. They differ in the view (local vs. global) and the time (static vs. dynamic) of their observations of traffic systems, as illustrated in figure 1.4. LogicTraffic is concerned with the static case at a single intersection, i.e., snapshots in a local view. This naturally leads to the question: Which traffic light settings are safe? QueueTraffic analyzes the dynamic case at a single intersection, i.e., an observation over time in a local view, leading to the question: Do waiting queues occur? DynaTraffic finally is about the dynamic distribution of traffic in situations with more than one intersection, i.e., an observation over time in a global view: How many cars are there on the streets at a given instant in time?

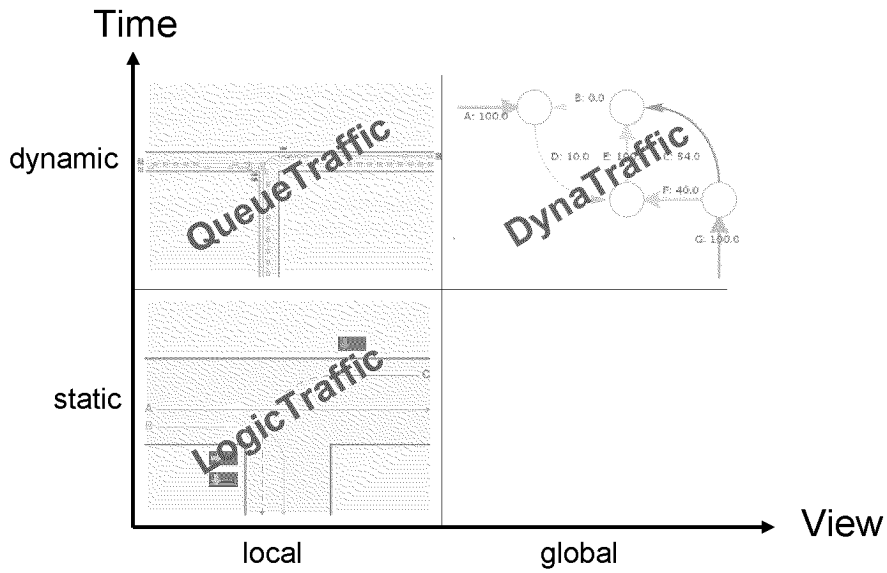

Figure 1.4: InfoTraffic with respect to *local vs. global* and *static vs. dynamic* view

The three ILEs of InfoTraffic can be used in class independently of each other, whenever they fit into the curriculum. The three programs do not build up on each other and come with independent ready-made teaching materials. It is also possible however, to use the common scenario of the three programs as an ongoing example to show important concepts hidden behind the control of traffic systems and to show the importance of computer science and mathematics in everyday life.

1.4 Contributions of this Thesis

The motivation for InfoTraffic is on the one hand to develop, provide and test a collection of new interactive learning environments. On the

other hand we seek to gain further insights into how to teach abstract topics and into how to implement ILEs for such topics.

A Real-World-Based Approach to Abstract Topics

Introducing abstract topics is generally known to be difficult. It is usually difficult to motivate the topic and to outline practical applications. Precisely because of this difficulty we recommend to start with real-world-based examples for introducing abstract topics as one possible promising way. This idea is not new, but we are among the first to strictly apply this approach to the design and the implementation of novel interactive learning environments for propositional logic, queuing theory and Markov chains.

We support our approach by extending the two well-known and widely used didactic techniques of *rule-eg-rule* and *advance organizer*. Our extended rule-e.g.-rule technique gradually builds on the students previous knowledge and experience from everyday life.

Guidelines for Pragmatic and Interdisciplinary Engineering of Interactive Learning Environments

The design, development and evaluation of interactive learning environments is a laborious task and it involves knowledge from three different areas: *software engineering*, *educational science* and *teaching practice*. Thus an interdisciplinary approach is required. We suggest an engineering science approach: take the best findings from all the three fields and build computer-based learning environments that are widely used in practice.

This interdisciplinary approach has proven successful in the past for several projects developing ILEs for computer science at ETH Zurich. We have condensed this past experience and our insights from the InfoTraffic project in ten pragmatic recommendations for the development of ILEs, as presented in chapter 6.

The complexity of interactive learning environments with its many variables sets tight boundaries to an evaluation by traditional methods of educational research. According to Schulmeister [69], putting the focus unilaterally on such evaluations often leads to artificial learning environments that are of no relevance to teaching practice. Intentionally we did not conduct a large evaluation study but put emphasis on

the impact in practice. While we also have conducted constant evaluation to improve InfoTraffic, it is primarily based on anecdotic evidence.

A Virtual-Enactive Introduction to Abstract Topics

Introducing and teaching the basics of logic is well known to be difficult because of the high level of abstraction inherent in logic. LogicTraffic offers an easy and obvious approach to logic, based on the visual representation of logic in an everyday situation (control of traffic intersections) and the possibility to interactively manipulate logical formulas and propositions. The concept of virtual-enactive [42] and iconic representations [27] is one of the key ideas behind InfoTraffic.

Increasing the Significance of Logic in General Education

A comprehension of logic is fundamental for decision-making, analyzing relationships and any other kind of rational arguing. Today though, logic has little significance in school practice. Responsible for this shadowy existence of logic in curricula is the fact that logic is and has commonly been taught in an abstract manner and reduced to a mathematical set of formulas. With the ready-made program LogicTraffic and its accompanying teaching materials we present a new tool for the introductory teaching of logic in general education. LogicTraffic offers a feasible and attractive approach to propositional logic. LogicTraffic thus contributes to a strengthening of computer science and mathematics instruction in general education.

1.5 Outline

The following chapter elaborates on why the three topics covered by InfoTraffic are *fundamental ideas* of computer science and on further important didactic concepts behind the InfoTraffic project. The chapters 3, 4 and 5 present the three new ILEs LogicTraffic, QueueTraffic, and DynaTraffic. For each of the three ILEs we elaborate on the topic covered and the targeted learning goals, present the program and discuss related work. Transferable insight we gained in the process of developing ILEs is presented in chapter 6. The pragmatic recommendations given in this chapter are based on experience from developing InfoTraffic as well as on know-how from the tradition of CSE research, especially in the development of ILEs, at the department of computer

science at ETH Zurich [21, 61, 77]. Chapter 7 reports on our evaluation of InfoTraffic in the context of different approaches to educational research. A final summary and conclusions are presented in chapter 8.

2 Didactic Concepts for Interactive Learning Environments

Interactive learning environments have the purpose to facilitate teaching and support learning by taking advantage of the capabilities of computers. When designing such educational software, a number of important issues have to be addressed. Identifying and settling these issues make the development of high-quality interactive learning environments in computer science a demanding task. A first crucial issue is the topic covered by the learning environment. Additional important issues to be addressed are suitable examples and a sensible incorporation into instruction. Furthermore a learning environment should not be restricted to mere animation, but it has to stimulate learners to investigate the topic by offering a high level of interactivity. In this chapter we point out the main didactic issues to be addressed when developing interactive learning environments and illustrate them with the example of InfoTraffic.

2.1 Addressing Fundamental Ideas

As with any teaching material, when developing a learning environment choosing the topics is one of the most relevant decisions to take. The development of interactive learning environments such as in InfoTraffic only makes sense and is worth the effort if it covers fundamental topics that ensure its longevity and widespread usage. Bruner [26] gives vague definitions of *fundamental ideas*. Schwill [70] bases on Bruner's work and provides four criteria for fundamental ideas of computer science. Schwill describes a fundamental idea with respect to some domain as a scheme for thinking, acting, describing, or explaining which

Horizontal Criterion: is manifold applicable or recognizable in different areas.

Vertical Criterion: may be demonstrated and taught on every intellectual level.

Time Criterion: can be clearly observed in the historical development and is relevant in the long term.

Sense Criterion: is related to everyday language and thinking.

The consideration of these criteria of fundamental ideas stimulates the selection of content which is relevant, long-lived, and cognitively demanding (e.g., according to the taxonomy of Bloom [17]). The topics for the InfoTraffic learning environments – propositional logic, queuing theory and Markov chains – are fundamental topics in computer science, as we will show below.

Note that Hartmann et al. [42] add a fifth criterion "representation" to the four criteria of Schwill [70]. Since section 2.4 in this chapter is dedicated to different representations, we do not take this fifth criterion into consideration here.

Propositional Logic as a Fundamental Topic

Propositional logic (PL) with concepts such as propositions, operators, truth tables, formulas and equivalence is of vital importance for any kind of rationality. If we do not obey the principle of bivalence or do not follow the concept of logical inference, we are not able to argue or act rationally. According to the four criteria given above, PL is a fundamental topic:

Horizontal Criterion: PL has obviously applications in many different areas, for example computer hardware, queries in online search engines or logical inference in natural language.

Vertical Criterion: PL can be demonstrated and taught on every intellectual level. Even a child understands that a certain sentence is either true or false and that sentences can be combined to build more complex sentences or formulas. Humans automatically learn the basic concepts of PL. On the other hand, dealing with formulas in PL leads quite naturally to more complex questions such as satisfiability or efficient testing of equivalence.

Time Criterion: A fundamental idea had to have been valid ten years ago and still has to be relevant in ten years. This is certainly the

case for PL, being the fundament of rationality and therefore for all sciences. There have for example been no mathematical proofs without the existence of PL.

Sense Criterion: PL is related to everyday language and thinking. Natural language and formulas in PL share for example constructs such as "or" and "and", though sometimes with different semantics, as pointed out in [4].

Queuing Theory as a Fundamental Topic

The theory of queues with concepts such as arrival rate, Poisson distribution, or utilization is a mathematical model for describing and analyzing phenomena happening in our world. According to the four criteria given above, queuing theory is a fundamental topic.

Horizontal Criterion: Queuing theory has applications in many different areas, be it physical-real (e.g., traffic jam or cashier desk in the supermarket), physical-virtual (e.g., post office with number system), or digital-virtual (e.g., congestion at a web server or job queue at a printer).

Vertical Criterion: Queues can be explained to a child, it will understand that cars have to wait if there are too many of them on the street. On a higher intellectual level, even the simplest mathematical models to analyze queues require some mathematical background.

Time Criterion: Queues continue to be relevant. As long as we need any kind of capacity planning and as long as humans keep or increase their physical mobility (e.g., cars, planes) and virtual connectivity (e.g., Internet, telephony), queuing theory will remain important.

Sense Criterion: Queuing theory is related to everyday language and thinking. We often speak of traffic jams when cars queue on streets. We speak of systems, e.g., airports or the telephony system, being overloaded. Queuing theory is used to optimize processes and can directly affect the behavior of people, e.g., when doing queue analysis and applying personal "fast-track heuristics" in order to chose the most suitable queue in a supermarket.

Dynamic Systems as Fundamental Topics

Our world is one big and very dynamic system. Many subsystems, such as for example the global stock market or the atmosphere are often modeled as dynamic systems. Also smaller systems such as a forest and the population of animals can be seen as dynamic systems and can be modeled as Markov chains. We demonstrate why concepts of dynamic systems, and more specifically why Markov chains are a fundamental topic in mathematics.

Horizontal Criterion: Many real systems are modeled as dynamic systems. Markov chain models are for example used to forecast our weather, to predict the development of our stocks, or to prognosticate the development of the number of the population of a certain country.

Vertical Criterion: Dynamic systems can be demonstrated and taught on every intellectual level. Children understand simple predator-prey dependencies such as "if there are many hares, the number of lynxes increases because they have a lot of hares available for food", which are easily modeled with a Markov chain. And of course there are systems of almost arbitrary size and mathematical complexity.

Time Criterion: Weather, stock market and population predictions have been of interest for a long time. People will be concerned about tomorrow's weather and politicians about the population of their countries in the future too.

Sense Criterion: Dynamic systems and models are related to everyday language and thinking. Terms such as *mutual dependency* or *steady state* are often used and most people have at least a vague understanding of what they mean.

Having defined the topics covered by a learning environment, a next important issue are examples and applications. Suitable examples and meaningful applications help support a theory and allow in general the creation of reasonable assignments.

2.2 Abstract Topics and Real-World Examples

The concepts covered by the InfoTraffic learning environments are abstract. Boolean operators and equivalence of Boolean formulas in Log-

icTraffic, utilization and arrival rate in QueueTraffic, or steady state and transition matrix are abstract concepts.

InfoTraffic gives an introduction to abstract concepts with everyday-life based examples. Anderson et al. [1] state in their article on *situated learning* that "numerous experiments show combining abstract instruction with specific concrete examples is better than either one alone" and summarize that "abstract instruction combined with concrete examples can be a powerful method."

According to the constructivist learning theory [10], the construction of new knowledge is performed individually, based on previous knowledge. This process of acquiring knowledge is more difficult for abstract topics because learners lack real world objects or problems to relate to. Trichina [76] who built a learning environment for Turing machines states that "cognitively plausible representation is a fundamental problem to every theory of instruction". Educational software supporting the introduction of abstract topics should therefore find and provide good representations to facilitate the construction of knowledge. As Trichina [76] puts it: "...one has to develop tools that make students aware of their own thinking and abstraction process by gradually building on the students' previous knowledge and experience."

For the InfoTraffic learning environments we use the example of traffic control, based on two reasons. First of all, everybody is familiar with traffic control: everybody knows what happens if two intersecting lanes show green traffic lights simultaneously and everybody has experienced traffic jams. Secondly, traffic control illustrates many abstract concepts. Of course, other everyday-life based examples might serve for the same purpose too. We do by no means claim that traffic is the only scenario that works.

Using terms of the *anchored instruction* [24] learning paradigm, InfoTraffic uses traffic control as interesting and realistic *anchor* (macro-context) for the instruction of abstract concepts of computer science. Blumstengel [18] states about this anchor: "Er soll Interesse wecken und Wahrnehmung und Verständnis des Lernenden lenken. Dadurch wird die Bedeutung des zu erwerbenden Wissens in der Anwendung herausgestellt."

Introducing abstract topics such as the ones targeted at by the InfoTraffic learning environments, educational software might do its best job. Sound examples help students understand abstract topics in their own mental structures. Rather than linearly following the trail laid

down by an author or a lecturer, students can explore this structure themselves. Often an understanding of the basic concepts and an intuitive knowledge is sufficient. This kind of approach has been around for years. We do not teach children the law of gravity by showing them physical formulas, but simply by showing them the effect of gravity. We let them discover! It is therefore not surprising that we have used LogicTraffic as an example for *discovery learning* [6] in presentations and workshops.

2.3 Extending the Rule-e.g.-Rule Technique

The sole existence of an attractive learning environment does of course not improve teaching quality. It is imperative that any such tool is properly incorporated into the classroom, e.g., by providing corresponding teaching materials. The InfoTraffic learning environments not only come with a set of accompanying teaching materials [3], but also introduce a novel lecture organization, which we call the *e.g.-rule-e.g.-rule* technique. Common lectures are often organized based on the *rule-e.g.-rule* technique, see figure 2.1. According to Bligh [16] the first *rule* stands for a concise statement, display, and re-expression of the topic. Bligh describes the *e.g.*-part (example) as elaboration (detail, illustration, reasons and explanations, relations, examples) and feedback. The second *rule* is then a summary with recapitulation and restatement.

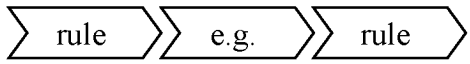

Figure 2.1: The widely used rule-eg-rule technique for teaching

For abstract topics, it is often not suitable to begin with a rule, i.e., with a concise statement displaying and summarizing the main concepts in an condensed form. For such cases, we suggest to prefix the initial overview with an introductory example based on a real-world experience. Our extended *rule-e.g.-rule* technique looks thus as depicted in figure 2.2.

Figure 2.2: The e.g.-rule-e.g.-rule technique extents the rule-eg-rule technique

Beginning with a real-world example, students get a first concrete impression of the topic and can relate it to their prior knowledge. This introduction is then followed by Bligh's first *rule*, i.e., a concise statement of the abstract theories and concepts, another elaboration, and the final summary.

The same idea lies behind the well-established teaching technique of *advance organizers* [8], where the main concepts of new topics are taught *concisely* at the beginning of a lesson, before moving on to the actual details. Advance organizers are particularly valuable if they use an analogy to knowledge or real world experience. This way, learners can not only connect new topics to prior knowledge, but are also less afraid of abstract topics.

Our idea is to go one step further, and not only connect abstract topics to prior knowledge, but also connect abstract topics to prior knowledge based on real-world applications.

An introductory real-world example helps students to comprehend abstract topics. Additional support results from using different representations and thus allowing different approaches to a topic, as presented in the following section.

2.4 Different Representations

Abstraction is a useful and powerful method to summarize complex concepts, yet it is often still demanding and requires a significant level of experience. While we learn over time to accept and understand explanations given to us just in pictures or as text, we started out as children learning from concrete objects and experiences only. Even though after finishing school we are well accustomed to conceptualize events and procedures solely in our imagination, we still appreciate concrete and relevant examples from our real-world experience.

Propositional logic, queuing theory, and dynamic systems are typically taught as abstract and formal topics, but less abstract representations might introduce these topics more readily. Does it really help if our first knowledge of logic is that "1" stands for "true", "$\wedge$" for "and", and "$1 \wedge 0 = 0$" holds? Do we really understand the relevance of queuing theory if it is introduced as a M/M/1 system in Kendall's notation? Or do we gain much understanding from reading that each ergodic finite Markov chain has a steady state? Finding representations that are more appealing and realistic than a formal introduction to abstract

concepts plays a major role in the process of teaching abstract topics.

Based on Piaget [59], Bruner et al. [27] introduce the theory of three basic modes of instruction, leading to three different representations; see figure 2.3. These modes can be seen as one possibility to move away from abstraction and mere formalism in teaching. We summarize the three representations here and extend them by a fourth one as introduced by Hartmann et al. [42].

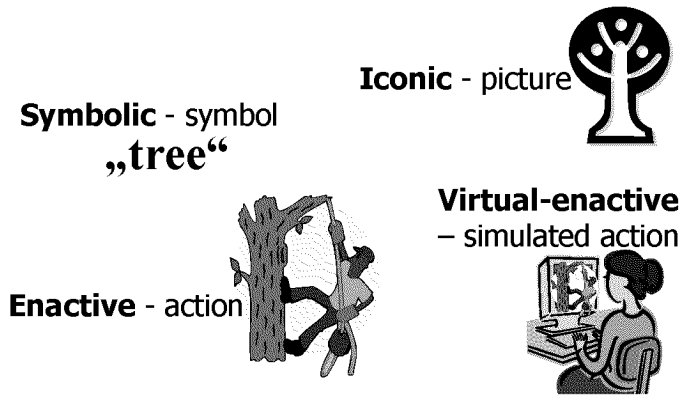

Figure 2.3: Illustration of the four different representations

Symbolic Representation: Acquire knowledge through symbols (e.g., text or signs). Symbolic representations are concise and are especially suitable if one has already an appropriate intuitive perception. Most people do neither need to climb a tree nor see a picture of a tree to imagine what "tree" stands for.

Iconic Representation: Facts are represented as pictures. Concrete objects, events or procedures are understandable as visualizations. A hotel brochure or a city map often suffices to get the picture and to orient oneself.

Enactive Representation: Learning by doing. This is especially the case for children, as they learn through their own actions, grouping of objects, and through observation. Kids need no manual for a tricycle.

Virtual-Enactive Representation: Through manipulation in a software environment enactive processes are simulated. Popular examples are learning environments where students can control virtual robots on their screens.

The enactive representation is especially useful for an introduction to a new topic. Through the students own actions, the subject matter becomes better accessible and better anchored in the learners mind.

Representations in InfoTraffic

These basic modes of representations are repeatedly employed throughout the InfoTraffic learning environments. In LogicTraffic, formulas and truth tables offer a symbolic representation, the static picture of a traffic situation corresponds to an iconic view, and through mouse-clicks on traffic lights and animation of the situation a virtual-enactive mode is achieved. In QueueTraffic, numeric data sets and charts give a symbolic representation of the situation. The static traffic situation and the phase clock are iconic, and through simulation, a virtual-enactive mode is reached. In DynaTraffic, the state vector and the transition matrix offer a formal representation, whereas the graphs with vertices and edges (arrows) provide an iconic representation. The macroscopic simulation of traffic (see section 4.5) gives an abstract form of virtual-enactive representation in DynaTraffic since only the accumulated numbers of cars change in the simulation.

As seen in this section, educational software providing different representations allows different approaches to a topic. Closely related to this issue is the question "How can I modify the representation or the content of the representation?" - i.e., the issue of interactivity. In general, a learning environment becomes more attractive, the more user interaction it allows.

2.5 Providing Interactivity and Immediate Feedback

Good educational software is according to Hartmann and Reichert [62] usually characterized by a high degree of interactivity. Already Kay [47] stated about computers in education that "the first benefit is great interactivity". We use a model by Schulmeister [68] that defines six levels (see figure 2.4) of increasing human-computer interaction. Level one means no interaction at all, only the display of information. Level two lets users navigate through the representation of information. Level three offers multiple representations of the content. On level four, the user can modify parameters of the representation. Additionally, on

level five, the user can manipulate the content itself. Level six means the user can create and manipulate objects and investigate how the system reacts, i.e., gets feedback.

Level	Definition
6	Constructing the object or contents of the representation and receiving intelligent feedback from the system through manipulative action
5	Constructing the object or representation contents
4	Manipulating the component content
3	Varying the form of representation
2	Watching and receiving multiple representations
1	Viewing objects and receiving

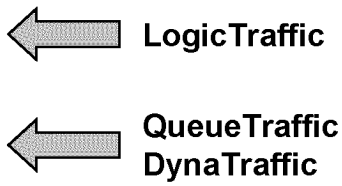

Figure 2.4: The six levels of interactivity in the taxonomy of Schulmeister

The InfoTraffic learning environments have been developed with a high level of interactivity in mind. According to Schulmeister's definitions above, LogicTraffic, QueueTraffic, and DynaTraffic all reach an interactivity level of four, as all three allow students to modify the simulation parameters. LogicTraffic additionally offers students to directly manipulate its formulas, which corresponds to an interactivity level of five. LogicTraffic further gives feedback about the current safety state, thus reaching level six.

Note that only few computer-based learning environments yield a high degree of interactivity. One reason is the high cost of development. As Berg [11] notes: "Highly interactive software using simulation strategies is almost non-existent in higher education. Clearly the cost of developing such software is a barrier." Only educational software that focuses on fundamental ideas has the potential to amortize the high cost of development.

2.6 Automatic Update of Corresponding Views

One major advantage of learning environments is their ability to provide multiple visualizations of the same content and to update these corresponding views automatically as soon as the content changes. This behavior offers the students new insights and allows different approaches to the content, depending for example on the students' cognitive preferences and capabilities. Corresponding views are much

deployed in the InfoTraffic learning environments and have for example as well been used extensively in the GraphBench project [21, 23]. Brinda [25] presents a similar *views concept* (Sichtenkonzept) in the context of learning environments targeted at an introduction to object oriented modeling. We now present corresponding views given in the three InfoTraffic learning environments.

One of the main correspondences in LogicTraffic is the current traffic light setting to the accordant row in the truth table. This correspondence is emphasized with a yellow trapezoid and a yellow underlaid row in the truth table (see figure 2.5). It works in both directions, i.e., if the student marks a certain row in the truth table, the traffic lights in the situation change accordingly and vice versa. This correspondence works analogously for the parse tree visualization in LogicTraffic. Another important correspondence is the one between the truth table and the formula corresponding to the truth table. Whenever the truth table configuration changes, the formula changes accordingly and vice versa. A last correspondence is the status indication, the icon of a policeman in different poses showing the safety state of the current truth table or formula.

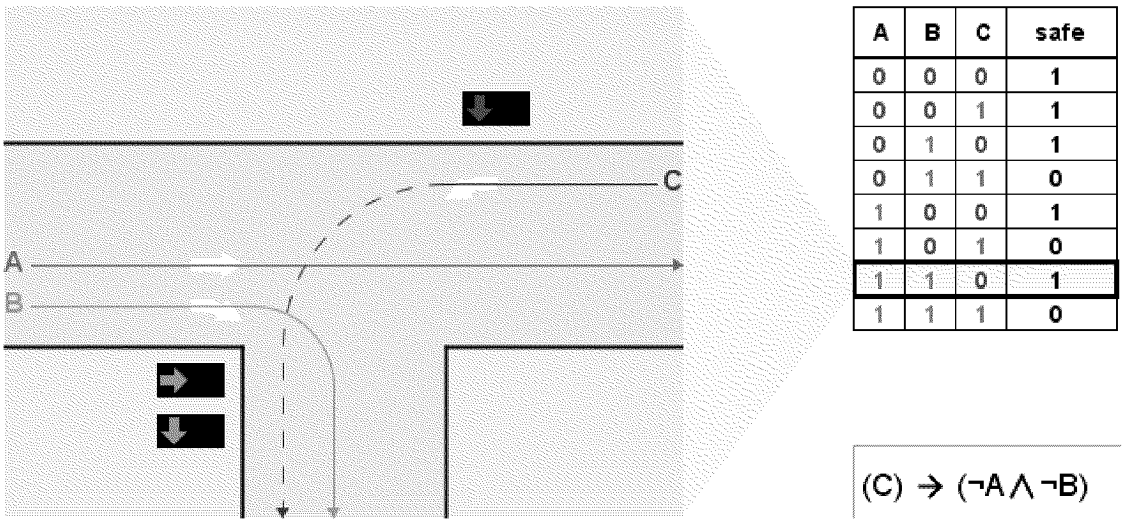

Figure 2.5: The correspondence between the traffic light setting and the selected row in the truth table. - LogicTraffic screenshot

The simulation in QueueTraffic has an obvious correspondence between the traffic light settings in the situation and the current phase. The main correspondence in QueueTraffic is the one between the visualization of the traffic situation and its corresponding numeric entries in the data and charts tab. Students can for example either directly

see in the situation that there are n cars waiting, they get this information from the data tab, or they can read this number from the charts tab. Again, all these views are updated automatically as the simulation runs.

DynaTraffic offers corresponding views between situations and their corresponding transition graphs, i.e., situations always correspond to the transition graph and vice versa. There are many different correspondences in DynaTraffic between the traffic situation, the transition matrix, the statistics and the transition graph. The colors of vertices in the transition graph for example correspond with the colors of the corresponding arrows in the situation, the colors of the labels in the transition matrix and the colors of labels and charts in the statistics. If a value in the situation is changed, this is immediately updated in the state vector too. Or if a value is changed in the transition graph, the corresponding entry in the transition matrix is changed. If the mouse moves over an edge or a vertex in the transition graph, the corresponding entry or row in the transition matrix is highlighted.

All these correspondences show and emphasize the connections and dependencies between the different views and thus allow students a deeper understanding of the presented concepts.

2.7 Conclusions

InfoTraffic incorporates best-practice knowledge and state-of-the-art findings from educational science. The InfoTraffic learning environments cover fundamental ideas of computer science and thus justify the cost of development. Abstract topics are presented based on real-world examples, allowing students to connect them to previous knowledge and experience. For usage in class, we further present an e.g.-rule-e.g.-rule technique which can as well be seen as an extension of an advance organizer. Our new learning environments provide formal, iconic and virtual-enactive representations, automatic updates of corresponding views, and a high level of interactivity.

Through simulation and analysis of everyday traffic situations and the manyfold possibilities to interact with the software, InfoTraffic is attractive for young people and addresses the "Nintendo generation" as described by Guzdial and Soloway [41].

According to Hartmann and Reichert [62], "The use of educational software can enrich the process of learning, provided that there is a

pedagogically sound concept for its use." With the didactic concepts for interactive learning environments presented in this chapter, we are convinced to have created such a pedagogically sound concept.

3 LogicTraffic: Safe Intersections and Propositional Logic

"It's logical!" – Logic is omnipresent in our everyday life. In schools however, logic is barely a topic and if it is, the classes are mostly unnecessarily abstract. This chapter introduces the learning environment LogicTraffic for controlling traffic intersections as an intuitive approach to logic as part of general education.

3.1 The Importance of Logic

The relevance of logic for general education is indisputable. Without logic we could not argue rationally. It is clear that a statement like "Zürich is the capital of Switzerland." is either true or false. And from the statement "It is raining cats and dogs" we conclude that the street is wet. From an educated person we expect as well that she masters complicated conclusions. From "Either Peter or Roger plays tennis" and "Peter does not play tennis" we conclude "Roger plays tennis". We use similar simple propositions to state queries in search engines. But as Jansen et al. [45] point out, many people fail at entering correct queries using Boolean operators. Stating correct propositions thus might not be that simple after all.

Although logic accompanies us in our everyday life, the concepts and terms behind it are often only intuitively clear. And Logic is nevertheless not part of general education. In their series of essays "Informatik als Grundbildung" Wedekind et al. [81] make this phenomenon clear: "Mit unserer natürlichen Sprache teilt die Logik das Schicksal, gewissermaßen wildwüchsig erworben zu werden. Dass dieser Erwerb der korrigierenden und fördernden Ergänzung durch die Schule bedarf, ist im Fall der Sprache eine Selbstverständlichkeit, im Fall der Logik jedoch nicht auf der Agenda."

Besides everyday life there are scientific areas where logic is of central importance. We are going to list three prominent representatives.

Logic is very important in mathematics. In fact, Logic and mathematics are tightly coupled. If one talks of logic, often mathematical logic such as in *propositional logic* (PL) or in formal systems is meant. Logic provides precise and formal notations for mathematics and therefore provides a toolbox, e.g., for mathematical proofs. There would be no mathematical proofs without logic.

Electrical engineering is tightly coupled with logic. Modern integrated circuits are nothing but formulas in PL cast into hardware; variables and operators are mapped to registers, gates and circuits.

Logic is central to practically every subarea of computer science. As soon as processes or data become formalized and thus can be executed or processed by a computer, there is no way around clear and unique notations and semantics. Denning [36] characterizes computer science with five "windows of computing mechanics". Table 3.1 shows exemplarily the role of logic in these five areas. The list of example applications of logic could easily be extended. This illustrates the importance of logic for computer science and indicates the close connection between logic and computer science.

Window	**Example application of logic**
Computation	Proofs, runtime analysis, program verification
Communication	Coding (e.g., CRC with XOR)
Coordination	Fuzzy Logic in HCI, mutex proofs for deadlocks
Automation	Artificial intelligence, logic programming
Recollection	Querying with logical operators (e.g., SQL queries)

Table 3.1: Logic in the five windows of computing mechanics

3.2 Logic in General Education

Even though logic is very important, it is commonly neglected in general education. Wedekind et al. [81] write about the reasons:

> Die Logik lädt in ihrer üblichen Darstellung nicht gerade zu einer näheren Beschäftigung mit ihr ein. Führende Logiker

> sprachen und sprechen zwar vom "natürlichen Schliessen", ein Blick in ein Logiklehrbuch zeigt aber in der Regel Zeichen und Formeln, die alles andere als "natürlich" aussehen. Dazu kommt eine Reihe von "Prinzipien", die man als plausibel, selbstverständlich, fraglos gültig o. ä. zur Kenntnis zu nehmen hat.

Wedekind et al. conclude that it is not too difficult for beginners to learn the formal game of logic, but is is hard to understand it.

If one takes a look at the educational landscape in Swiss and German high schools, logic instruction is little standardized. No guidelines exist and if logic is taught at all, usually only some aspects of it are covered.

In the course of mostly optional classes in philosophy, principles such as Aristotle's *law of noncontradiction*, the *law of the excluded middle* or the concept of *natural deduction* are covered on a high level. A real dealing with propositional or predicate logic does not take place.

In math, notations, conventions as well as the symbols used and their semantics play a major role. Universal and existential quantifiers are directly borrowed from predicate logic. In calculus many formulas of the kind "$\forall \epsilon > 0 \quad \exists \delta > 0 ...$" exist. And constructs from propositional and predicate logic such as "$\forall$ lines g and h, g $\neq$ h holds: g intersects h or g and h are parallel" are used in geometry, for example.

In the resolution of the *German Kultusministerkonferenz zu den einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA)* in mathematics [53], logic is not of big importance. The word "logic" for example only appears once in this document of 50 pages, namely as one out of eleven especially suited topics for an oral exam. Terms such as "quantifier" or "predicate" do not appear in this resolution at all. Methods and techniques for mathematical proofs are covered in this document only marginally.

Traditionally class books for logic [9, 43, 67] or introductory lectures [83] to logic targeted at university level typically cover topics such as propositional logic (propositions, truth values, syntax and semantics, Boolean algebra, normal forms), predicate logic (predicates and quantifiers, syntax and semantics), resolution, and formal proofs. This approach to logic is obviously aligned with semantics of the field of logic, the contents are usually taught in a formal and abstract manner.

Concise notations and the language of formulas are a useful tool for experts. For beginners though, motivation and a link to their everyday life experience get lost. This might be one of the reasons, why logic was

largely removed from curricula together with set theory in the 1970s. Together with set theory, logic was reduced to an abstract and artificial entity. Many students were able to draw Venn diagrams, connect elements of different sets with arrows, but they did not gain any deeper insight.

If logic should become an important part of general education, one needs to find an approach that better accommodates the students prerequisites. Instruction is especially effective, if learners can create a reasonable link between the item to learn and their everyday life, their own sphere of experience. For an instruction to logic, this asks for an approach from close-by examples to the general theory.

Logic obeys strict formal rules. Thus a computer-aided learning environment suits well for an introduction to logic. We introduce the new learning environment LogicTraffic which illustrates basic concepts of logic with the help of traffic intersection control, thus building on experience of learners.

3.3 The Program LogicTraffic

The main idea behind LogicTraffic is to find a formula in *propositional logic* (PL) that makes a given intersection "safe", i.e., which prevents collisions through appropriate signaling on the corresponding traffic lights. With each lane corresponding to a variable, and *true* and *false* indicating a green and red light respectively, a "safe" formula is one that avoids any two crossing lanes to simultaneously have green traffic lights. Orange lights are omitted for simplicity and because LogicTraffic is only concerned about safety in a static traffic light state. In the static traffic world of LogicTraffic only the two states *traffic allowed to run* and *traffic not allowed to run* exist, there is no state in between. Safety in transitions (i.e., the dynamic and more realistic case with some time in-between changes of traffic light) are not considered, and for the simulations in the test mode, a fixed red time in-between is used in the case of a traffic light change. The exact semantic interpretation of an orange traffic light is not clear anyway, and propositions in PL are by definition entities that can clearly be assigned one of the two possible truth values *true* or *false*.

The user interface of LogicTraffic [4, 7] consists of five parts (see figure 3.1). The central part shows a particular *traffic situation*, representing an intersection with a number of intersecting lanes and their

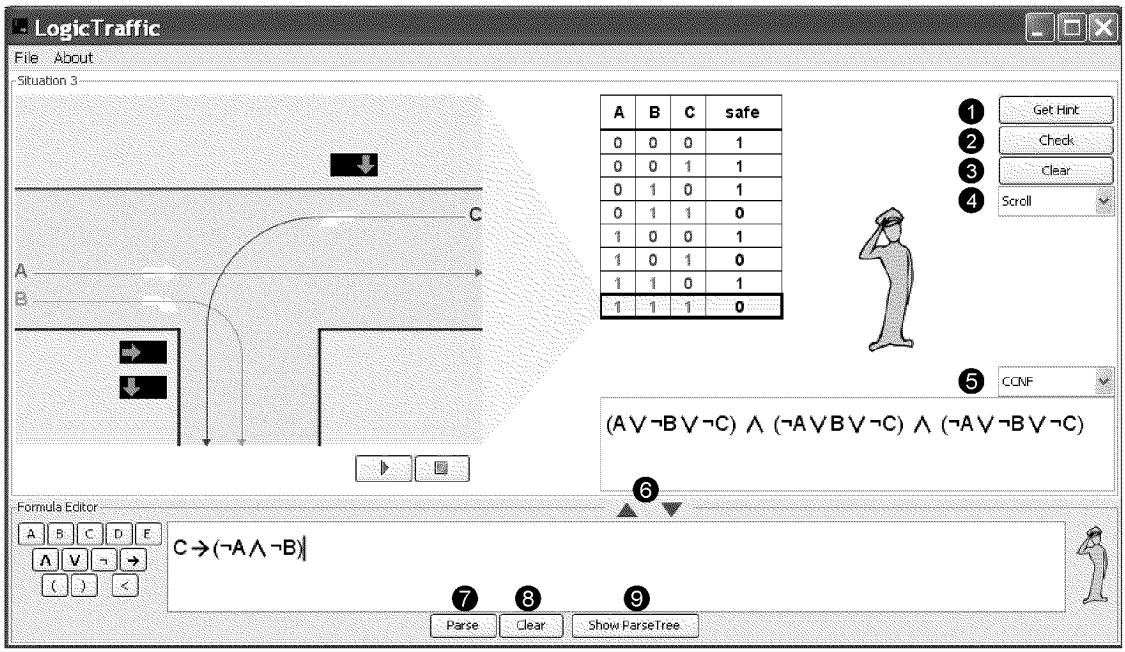

Figure 3.1: The interactive learning environment LogicTraffic

corresponding traffic lights. A *truth table* to the right shows all possible traffic light configurations for the intersection, allowing students to manually designate certain configurations as "safe" or "unsafe". A configuration in LogicTraffic stands for one specific setting of the traffic lights, i.e., each row in the truth table represents one possible configuration. Below the truth table, a *corresponding formula* in PL is shown that summarizes the truth table above, according to whatever settings of "safe" and "unsafe" values the student entered. Instead of manually filling out the truth table, students can also use the *formula editor* at the bottom to directly enter a PL formula and initialize the truth table accordingly. Last not least, a set of buttons on the right for example allows students to simulate each line of the truth table in the traffic situation window (figure 3.2), thus gaining a first-hand understanding of potentially conflicting traffic light configurations.

The semantics of the five buttons on the right of the truth table as numbered in figure 3.1 are as follows.

Get Hint (1): Provides hints about unsafe and suboptimal rows in the truth table, i.e., which row is not safe or not optimal yet, if any.

Check (2): Checks for the safety of the actual truth table. Each row marked safe (i.e., a 1 in the safe column) is simulated for a certain

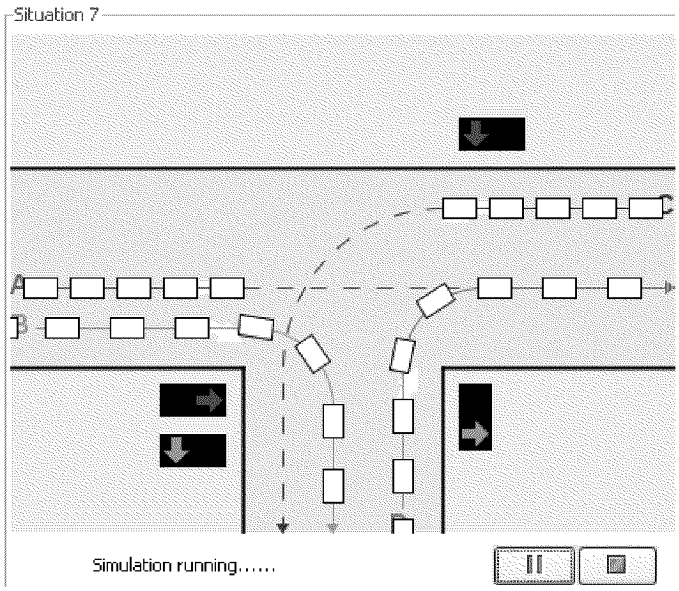

Figure 3.2: Traffic animation in LogicTraffic, with the control buttons for start/pause and stop

period of time to check whether a crash occurs or not.

Clear (3): Clears the truth table and accordingly the corresponding formula.

Scroll/Shrink (4): Changes the display mode of the truth table between the modes scroll (show truth table with fixed row height and scroll bar if needed) and shrink (show the whole table and shrink/stretch if needed).

Formula form chooser (5): Sets the desired form for the automatically generated formula corresponding to the actual truth table.

As stated above, LogicTraffic automatically generates and updates the formula to the actual truth table. There are six different forms for these formulas available, see picture 3.3:

DNF: disjunctive normal form (optimized).

CNF: conjunctive normal form (optimized).

CDNF: canonical disjunctive normal form (minterms exclusively).

CCNF: canonical conjunctive normal form (maxterms exclusively).

Implication: an optimized formula, including the implication operator, if possible.

Simplest: optimized formula.

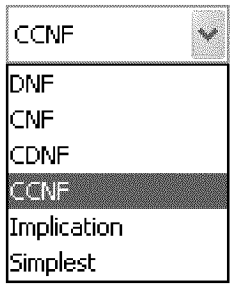

Figure 3.3: The six forms for automatically generated formulas in LogicTraffic

All optimized formulas are generated with the Quine-McCluskey [46] algorithm, which is functionally identical to Karnaugh mapping, but more efficient for use in computer algorithms. Optimization happens with respect to the number of variables and operators appearing in the formula, i.e., a formula is better the smaller their weighted sum is. See appendix B.3 for details on the algorithms for the optimized *Implication* and *Simplest* forms.

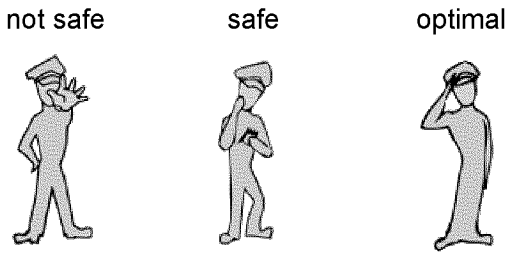

Figure 3.4: Icons for the three states *not safe*, *safe* and *optimal* in LogicTraffic

Formulas in LogicTraffic are *not safe*, *safe* or *optimal* with respect to the given situation. To represent these states, three different icons of a policeman are used, see figure 3.4. *Not safe* means that collisions are possible (i.e., there is at least one "unsafe" row with a 1 in the safe column). In *safe* state, no collisions are possible, but there are more safe configurations (i.e., there is at least one "safe" row having a 0 in the safe column). And *optimal* finally stands for a safe state with all safe configurations set as safe (i.e., all "unsafe" rows have a 0 in the safe column and all "safe" rows have a 1 there).

As stated above, the formula editor of LogicTraffic lets the user enter and edit formulas. Editing happens either with the buttons on the left of the formula editor panel or through keyboard input. The semantics

of the other four buttons is the following, see figure 3.1 for the number references.

Buttons △ and ▽ (6): button △ adjusts the truth table according to the formula in the formula editor and button ▽ vice versa.

Parse (7): parses the formula and updates the safety status indication (policeman) to the right of the formula editor accordingly.

Clear (8): clears the formula.

Show ParseTree (9): displays the parse tree of the current formula in a different window, see figure 3.5.

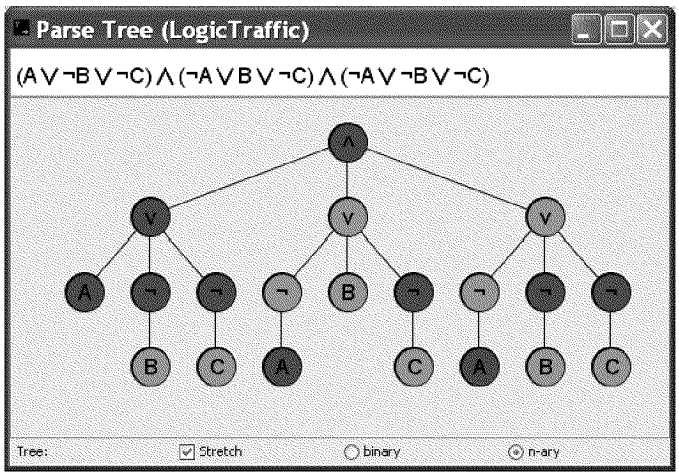

Figure 3.5: An example parse tree in n-ary form as visualized by LogicTraffic

LogicTraffic offers two different size modes for parse trees: *normal* renders the tree in fixed size and *stretch* adapts the size of the tree to the current window size. LogicTraffic further allows two kinds of trees, namely *binary* and *n-ary* parse trees. N-ary trees are especially useful to visualize the structure of formulas in CNF and DNF. Figure 3.5 shows a parse tree in n-ary form in stretch size mode.

3.4 Learning Goals and Use of LogicTraffic

The learning goals of LogicTraffic include the basics of PL, and cover concepts such as variables, truth values, logical operators, formulas, equivalence of formulas, and normal forms. By using the intersection/traffic light metaphor, students can immediately connect the concept of PL to real-world situations, and animated collisions allow direct feedback whenever conflicting assignments are made.

LogicTraffic allows a stepwise introduction to the basic concepts of PL. In a first step, students can control the traffic lights manually and gradually fill the truth table. For each possible configuration of traffic lights, the student declares whether the resulting traffic control is safe or not. Using the simulation allows a visual validation of the configuration set by the user.

LogicTraffic comes with a set of intersections of growing complexity, which can be loaded and solved in succession. While a simple intersection with two lanes can still be configured in the manual fashion described above, students quickly realize that increasingly complex situations lead to unmanageably large truth tables, hence a transition to equivalent PL formulas is motivated. Consequently in the next step, students try to control a given intersection only with the help of a PL formula, without using the truth table. In a last step, students try to find more compact descriptions of those formulas that describe safe intersection. The controls allow students to transform a given formula into different representation, e.g., into canonical disjunctive normal form, which in turn leads to the investigation of the relationship between normal forms and the truth table. A final step thus might involve devising an algorithm that allows for the construction of a formula in CDNF, given a corresponding truth table.

After having introduced the basic concepts of PL, further exercises with LogicTraffic could lead to concepts such as tautology or even more advanced concepts such as satisfiability, efficient testing of equivalence, and an introduction to complexity theory.

Example Traffic Situations and Exercises

All the teaching materials accompanying LogicTraffic are found online under [3]. To give an impression of the different levels of difficulty of exercises possible with LogicTraffic, we show an example with two different traffic situations, one with two and the other with five lanes. Figure 3.6 shows the given traffic situations (a), the safe truth table (b), a formula as it might be found by students (c), and finally a formula generated by LogicTraffic (here in *Implication* form and in *DNF*, respectively).

Safe formulas can be interpreted in natural language. The safe formula $C \rightarrow (\neg A \wedge \neg B)$ for the situation shown in figure 3.1 can for example be read as "If C runs, then A and B are not allowed to run."

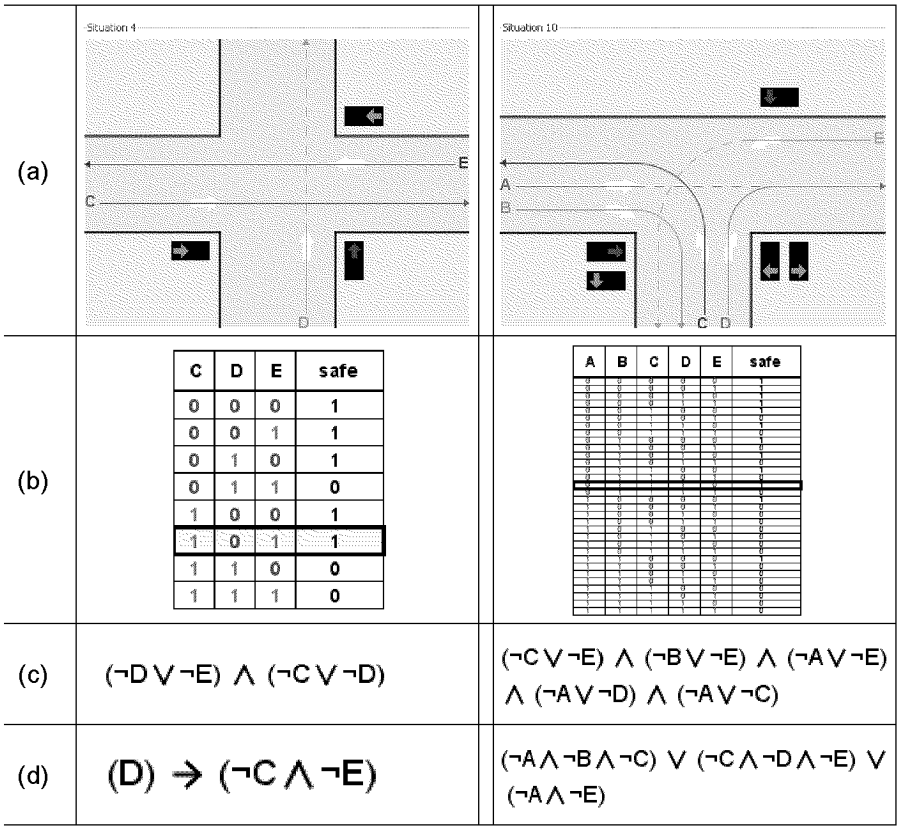

Figure 3.6: Example situations in LogicTraffic

This offers a deeper understanding of formulas and illustrates the link between formulas and natural languages and vice versa.

One interesting exercise is to find an algorithm to directly create safe formulas for a given traffic situation. There are several solutions; as an example we here present an algorithm in pseudo code, producing safe formulas in CNF:

```
cSet = empty set of clauses
foreach(collision point of any two lanes X and Y)
  cSet.add("(¬X ∧ ¬X)")
build one big conjunction over all clauses in cSet
```

Another algorithm for example involves considering each lane (e.g., X), the lanes (e.g., Y and Z) intersecting with this one and creating clauses of the form "$(X \rightarrow (\neg Y \wedge \neg Z))$". Ideas for other algorithms are mentioned in the accompanying teaching materials.

An exercise could also be to draw a possible traffic situation to a given formula; this is an ambitious task. For this type of exercises an editor for intersections might be desirable. We abandoned such an

editor deliberately in order to keep the learning environment for the user as simple as possible.

3.5 Related Work

Of course LogicTraffic is not the only and first software concerned with an introduction to logic. The following section shortly reviews three exemplary programs. The first is one of the classics in the field. The second is an up-to-date approach to logics at university level. The third is a representative online applet. To round off this section, we report on how safety at intersections with traffic lights is guaranteed in reality. We present the approach of the city of Zurich and compare it to the approach of LogicTraffic.

Tarski's World

The program Tarski's World [9] offers an introduction to predicate logic and lets learners build two- and three-dimensional worlds with simple geometric objects, see picture 3.7. The objects are ordered on a chess board like field. They are of different shapes such as cube or pyramid and differ in attributes such as size or color. The learner can formulate sentences in predicate logic and the system will check and indicate whether these apply in the given situation or not.

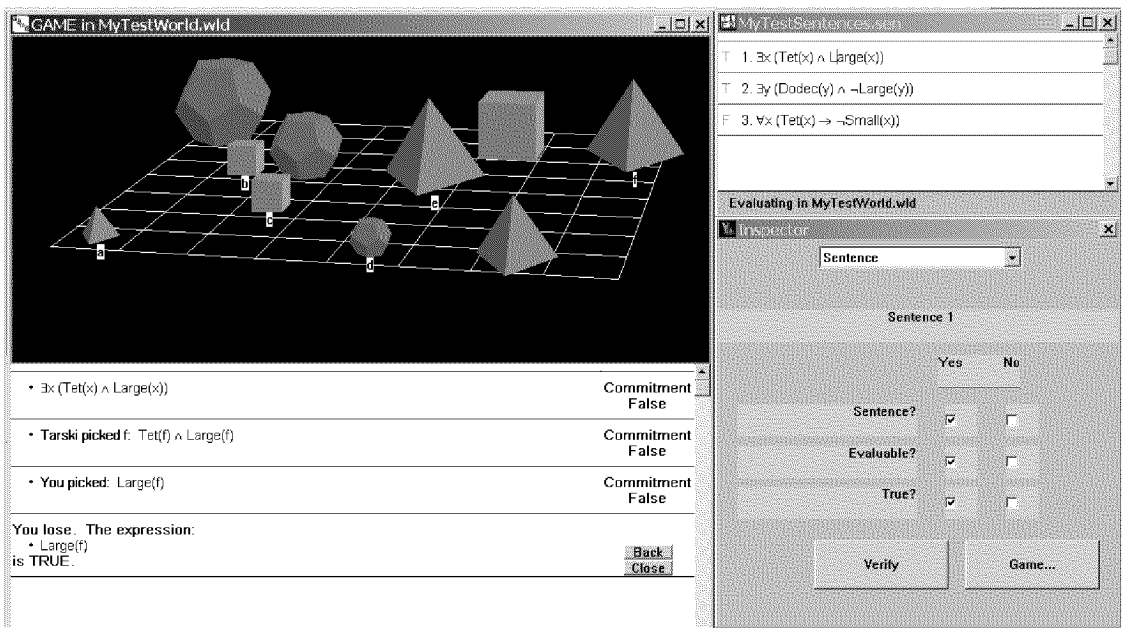

Figure 3.7: Predicate logic in Tarski's World

Formulas in predicate logic like the following can be created:

- For all objects holds: If an object is a tetrahedron, then it is not large: $\forall x(Tet(x) \rightarrow \neg Large(x))$
- There exists a small object a, a medium sized object b and between a and b there is an object c: $\exists a \exists b \exists c(Small(a) \wedge Medium(b) \rightarrow Between(c, a, b))$

With a question-answer game Tarski's world offers the possibility to test the truth value of particular sentences in predicate logic. An example (see figure 3.7): Given is the sentence $\exists x(Tet(x) \wedge Large(x))$. In a first step we declare this sentence to be wrong. Then the system asks us, whether we are convinced that each object falsifies $(Tet(x) \wedge Large(x))$. We affirm. Upon that the system chooses a concrete object f and asks us, whether $(Tet(c) \wedge Large(c))$ is wrong. We affirm our belief that $Tet(f)$ oder $Large(f)$ is false and choose $Large(f)$ as false expression. In the end, the systems tells us that we have lost, since the expression $Large(f)$ is true.

Tarski's world has an appealing, visual user interface and illustrates with few predicates an application of predicate logic. The world is simple and the possibilities for interactions are convincing. But Tarski's world is not able to create a real link to everyday life, the representation is limited to an iconic and symbolic view.

Propositional Logic (PL) in the Virtual Logic Laboratory

ViLoLa (Virtual Logic Laboratory) [79] is a collection of ten logic-oriented modules for logic education at university level. The modules range from basics to advanced and are parted into *mathematical logic*, *logic in computer science* and *philosophical logic*. They cover topics such as *introduction to predicate logic*, *computability and complexity* or *logic and argumentation*.

As an example we take a closer look at the module *classical propositional logic* (CPL). This module primarily consists of a text document of 131 pages, giving a compact and formal introduction. The scriptum is on the one hand supplemented by multiple-choice questions to selected chapters. These questions can be answered and evaluated online. On the other hand, there are many exercises referring to the software logics workbench [78] by the same authors. This program allows manipulation of formulas in PL in a command-line style; there are

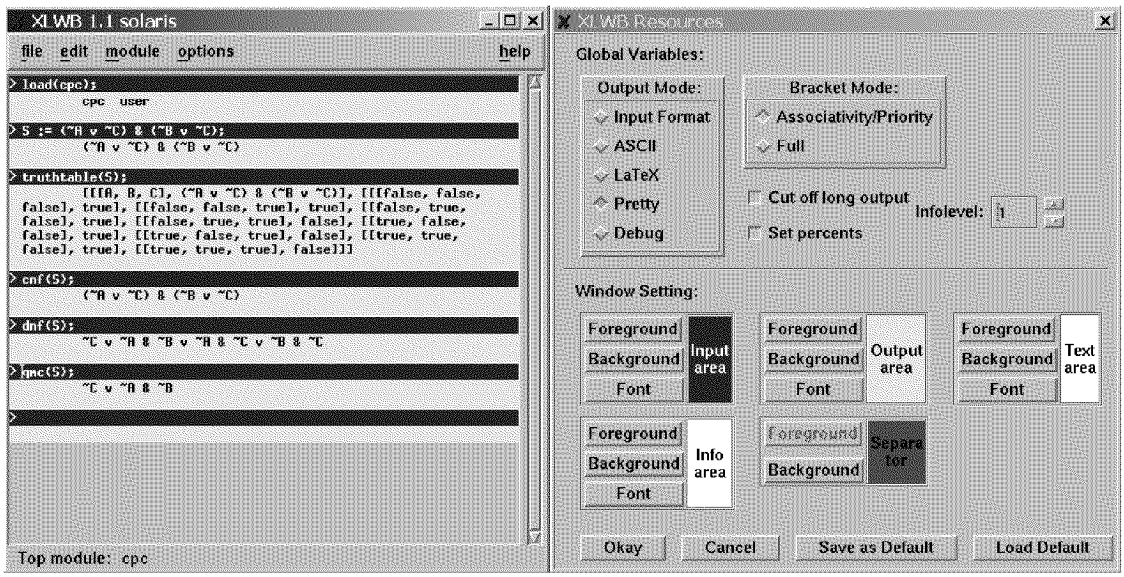

Figure 3.8: Screenshot of an example session with the Logics Workbench

for example functions to test for satisfiability or to transform formulas. Figure 3.8 shows a session with the display of the truth table of a formula and transformations to equivalent formulas. Note for example the not very appealing textual display of a truth table in figure 3.8.

This module offers thematically a broad formal introduction to PL. The direct links from the scriptum are useful, the user is for example directly forwarded to a multiple-choice question or to the logics workbench with a single mouse click. The former allows testing of the learners understanding through given test questions at the end of a chapter. With the help of the logics workbench, exercises can be solved and evaluated directly on the computer. The software logics workbench is quite spartan and can only be controlled via command line, see picture 3.8. Overall the module is very formal and abstract.

Truth Table Constructor, an online applet

Many online tools to visualize simple formulas in PL or to display truth tables exist. As a representative tool, the Truth Table Constructor [20] is a freely available Java applet displaying truth tables to a given formula in PL, see figure 3.9. This simple program lets the user enter a formula in ASCII-encoding. It then displays the corresponding truth table if the formula is well formed. In case of a parse error the cursor prompts at the position, where the error occurred. The truth table is filled by the program at once or step-wise per row or column at the

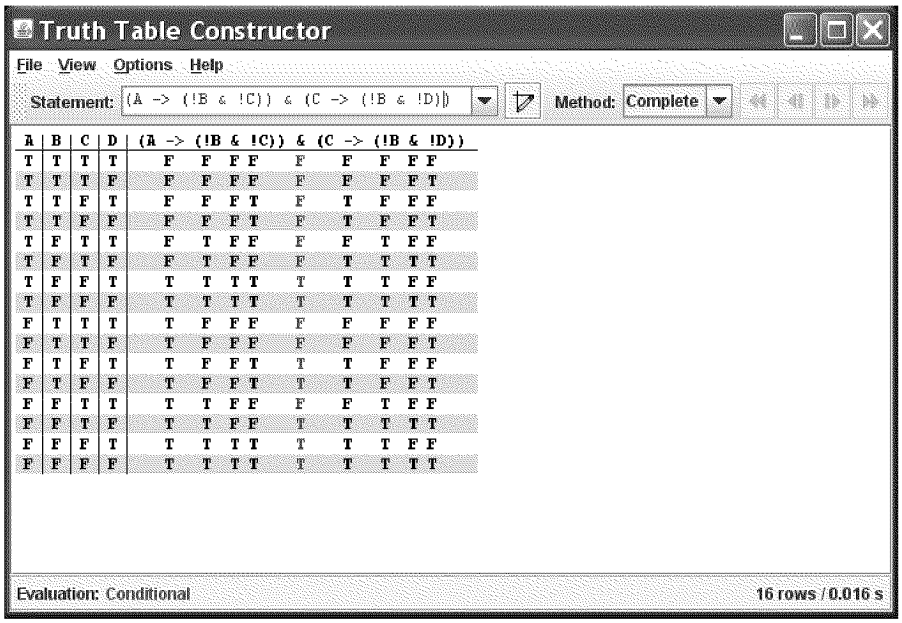

Figure 3.9: Screenshot of the Truth Table Constructor

user's desired speed. Truth tables can be saved as text or exported as images. There is no further functionality such as formula simplification or creation of normal forms. The Truth Table Constructor is a simple online application with quite limited functionality.

LogicTraffic and the Reality

To motivate the idea behind LogicTraffic and see how safety at intersections is guaranteed in reality, we contacted the responsible people in the department for traffic control of the city police of Zurich [30].

The city police of Zurich has sketches of every intersection with traffic lights in the city, and these sketches are similar to the situations depicted in LogicTraffic. As shown in figure 3.10, in a sketch of one specific intersection, all lanes are numbered. And in a corresponding matrix, the minimal times in-between a traffic light change are given for any pair of intersecting lanes. If two lanes do not intersect, there are no minimal times in the corresponding fields. This matrix contains the same (and more) information as the truth table in LogicTraffic.

A last detail to mention is the fact that the information of this matrix is cast into hardware and locally present at each intersection. This fail-over mechanism prevents locally unsafe states of the globally run traffic control system, i.e., if the global control assigns green light to two intersecting lanes, the whole intersection will immediately switch

to an orange traffic light blink mode.

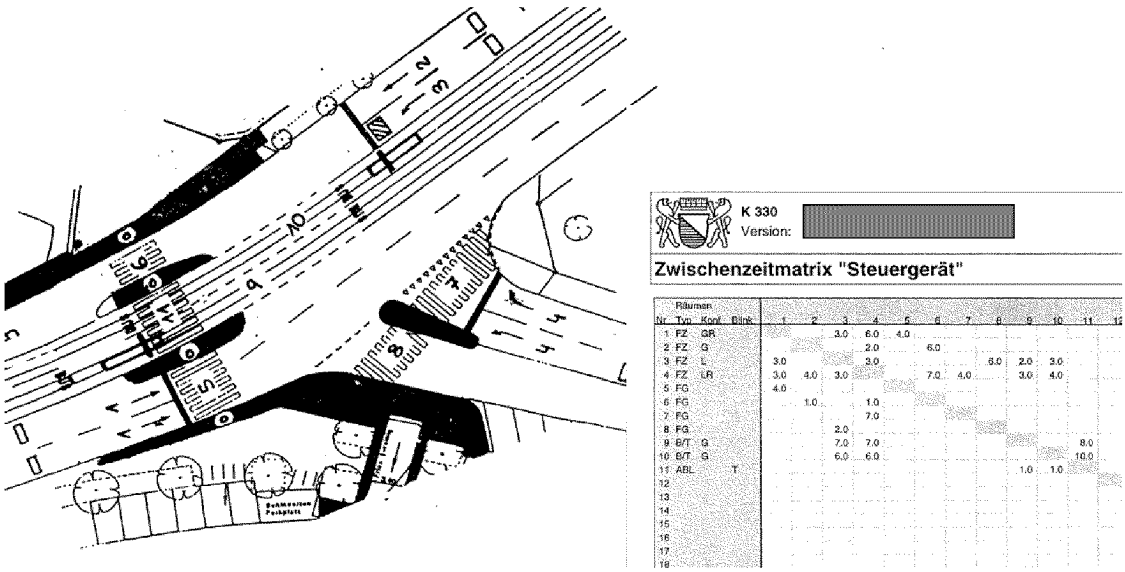

Figure 3.10: Sketch of a real intersection and data sheet "Zwischenzeitmatrix"

LogicTraffic obviously represents a model of reality, it is just a little simpler and omits details. This is a typical approach in teaching: Through abstraction, reality is presented in models that show on the one hand the fundamental concepts beyond real-world phenomena and that on the other hand are simple enough to be understood by students. Other well-known examples of this approach include typical experiments in chemistry or physics school laboratories. In that sense, LogicTraffic is a virtual laboratory for students to explore propositional logic embedded in a real-world example.

3.6 Conclusions

LogicTraffic is a new learning environment for an introduction to PL presenting an application of PL in the real-world scenario of traffic control. The software is highly interactive and allows to tackle problems of different levels of difficulty.

The three related software examples in the previous section illustrate the strengths of computer-aided learning environments like automatic transformation, calculation and checking of formulas and graphical representation of certain situations and states. LogicTraffic makes use of these advantages too. The three related programs show also possible stumbling blocks for learning environments for an introduction to

an abstract topic such as logic: All three environments are relatively formal, dry and clinic.

LogicTraffic is based on a realistic scenario as shown by a comparison to the approach taken by the city of Zurich.

LogicTraffic is to the best of our knowledge the first interactive learning environment targeted at PL in a reality-near way. Its real-world-based approach with virtual-enactive representations offers a new access to logic and is a new chance for teaching logic in general education.

4 QueueTraffic: Traffic Jam and Queuing Theory

Traffic jams are a well-known phenomenon and queuing theory is a mathematical tool to model and analyze traffic jams. This chapter introduces the interactive learning environment QueueTraffic as the second component of InfoTraffic. QueueTraffic offers the functionality to control, simulate, and analyze traffic intersections and thus gives a gentle introduction to the theory of queues.

4.1 The Importance of Waiting Queues

Even though queuing theory is far from being noteworthy knowledge among the general population, queues are generally well known: We have to wait in queues quite everywhere we go and whatever we do, be it on the road, in supermarkets, at the post office, or at our doctor's. How do queues work and why do they occur? Or maybe more importantly: How can they be avoided? While not everybody might care for answers to these questions, some people are faced with them every day: the manager of a supermarket deciding how many cash registers to open on which day and daytime; the hospital staff trying to provide expedient treatment at peak times with fewer doctors and nurses; or airport control faced with bad weather and the start of travel season. Mathematical analyses of the above stated real problems typically involve modeling the problems using queuing theory.

Besides the connection to everyday life described above, queuing theory is important in scientific disciplines that require some form of dynamic capacity planning, prediction, or analysis, e.g., in any kind of logistics or communications (e.g., telephone, packet routing in the Internet, printers). While an in-depth understanding of queues requires a substantial amount of higher mathematics (e.g., calculus and probability theory), an intuitive understanding and approach might be sufficient

in many cases, e.g., for directing traffic around an accident, or when staffing point-of-sales during special sales. Having a general grasp of the underlying problem can ease further comprehension of more advanced models and levels of abstraction.

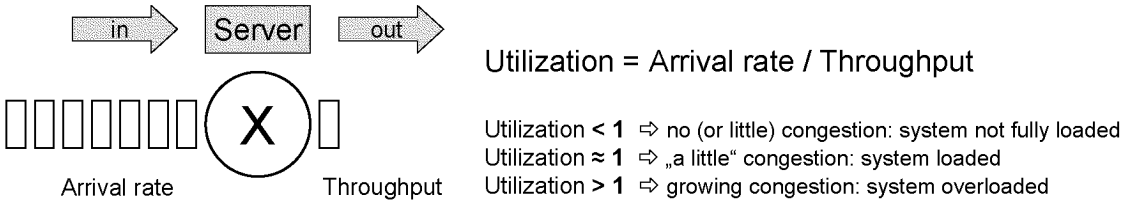

Figure 4.1: A typical sketch of a queuing model

In computer science, queuing systems [80] are typically drafted as illustrated in figure 4.1. The circle with the X stands for example for a web server. We see requests entering the system from the left at a certain arrival rate. The throughput corresponds to what is possibly coming out of the system on the right. The utilization finally is the ratio of arrival rate and throughput, its interpretation in terms of whether the system is likely to be congested or not is given at the right bottom of the figure.

This abstract model does not change, be the system a printer managing print jobs, a telephone center handling calls, or a supermarket cash register serving its clients. Queuing systems are reduced to one or several servers with respective arrival and service processes. This abstraction is exactly the idea behind the standard mathematical notation of Kendall for modeling queuing systems. In Kendall's notation $X/Y/m$ for queuing systems, X describes the arrival process, Y the service time distribution, and m the number of identical servers, see for example [66]. Standard notations to describe X and Y are M for memoryless (e.g., Poisson process or exponential distribution), D for deterministic or G for general. So figure 4.1 displaying one server shows for example a M/M/1 queuing system in Kendall's notation if the arrival process is a Poisson process and the service time follows an exponential distribution.

4.2 Queuing Theory and Simulation

Additionally to focusing on applications in everyday life for introducing queuing theory, simulation can help too. Simulation as a virtual-

enactive representation visualizes the behavior of different settings in a system. Of course, simulation is not new in computer based instruction. New in QueueTraffic is the simultaneous view of a real intersection situation with simulated cars and statistical data and charts. These simultaneous views force an understanding of the values of the abstract key parameters (e.g., arrival rate, throughput) in a queuing model. In this way, students can relate to their prior experience with traffic jams and relate it to new concepts such as arrival rate or throughput.

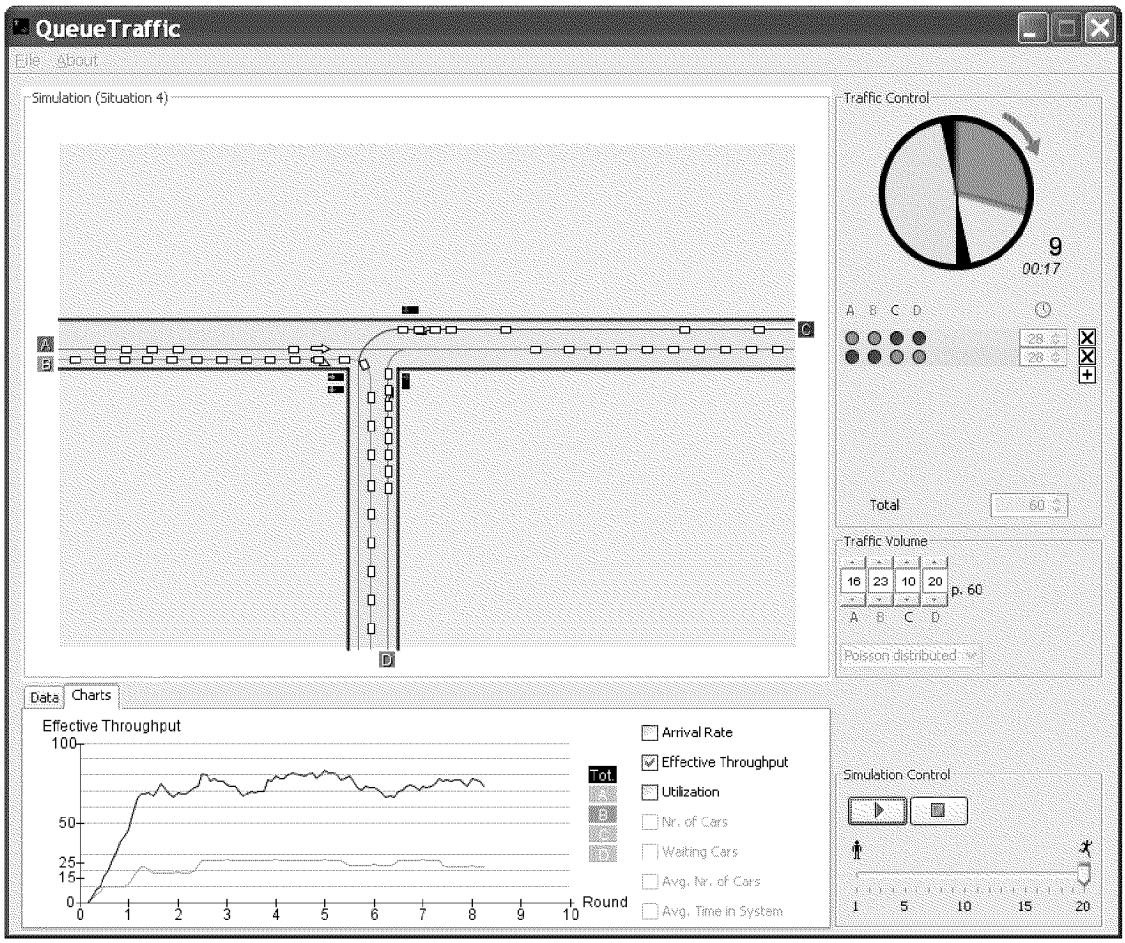

Figure 4.2: The interactive learning environment QueueTraffic

4.3 The Program QueueTraffic

QueueTraffic offers an introduction to queuing theory by letting students simulate and analyze traffic at an intersection. Important system parameters can easily be changed.

The user interface of QueueTraffic consists of five parts (see figure 4.2). The central part of the window is as in LogicTraffic taken up

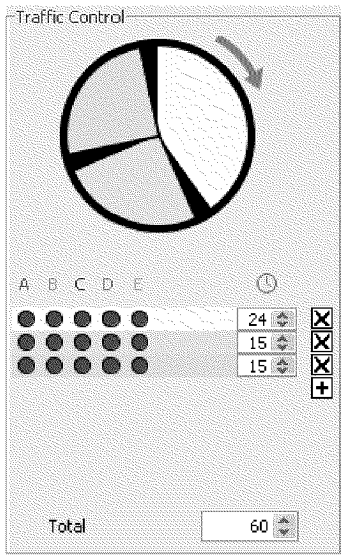

Figure 4.3: QueueTraffic: the panel for traffic control

by a view of the *traffic situation*, representing an intersection with its lanes and traffic lights. The *traffic control* panel (figure 4.3) at the right top allows students to define phases, i.e., time slots when certain lanes show a green light. Each phase is represented by a combination of red and green signal settings and a duration. All phases together make up a single round of the simulation. The circular diagram in the top right shows the relative length of each phase within a round. The example in figure 4.3 shows a rounds of 60 seconds consisting of three phases, one of 24 seconds and two of 15 seconds. Note that QueueTraffic automatically adds a fixed two second change time between any two directly consecutive phases.

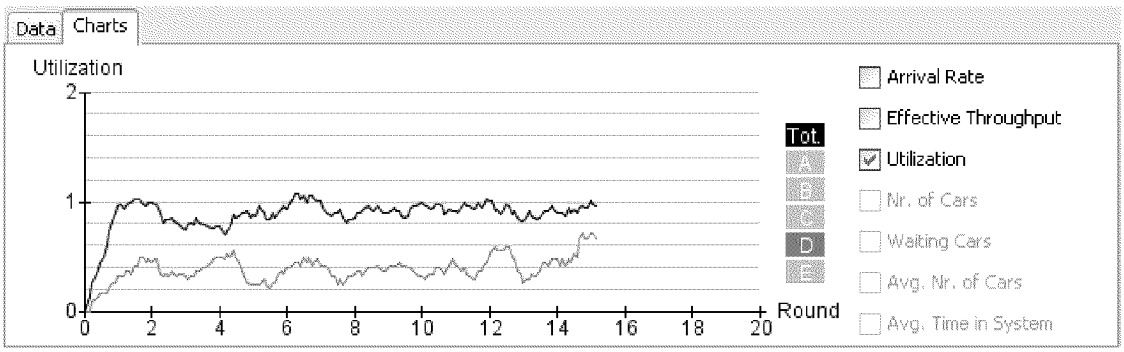

Figure 4.4: Charts in QueueTraffic

In the *traffic volume* panel, students can define the arrival rate of cars for each of the lanes and whether the cars arrive Poisson dis-

tributed or with constant time intervals. Two buttons and a slider in the *simulation control* panel at the bottom right of figure 4.2 allow students to start and stop the simulation as well as to control its execution speed. The *data and chart* tabs at the bottom left of figure 4.2 display statistic information about the current system settings in numeric and graphic format. During simulation, these figures are constantly updated. The user can choose which parameters (e.g., arrival rate, throughput, utilization, number of waiting cars) he wants to have displayed for which lanes. Additionally, parameters accumulated over all lanes are available. Figure 4.4 shows an example chart displaying the total utilization (averaged over all lanes) as well as the utilization of lane D.

4.4 Learning Goals and Use of QueueTraffic

The learning goals of QueueTraffic include the basics of queuing theory [13, 38, 66] and cover concepts such as throughput, arrival rate, average waiting time, and Poisson distribution. The main idea is that students gain an intuitive understanding of queuing theory through experimentation with a range of different intersections, by changing parameters like the arrival rate of cars for individual streets and the timing of the individual traffic lights, and observing the resulting system behavior. As an introduction, students start with a simple situation with only one lane A. They are asked to define two phases of the same length, with lane A having a green traffic light in one phase and red in the other one. The students run the simulation, and observe the resulting effects. They can reason about where in reality such situations happen (e.g., pedestrian crossing or construction work) and get to know how the program works from this simple example. They can see how cars appear in the system and leave again, and whether queues are built up or not. Students further get a qualitative impression and feeling of Poisson distribution and a constant interval time as well as the differences between the two.

It is worth pointing out that the situation described above with only one lane corresponds exactly to the standard queuing model with one server. This simplest model is often used to model a waiting system at one pay desk in a supermarket, at a printer, in a call center with one phone, at a doctor's waiting room, or at a post-office counter. If we assume the arrival process to be a Poisson (i.e., random) process and

the service time to have an exponential distribution, this corresponds exactly to a M/M/1 system in Kendall's notation as introduced in section 4.1. Note that in QueueTraffic, with the arrival rate, one key parameter can be directly entered in the traffic volume panel. The other key parameter, the service time, can only be set indirectly as it depends on the ratio between the lane's green time per round and the length of one round.

In a next step, students are taught basic concepts of queuing theory, such as arrival rate and throughput, and they are asked to analyze the given situation and parameters accordingly to these concepts. Students can on the one hand run simulations in QueueTraffic and use the situation visualization and the statistics and charts. On the other hand, they can do calculations based on the actual parameter settings of the simulation and for example check, how close the effective throughput simulated by QueueTraffic and the theoretical throughput – calculated based on the corresponding simulation settings – are.

Finally, students can try to configure a given system such that no traffic jam occurs, or so that waiting times are minimized. As in the LogicTraffic application, different predefined intersections can be loaded in order to gradually increase the difficulty level throughout the exercises. Detailed problems with sample solutions can be found online under [3], where among other teaching materials also slides for an introductory presentation to QueueTraffic are available.

4.5 Related Work

Of course other software for traffic control and analysis exists, typically with different purposes though. Before we take a look at some representative tools, we point out that there is an important distinction between two main models. Models in the area of traffic simulation and analysis are generally divided in macroscopic and microscopic traffic models [50]. Macroscopic traffic models on the one hand consider parameters summed or averaged over a certain period of time, i.e., they do not take each individual object (e.g., car) into account. Microscopic traffic models on the other hand resolve traffic processes into movements of individual vehicle. QueueTraffic provides a microscopic traffic simulation.

Professional Tools

There are many professional and commercial tools for traffic simulation and analysis available for both macroscopic and microscopic traffic simulation. To give an impression of tools using macroscopic simulation, we mention the popular programs Ampel and Knobel by the company BPS GmbH [73]. The institute for transport planning and systems at ETH Zurich uses this software. Ampel is a tool for traffic planning at traffic light facilities and Knobel for planning intersections without traffic lights. Figure 4.5 shows a screenshot of Knobel, visualizing traffic flow at an intersection.

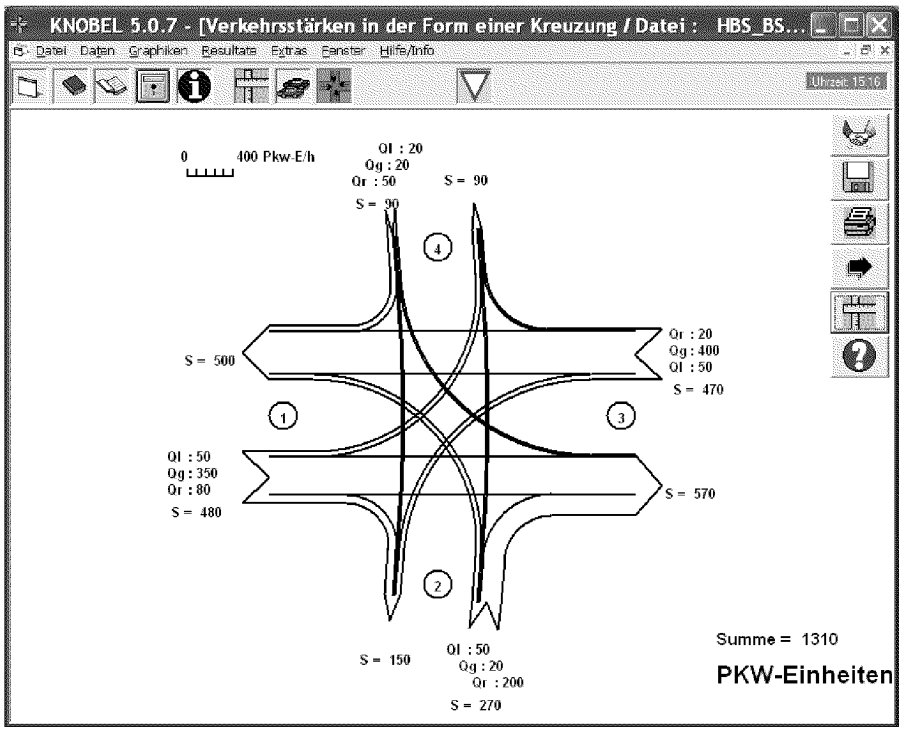

Figure 4.5: Screenshot of Knobel

VISSIM [60] on the other hand claims to be the leading microscopic simulation tool for modeling multimodal traffic processes and being used in more than 70 countries worldwide. This comprehensive professional tool (see figure 4.6) offers realistic traffic models, 3D animation and besides every kind of motorized traffic objects for example also takes pedestrian and cyclists into account.

These two tools are not suitable for use in general education. They are powerful but are not designed for educational purposes:

High Complexity: The two programs offer a lot of functionality with detailed settings and adjustments. The programs are simply too

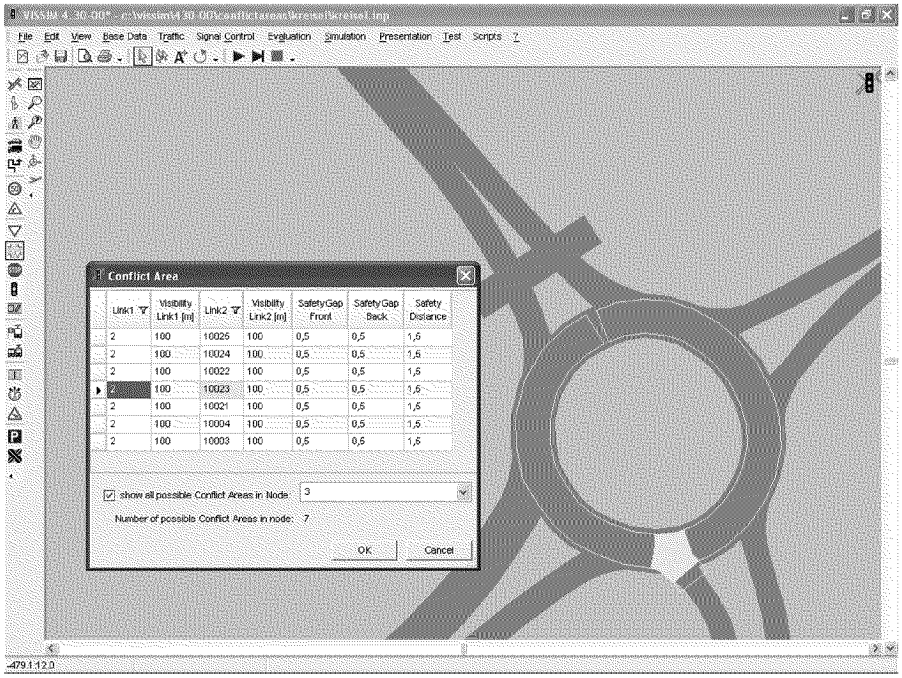

Figure 4.6: Screenshot of Vissim

complex for educational use. Software for educational purpose should be reduced to the basics. Or as Reichert [61] puts it: "In education, we should strive for utmost conceptual simplicity."

Difficult Familiarization: As a direct consequence of the high complexity, these programs require a lot of time to get used to. This is in general undesirable in education.

Different Target User: These tools are built for professionals and require knowledge accordingly. They are not intended to be used by students. As an analogy, students in chemistry are not expected to be able to operate an industrial laboratory, but to work with a school laboratory.

Free Online Programs

There are a number of traffic simulation programs freely available online, most of them are applets. We list two representative examples here.

As can be seen in figure 4.7, the microscopic simulation applet from [75] simulates a street with two lanes, it offers different situations like lane-merging, obstacles or ringroads. The user can set parameters such

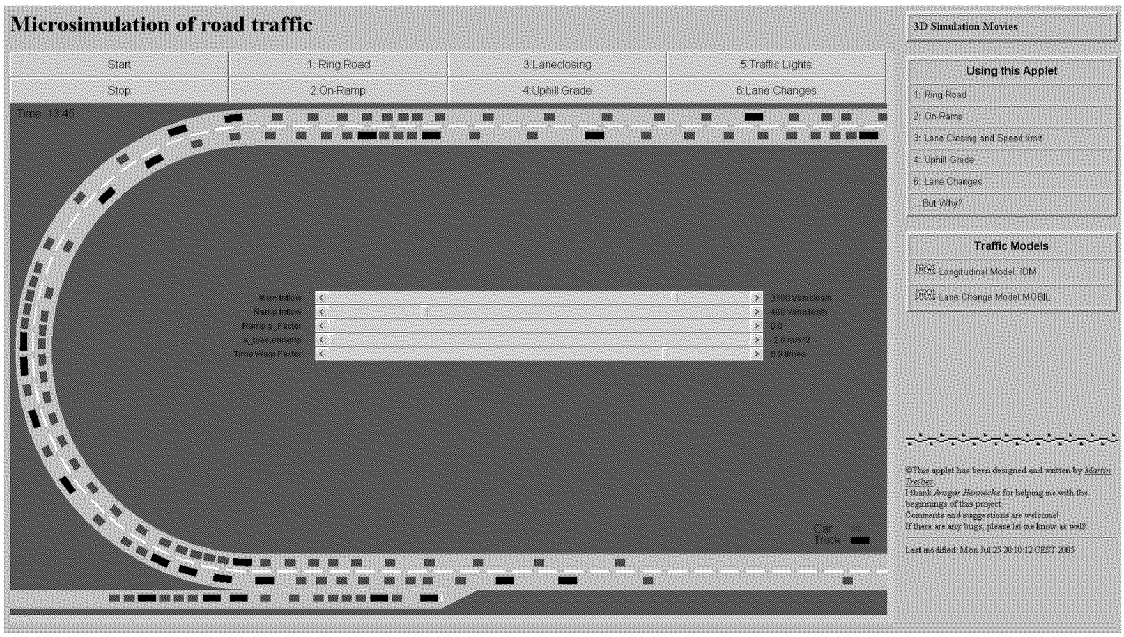

Figure 4.7: Screenshot of the microsimulation of road traffic applet

as the inflow (arrival rate) of vehicles or the ratio between cars and trucks.

A second example is the applet from [65] which has two windows, one displaying the current simulation and the other for controlling the simulation, see figure 4.8. Parameters such as the number of cars per lane or the duration of green phases can be altered. The program offers as well a rudimentary traffic statistic.

While these two programs offer a visualization of traffic and animate individual cars, they both lack meaningful statistical data about the simulation. They have no corresponding views where the student can observe what is going on in the lanes and with the values of the key parameters of the simulation. Compared to QueueTraffic, these two programs offer no selective display of important values of the current simulation. They therefore miss to relate the simulations to the important parameters of queuing theory.

4.6 Conclusions

QueueTraffic is an interactive learning environment offering an intuitive way to modify the important parameters of a simple traffic simulation. It displays all the relevant information about the current state of the intersection and visualizes the data both statistically and pictorially.

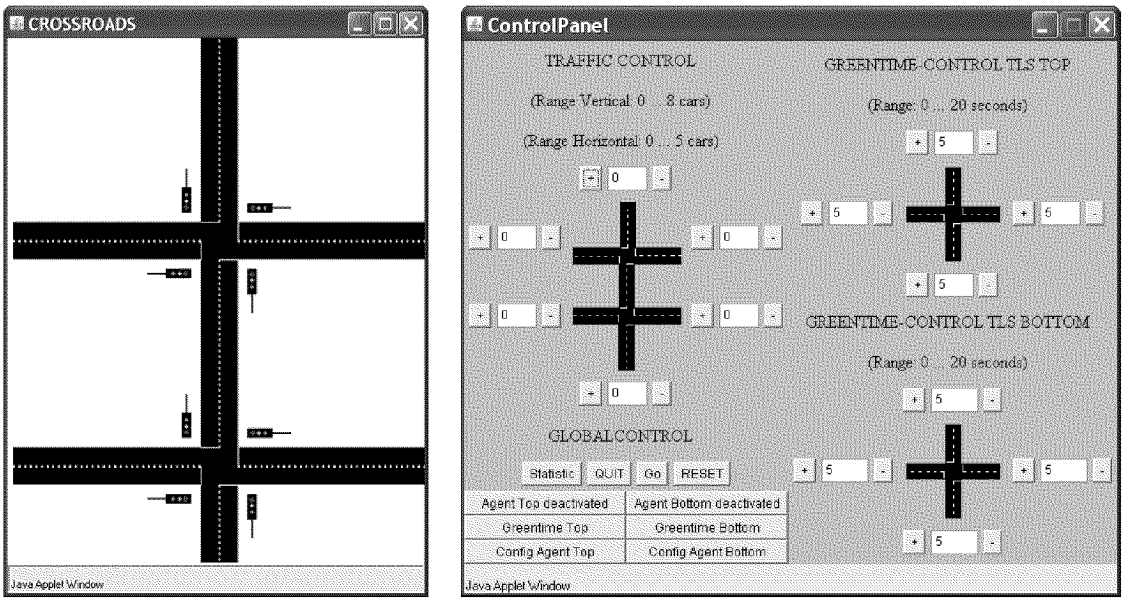

Figure 4.8: Screenshot of the crossroad simple traffic simulation applet

QueueTraffic thus allows a gentle introduction to queuing theory.

As shown in the previous section, QueueTraffic is not the first and only software system dealing with traffic simulation and the analysis of queues. Professional tools have a different goal than QueueTraffic. They are intended for analyzing real and complex situations and therefore have to offer a broad functionality, allow a lot of different settings, and obey professional standards. Professional tools are not suitable for educational purposes. Furthermore quite a number of applets are available online targeted at traffic simulation and the analysis of queues. For educational purposes they lack a display of important key parameters and relations between them and do not offer corresponding views. QueueTraffic in contrast displays different views of important key parameters of queuing theory and they are constantly updated during simulation.

To conclude, QueueTraffic closes the gap between complex professional tools and simple online applets. We hope that this attractive simulation tool with its accompanying teaching materials becomes increasingly used in classrooms as an introduction to queuing theory.

5 DynaTraffic: Markov Chains and Analysis of Dynamic Systems

The third component of InfoTraffic after propositional logic and queuing theory addresses dynamic systems. Dynamic systems based on Markov chains are a topic that involves more mathematical background than the first two topics dealt with in InfoTraffic. This chapter introduces the interactive learning environment DynaTraffic to analyze the traffic distribution over time in a system with several intersections and thus illustrating an exemplary use of Markov chains to model a dynamic system given in a real-word situation.

5.1 The Importance of Markov Chains

An understanding of the concepts behind Markov chains is not essential for everybody. These concepts appear only in curricula of mathematics classes in higher education. Markov chains [13, 38, 66] are of significant importance for engineering and mathematics, when dynamic systems are modeled and analyzed. Markov chains have many applications in natural sciences (e.g., population models), technical areas (e.g., failure models) or economics (e.g., market models).

As a classic application of Markov chains, let us consider a weather prediction model. Our simple model only consists of the three different weather states "sun", "rain", and "cloudy". From a weather state each other weather state can directly be reached as depicted in the transition graph on the left of figure 5.1. The weather changes from day to day according to the given transition probabilities, i.e., the labels of the directed edges in the same graph. The transition matrix P describes the dynamics of this weather model and the initial state vector q_0 describes the initial distribution, i.e., in our example "day 0" is a sunny day. q_1 describes the state vector after one transition and q_∞ the steady state vector of this system.

Whether or not a steady state exists is a typical question for a dynamic system. Other questions might be: "Given that it is a rainy day today, how many days does it take on average until it is a sunny day again?" or "Assume it is a sunny day today. What is the probability that the day after tomorrow is a sunny day too?"

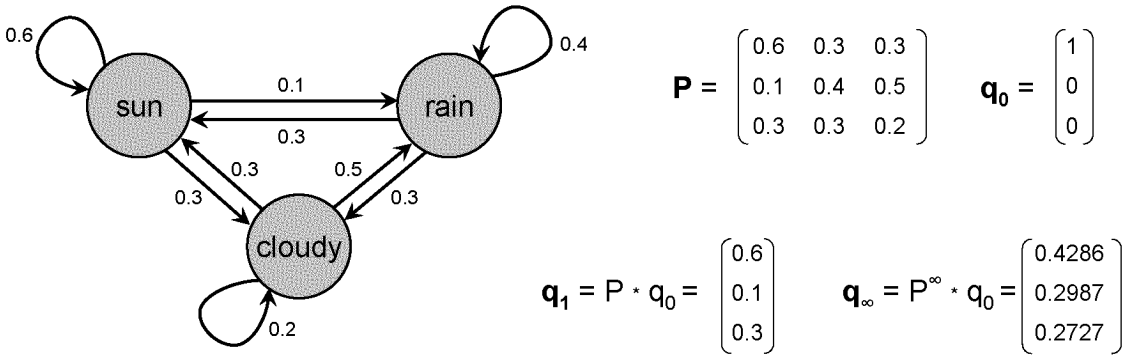

Figure 5.1: Markov chain for a simple weather model

In computer science Markov chains are a common tool to model and analyze for example the likelihood of service availability. Also the mathematical component of Google's PageRank algorithm is the stationary distribution of a finite-state Markov chain, see [48].

5.2 Markov Chains and Linear Algebra

A Markov chain is formally defined as a time-discrete step-wise stochastic process. A full understanding requires previous knowledge from probability theory, linear algebra, and modeling. In this section we show that Markov chains can serve as a suitable example to introduce linear algebra, a topic that is typically covered in high schools.

Matrices are commonly introduced in high schools as two- or three-dimensional transformations. This introduction with reference to geometry does not allow matrices with more than three dimensions and concepts such as eigenvalues and eigenvectors lack obvious interpretations. Markov chains allow matrices with more than three dimensions, the simple weather model from figure 5.1 could for example easily be enhanced with one or more additional weather states. Markov chains further give obvious meaning to terms such as eigenvalue and eigenvector, as the steady state vector is the eigenvector of the transition matrix associated to the eigenvalue 1.

In a scenario where Markov chains serve as an introductory example for linear algebra, not all concepts relevant or related to Markov chains need necessarily to be taught, e.g., concepts specific to Markov chains such as periodicity or recurrence.

5.3 The Program DynaTraffic

DynaTraffic models and analyzes dynamic traffic systems with several intersections and lanes and offers an introduction to Markov chains. For predefined situations, important parameters like the transition probabilities and the current distribution of cars can easily be modified. The program calculates transitions and visualizes the history of the state vector graphically. DynaTraffic uses a Markov chain to calculate the number of cars per lane according to the given transition probabilities.

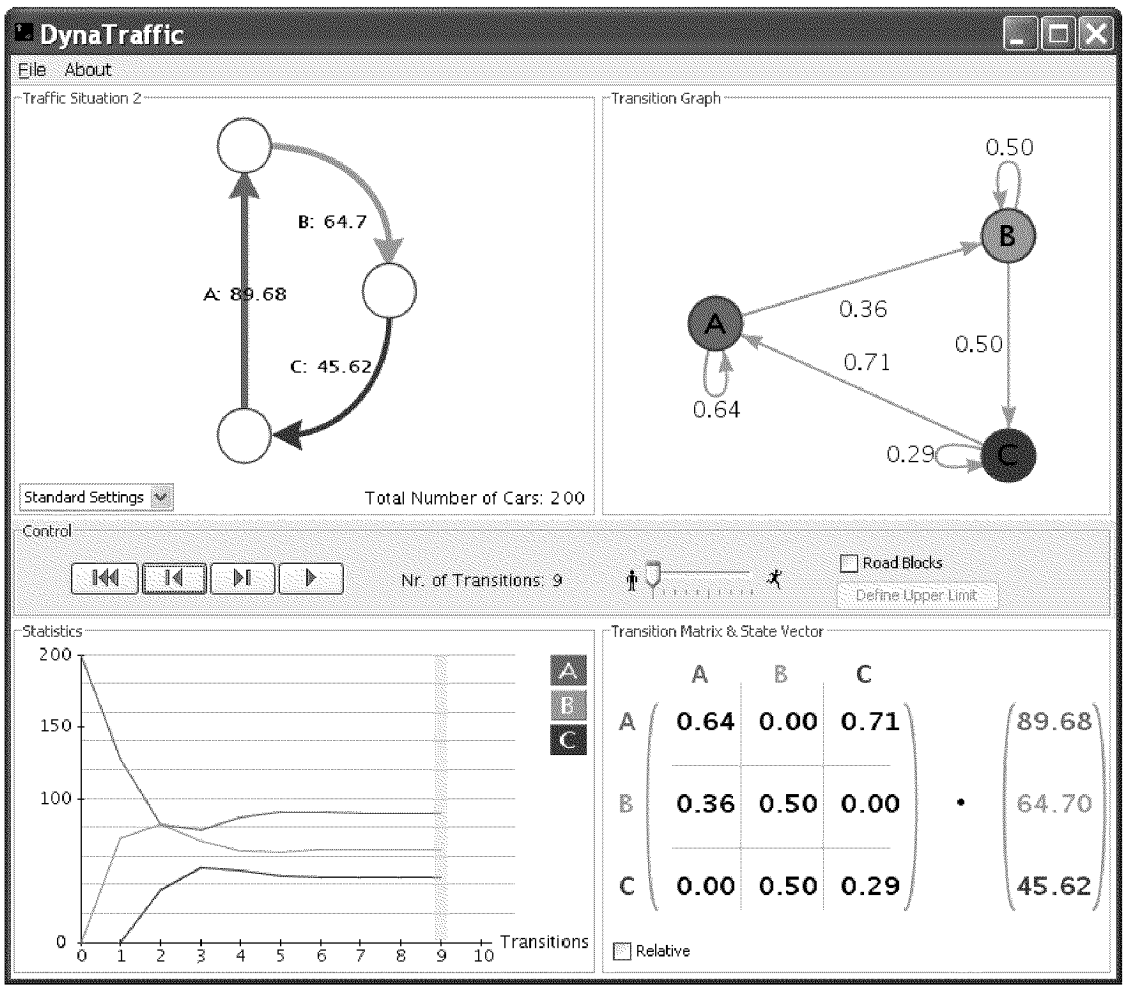

Figure 5.2: The interactive learning environment DynaTraffic

In terms of traffic models as introduced in section 4.5, DynaTraffic

uses a macroscopic traffic model. Traffic is only considered in terms of numbers of cars on certain lanes.

The user interface of the program consists of five parts (see figure 5.2). As in LogicTraffic and QueueTraffic, the top left part of the window shows the current *traffic situation*, here as a directed graph. Vertices represent intersections and edges represent lanes. Labels on the edges display the current number of cars on this lane. The user can click on these labels in order to set the current number of cars.

In reality far more cars are on lanes than on intersections. DynaTraffic thus uses the line graph of the situation graph for its analysis. This line graph has one vertex per edge of the situation graph and is taken as the transition graph of the Markov model used in DynaTraffic, i.e., vertices in this transition graph represent the lanes from the situation graph. The *transition graph* is shown in the top right area of the DynaTraffic window. Arrows depict the possible transitions and labels display the transition probabilities. The user can set transition probabilities by clicking on the edges' label, see figure 5.3. The *control* panel in the middle of the window lets the user execute one transition or a consecutive sequence of transitions at selectable speed. The bottom left part shows a *statistics* view, i.e., a history of the past state vectors. The user can chose for which lanes the charts are displayed. The *transition matrix & state vector* panel at the bottom right finally displays the current transition matrix and state vector.

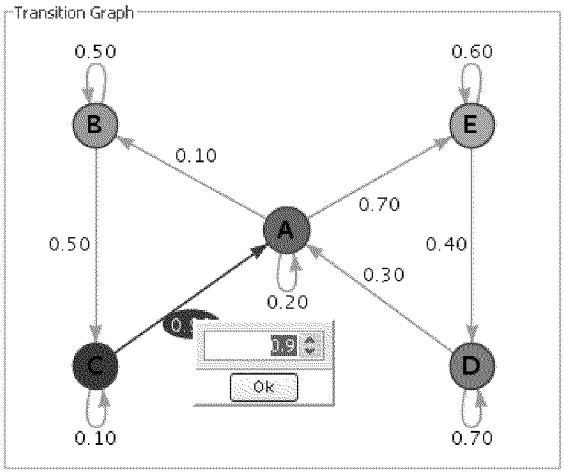

Figure 5.3: Edge highlighting and changing transition probabilities in DynaTraffic

The Markov chain model used in DynaTraffic obviously allows an unlimited number of cars per lane, this can occur, e.g., with a non-stochastic matrix. Such behavior is not realistic, real lanes cannot hold

an arbitrary large number of cars. Therefore in the default mode of DynaTraffic, the number of cars per lane is limited, currently to 2000. The user is not allowed to set the number of cars to a higher value and if one lane reached the limit after the next transition, the execution of this transition is aborted and a message is displayed. To make DynaTraffic more realistic, user-defined upper limits per lane are also allowed to be set. In this "upper limit"-mode, the limits of each lane are displayed in the situation panel as depicted in figure 5.4. If such a limit for one lane were reached after the next transition, the lane gets "closed", i.e., the probabilities of all incoming arrows in the transition graph are set to 0. In order to work with a stochastic matrix, the other rows of the transitions matrix are normalized after closing a lane. In figure 5.4 lane C is closed and therefore colored grey. Whenever the number of cars on a closed lane is less than a certain percentage, currently 70 percent of the upper limit, this lane is opened again and the probabilities of the corresponding incoming arrows are reset to their previous values, followed by a re-normalization of the transition matrix. The detailed algorithms are described in appendix B.3.

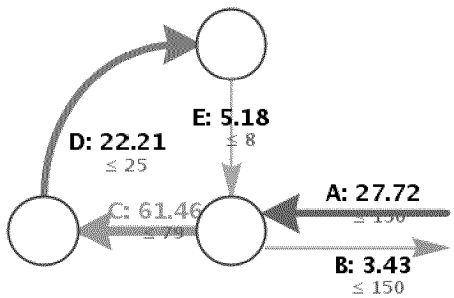

Figure 5.4: Display of upper limits per lane

5.4 Learning Goals and Use of DynaTraffic

The learning goals of DynaTraffic include concepts of Markov chains [13, 38, 66] such as transition graph, adjacency matrix, state vector, periodicity of Markov chains, recurrence of states (i.e., transient, recurrent, absorbing), ergodicity, steady state, convergence, and convergence speed. DynaTraffic allows direct manipulation of the state vector and transition probabilities. Students gain insight into Markov models and the behavior of Markov chains through corresponding views of the situation graph, the transition graph, the statistics, the transition matrix,

and the state vector. Furthermore DynaTraffic illustrates an application of Markov chains and embeds Markov chains in the real-word scenario of traffic analysis .

The use of DynaTraffic in school requires previous knowledge of concepts from modeling [57] and graph theory [31, 82] like graphs, edges, vertices, directed edges, or line graphs. These concepts are not learning goals of DynaTraffic.

Compared to LogicTraffic and QueueTraffic the program DynaTraffic requires more time to get used to because its underlying model is more elaborate. A first introduction to DynaTraffic should explain its model which basically consists of three steps, as depicted in figure 5.5. DynaTraffic is concerned with real traffic situations and as a first step, students must understand that real situations can be modeled as graphs. Since DynaTraffic applies a Markov model to analyze the traffic distribution in lanes and not in intersections, in a second step the situation graph is transformed to its line graph. Edges are interpreted as possible transitions. In a third step, leading to the fourth view of figure 5.5, the transition graph is represented as transition matrix and the current number of cars per lane as state vector.

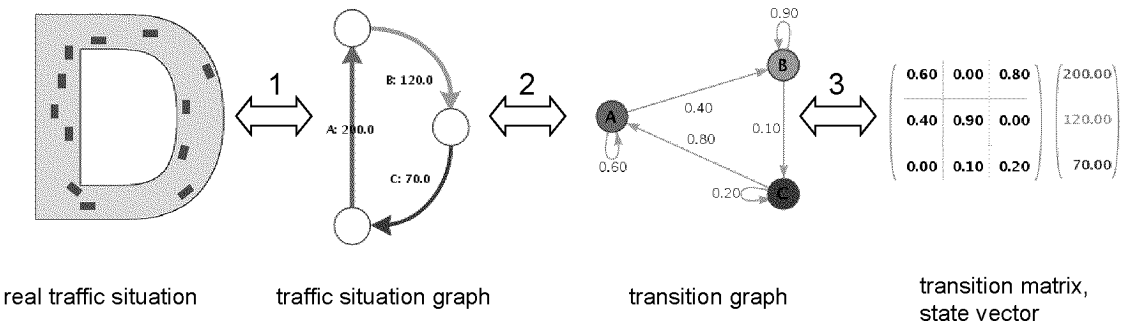

Figure 5.5: The four different views in the model of DynaTraffic

Once the model behind DynaTraffic and the different views of the program are understood, students can tackle typical tasks with respect to Markov chains. First exercises include observation and inquiry of a steady state distribution (see figure 5.6), the behavior of a periodic transition graph, effects of absorbing states, characteristics of transient states, effects of non-stochastic transition matrices, or differences in convergence speed.

Further uses of DynaTraffic might include matrix-vector multiplication, which becomes comprehensible with the displayed matrix and vector including corresponding colors. In advanced classes this might lead

to the concepts of eigenvalues and eigenvectors, especially in connection with the steady state distribution. Additional uses of DynaTraffic can integrate the "upper limit"-mode and use exercises where lanes get closed.

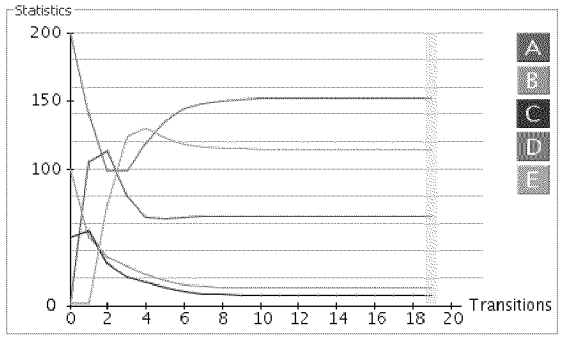

Figure 5.6: DynaTraffic chart showing the convergence to a steady state

5.5 Related Work

DynaTraffic is not the only software that addresses dynamic systems. The following section shortly reviews three exemplary programs.

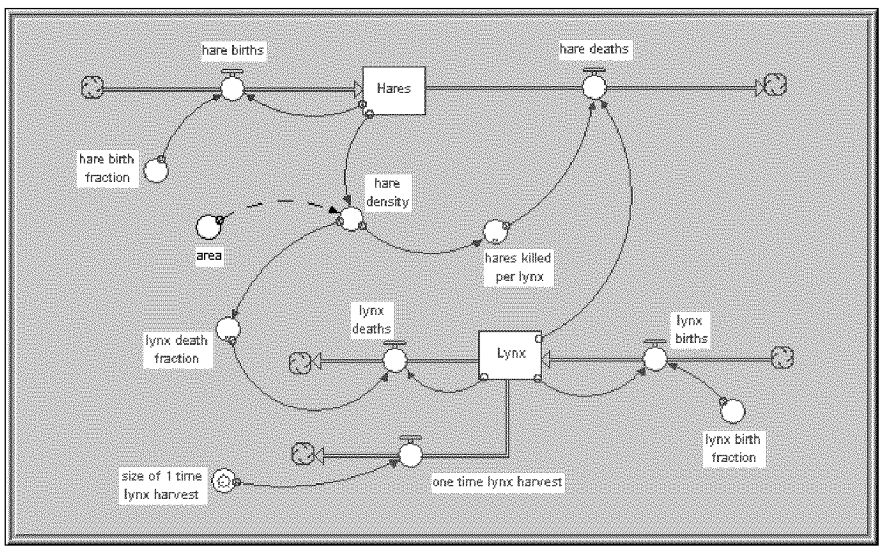

Figure 5.7: Predator-prey model in Stella

Stella

Stella [44] is arguably the most popular educational software for modeling and analyzing dynamic systems. Models in Stella are assembled

from different predefined graphical elements such as sources, sinks, and different dependencies. Mathematically Stella is based on the model of discrete differential equations. These equations are generated automatically and made accessible beneath the graphic model layer.

Figure 5.7 shows a predator-prey model of lynxes and hares in Stella. The populations of lynxes and hares change because old animals die and young ones are born. Furthermore some hares die because lynxes eat them and some lynxes starve if there are not enough hares to eat. Such dependencies can easily be modeled with Stella and the resulting curves are visualized with charts, as shown in figure 5.8.

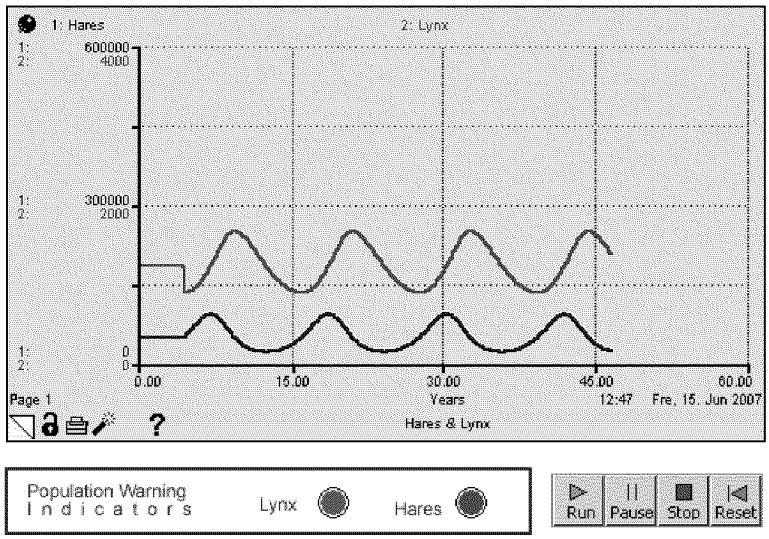

Figure 5.8: Development of lynx and hare populations in Stella

Stella is a commercial tool for analyzing dynamic systems, based on discrete differential equation models. In contrast, DynaTraffic uses time-discrete Markov chain models for its analysis. Both tools are designed for educational purposes, come with extensive teaching materials, and allow a high level of interactivity.

Monopoly in the View of Mathematics

As many other board games, the popular game of monopoly can be analyzed by means of the Markov chains. The web-site of Bewersdorff [15] contains an animation illustrating the probabilities to reach a certain field after a certain number of steps in a monopoly game, see figure 5.9. This animation is a supplement to a book by the same author [14] which gives a short introduction to Markov chains and discusses in

detail how the game of monopoly can be modeled as a Markov chain.

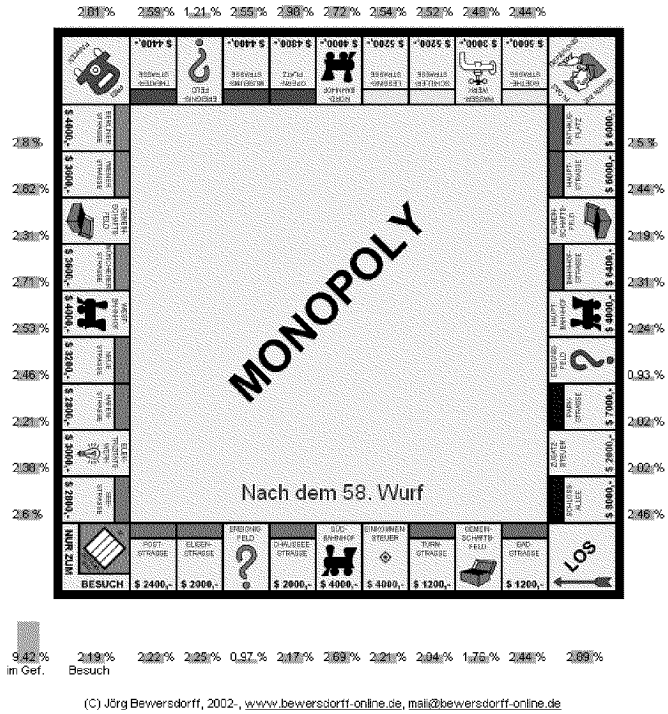

Figure 5.9: The game of monopoly analyzed with a Markov chain

This example shows an interesting application of Markov chains. But the presented animation offers only little interactivity according to the taxonomy introduced in section 2.5.

Matrix Vector Multiplication Applet

There are many visualizations of matrix-vector multiplications freely available online. Figure 5.10 shows as an example two screenshots of an animated matrix-vector multiplication from the MathDemos project [52]. Such visualizations support the teaching of matrix-vector multiplication as they allow a new dynamic view of the multiplication algorithm.

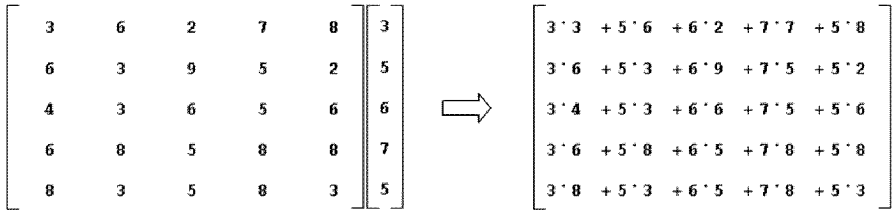

Figure 5.10: Two screenshots of an animated matrix-vector multiplication

This applet is restricted to a symbolic representation of matrices and vectors. A connection between the methods of linear algebra and the description of dynamic systems is not established.

5.6 Conclusions

DynaTraffic is a new interactive learning environment offering the functionality to learn and apply concepts of Markov chains by analyzing a traffic system. There is a considerable number of mathematical concepts and levels of abstraction behind the model of DynaTraffic. Whereas they are all required for a complete understanding of the mathematical model behind DynaTraffic, not all of them need to be introduced formally in order to successfully use the software.

The software examples in the previous section illustrate the strength of computer-aided learning environments like simulation, giving feedback, or animation of processes. The two free online tools presented in the previous section expose a possible pitfall for the implementation of a learning environment targeted at an introduction to an abstract mathematical topic. Despite their animations, both tools offer no interactivity. Students can only observe, they have no control of what is happening. DynaTraffic in contrast has a high level of interactivity as students can dynamically change many parameters.

DynaTraffic displays all the relevant information about the current state of the system as graphs and in the form of matrices and vectors, emphasizing correspondences between these views. An additional chart shows how the system evolves over time. DynaTraffic allows a new attractive access to Markov chains based on the real-word scenario of traffic analysis.

6 On the Development of Interactive Learning Environments

When integrating a medium such as computers into teaching, a number of questions arise. A main pedagogic concern is the usefulness or advantage compared to the use of other or no media. From an economic point of view also the costs of development, maintenance, and infrastructure requirements are relevant.

Since the 1970s, computer-aided interactive learning environments were said to have had a big potential. Reality looks mostly different. Many learning environments are restricted to "Drill & Practice", and they are hardly interactive. Others are for technical reasons – and despite high development costs – not operational any more after a few years. A sustainable development of interactive learning environments has to accommodate at the same time school practice, rules of software engineering, and knowledge of educational sciences. Such an interdisciplinary approach deliberately faces a "scientific fuzziness" and therefore exposes itself to criticism from the communities of the individual scientific domains.

In this chapter, we first give a historical review of the introduction of different media in teaching and identify analogies among their use in education. We then concentrate on computer-aided learning environments and show what we might learn from the history of computer-aided instruction. Finally, we present our engineering science approach that stands the test in practise and introduce and illustrate our pragmatic recommendations with examples from InfoTraffic.

6.1 Media in Education: Expectations and Disappointments

Media play an important role in the history of teaching. As soon as new means for storing, displaying, or broadcasting information become

popular, possible applications for educational purposes are discussed. We now analyze the use of the media radio, TV, and computer in education and identify some analogies.

In the mid-1920s, the first institutions broadcasted school radio programs. The expectations towards the new medium were immense. Wülser [84] writes:

> Vorträge, Sprachkurse und Beratungen für fast alle Lebenslagen prägten von Beginn an die Programme, wie beispielsweise ein Blick in die Sendewoche vom 5. bis 11. Februar 1927 zeigt. Radio Zürich bot Kinder-, Jugend-, Schüler- und Frauenstunden an, gab eine Englischlektion, liess den Vortragsdienst der Volkshochschule zweimal sprechen und veranstaltete einen bunten Strauss von Vorträgen wie "Klassische und moderne Bildhauerei", "Winterschnitt am Kernobst", die "Physik des Mondes" oder "Zürcher Verkehrsfragen".

Initiatives to equip schools with receiver stations (similar to today's "schools go online" initiatives, see figure 6.1) accompanied these school radio programs.

With the appearance of the visual medium TV, radio lost its fascination. School TV programs became popular and school radio programs played only a marginal role. With the advent of computers in the 1970s serious competition grew to school TV, which the boom of the Internet later fortified. Educational software and learning environments in the form of web-based applets up to edutainment systems were expected to offer cost-effective, individualized, place and time independent learning along with adaptive feedback. Meanwhile it is clear that this software failed to meet many of the expectations. Clark [32] even claims that "media will never influence learning".

As Mattern [55] describes, the computer as a teacher triggered many visions of futuristic scenarios for our information age. Mattern's survey further quotes computer science professor Philip Agre from the University of California in Los Angeles: "The universal online university has been predicted in pretty much its currently hyped form for almost forty years. And we see here the characteristic shortcomings of these predictions: the lack of emphasis on education as socialization into a professional culture, the desire to automate teachers completely rather than providing teachers with advanced tools."

Figure 6.1: School goes radio. (Source: archive Swiss Radio DRS, Zürich)

Some parallels can be identified when considering the usage of the three media radio, TV, and computer (including the Internet) for educational purposes:

Lack of Interactivity. Students find themselves too often in the role of the passive consumer. For learning environments, interactivity has to go beyond mere navigation. Students should have active control over content, activities, and aspects of the representation. Laurel [49] makes the point: "You either feel yourself to be participating in the ongoing action of the representation or you don't."

It is interesting to note that TV companies today place more and more emphasis on interactive TV such as video on demand or tele-shopping for the sake of customer retention.

High Costs of Development, Short Half-life. For all three media, the cost of development of educational materials is typically high. This cost in general is only justified for topics whose expected relevance outlasts a few years and topics that target a broad audience.

For the development of professional radio or TV programs, amateur skills are not sufficient. The same holds for computer-based learning environments. Today, especially young people have high expectations because of their lifestyle with permanent exposure to media [41]. Edutainment produced under a lack of resources quickly degenerates to "edupainment".

Maintenance and update of radio-, TV-, or computer-based teaching materials is more demanding than for classic printed materials. In the case of computer-based learning environments, the short development cycles of soft- and hardware turn teaching units too often inoperable after a few years.

High Requirements to the Infrastructure. Paper and pen as well as printed teaching materials can be used "just in time and anywhere". The black board is as well a quite affordable, many times usable and thus a long-living teaching help.

The cost is different with the use of radio, TV, and computers including the Internet. All these media require notable investments in infrastructure. Handling this infrastructure is in general not easy for instructors and accompanied by pitfalls. Computer-based learning environments for example often require installing additional software.

Assuming these parallels bear some universality, they should be kept in mind when developing teaching materials using media. These observations might become valuable hints to avoid possible pitfalls for future developments.

6.2 Computer Aided Instruction

In the following, we concentrate on the development of computer-aided learning environments for *computer science* (CS) education. Concepts from CS are obvious teaching topics for *computer aided instruction* (CAI). First of all, formalization of facts from CS is often easy and therefore feasible for a computer and secondly, CS teachers are familiar with the use and maybe even with the development of software systems.

CS played a precursory role in using computers in teaching. Among the early important projects are Seymour Papert's activities around the programming language LOGO for children or John Kemeny's and

Thomas Kurtz's programming language BASIC. During the 1970s, PLATO (Programmed Logic for Automated Teaching Operations) went one step further and thousands of students used it for about two decades on a regular basis.

The sometimes euphoric reports from the 1970s about the potential of CAI differ only little from today's reports. Nievergelt [58], who was involved with the project PLATO, warned already 1975 of excessive expectations, for example:

> Restriction to a few fixed teaching strategies appeared to be unreasonable. Programmed instruction and drill in particular, with their rigid control of the dialog by the program, should yield to (or at least not exclude) modes where the user controls the dialog, such as inquiry and simulation. [...]
>
> Resources had been diluted into too many projects of insufficient size; CAI research and development should be carried out by sizable groups of system designers and authors.

Nievergelt [58] finishes with the following recommendations:

> I came to the conclusion that there is no systematic body of knowledge which is of relevance to such a task. I am afraid that this paper does not change this situation at all. The advice I might give to someone intent on building a computer-based instructional system could be summed up in a few phrases: get the best terminals you can pay for, good programmers, try everything out in actual instruction as soon as possible, and follow your nose.

To some researchers these 30 years old recommendations from Nievergelt might seem non-scientific, but they reflect the ongoing tension between a scientific claim and a simultaneous impact on school practice.

6.3 Innovation vs. Evaluation

In Switzerland, a number of interactive learning environments for teaching CS have been developed within the last few years. The programmable ladybug Kara [61, 63] for an introduction to programming is one prominent example and won the European Software Award in 2002. Other examples are GraphBench [21, 23] (NP-completeness and graph

algorithms), Exorciser [77] (regular languages, context-free grammars, and Markov algorithms) or Soekia (a didactic search engine). All these learning environments along with teaching materials are freely available on the educational server SwissEduc [3].

During the development of these environments, their authors have repeatedly experienced the tension between "doing science" and educational innovation. From the point of view of educational research these environments lack a scientifically justified evaluation of their impact. But the complexity of interactive learning environments with its many variables sets tight limits for a methodically supported evaluation. Schulmeister [69] elaborates how educational researchers tend to almost compulsive differentiation and control in their methodical design of studies to gain more generalizable results. According to [69] this leads to artificial learning environments which are irrelevant for instructional practice. For a deeper elaboration of this issue see section 7.1.

6.4 An Interdisciplinary Engineering-Science Approach

Besides not being scientifically evaluated according to the standards of educational research, the development of learning environments misses the research aspect from the point of view of CS as an engineering science. The curriculum of the schools determines the topics covered which are well established and thus not a current area of research. Furthermore the famous "as simple as possible" principle is true for learning environments. The user interface should be restricted to the minimum to allow fast familiarization with the system, thus preventing a waste of time. Finally software architecture should be slim to allow maintenance with scarce resources.

Developers of interactive learning environments thus face a dilemma: The educational research community degenerates them almost inevitably as not being scientific enough, even though these environments might sustainably influence teaching and learning. In the eyes of the computer science community, innovation is missing.

A possible approach to overcome this dilemma is to intentionally integrate the different communities involved. As presented in [5], the development of computer-based learning environments requires the interdisciplinary collaboration of three different groups of persons:

Experienced Instructors. Active instructors with a lot of school practice must identify and determine topics, exercises, and use.

Software Engineers. The efficient development of high-quality learning environments with a reasonably long half-life is an engineering task and thus a software engineer must perform it.

Educational Researchers. Only educational science can offer basic didactic concepts such as learning theories and the particular methodology required to evaluate the effectiveness of interactive learning environments.

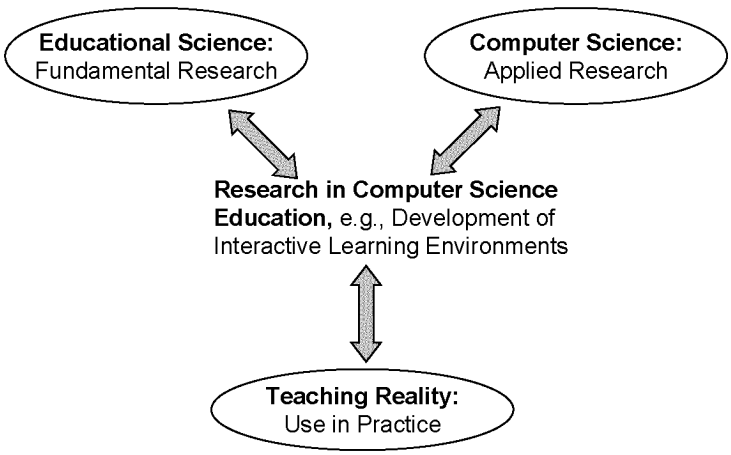

Figure 6.2: Interdisciplinary research in computer science education

If in a development team one of the above groups is not represented, development will most likely be suboptimal. Within the last decade, e-learning systems have been developed for which there was no demand in school practice. And when instructors or educational researchers acted as programmers, this often resulted in user unfriendly and technically only partly operable programs.

6.5 Pragmatic Recommendations

Good educational software and e-learning systems ideally satisfy three superior criteria:

Didactic. The learning environment offers a didactic added value; students learn more and better.

Organizational. The learning environment creates organizational added value, instruction becomes "easier" for instructors and students.

Economic. The learning environment pays off economically, instruction becomes "cheaper" for educational institutions.

It is wrong to believe that all these goals can be achieved at the same time substantially. However, we believe that our following pragmatic recommendations are a step in the right direction. Where appropriate, we shortly illustrate or comment each recommendation with examples from the InfoTraffic project.

1. **Is the computer needed at all?** It makes no sense to develop learning environments for topics that can equally well or even better be taught without computers. Especially suited for learning environments are complex topics, where individualization allows accommodation of different pre-knowledge and learning rates. Paper implementations of simple "Drill & Practice" exercises often require less effort.

   As elaborated in chapter 3, logic instruction is usually unnecessarily abstract. LogicTraffic with its intuitive and everyday-life-based approach for controlling traffic intersections offers a didactic added value. Computers do a great job at allowing virtual-enactive representations.

2. **Is the topic to be taught still relevant in 10 years?** The development of interactive learning environments is in general expensive and thus only justifies for long-term relevant topics. This point can be seen as a short-question version of fundamental ideas in computer science as elaborated in section 2.1.

   Logic accompanies us in everyday life and will as well be of crucial importance for many areas of science in the future.

3. **Use-cases: Is interactivity possible?** A high level of interactivity, for example following the taxonomy of Schulmeister [68], is crucial for the quality of a computer-aided learning environment. The sole reading of a text from the screen or the observation of an animation alone does not trigger a learning process.

   Not every topic is suitable for computer-aided human computer interaction. It is recommended to collect example exercises already

in the specification phase. In the language of software engineering these exercises represent use-cases for the program. These example exercises allow to verify that the learning environment leads to real interaction with the students and whether different cognitive levels are addressed.

"Analyze this given traffic control. Which measures can reduce the global waiting time?" was one example exercise for QueueTraffic. Such examples define the functionality that the learning environment must provide.

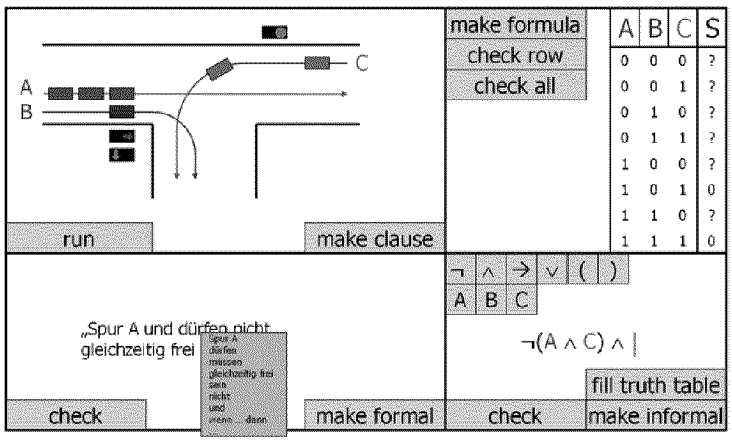

Figure 6.3: Mock-up GUI of QueueTraffic

4. **Paper-based prototyping.** It is well known from software engineering that an early inclusion of *graphical user interface* (GUI) aspects substantially saves costs. In learning environments, GUIs play a particularly important role. It is well worth to work on paper as long as possible. In this phase besides paper other simple tools such as sticky tape, post-its, scissors, and flip charts are useful. Also presentation tools are suitable means for a quick development and variation of GUIs. A good introduction to the method "paper prototyping" can be found in [72].

   Developing InfoTraffic we worked a long time with paper prototypes only. One of the main advantages of paper-based prototyping: Reviewers more easily dare criticize and point out weaknesses or question parts of the project. If a programmed prototype exists, the inhibition threshold for fundamental criticism is generally high.

   Figure 6.3 shows an early PowerPoint prototype of LogicTraffic. In

this mock-up there was still a panel for a natural language description of formulas in PL. This option proved difficult to incorporate while solving exercises with the paper prototype and was therefore canceled.

5. **Rapid prototyping.** After having fixed use-cases and the GUI on paper, a first prototype should be implemented. Developers should show it as soon as possible to selected test persons – students and instructors – which should use it experimentally.

   Figure 6.4 shows the first publicly presented version of LogicTraffic. The visual differences to the actual version (Figure 3.1) are obvious. The internal program structure has changed a lot too, and traffic situations are for example not produced with an image file anymore, but automatically based on a XML file.

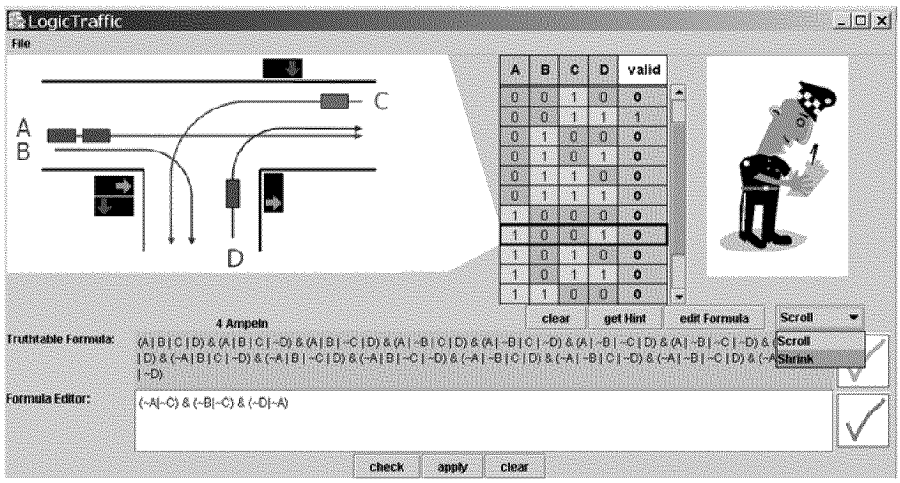

Figure 6.4: Older version of the learning environment LogicTraffic

6. **Technical requirements: as simple as possible.** The big development effort for a learning environment is only justified if the software runs on different platforms and maintenance is ensured for longer periods. This calls for simplicity and abandonment of unnecessary functionality.

   InfoTraffic is programmed in Java and needs only a *Java runtime environment* (JRE); this is today available on most school computers.

7. **Early testing.** As soon as possible, first trials in classes should be performed. These trials force the development of concrete exercises and accompanying teaching materials and discover weak-

nesses. Students think and act often differently from what instructors imagine.

With LogicTraffic and QueueTraffic the testing showed that students easily manage to use the programs without elaborate labeling of individual buttons. German speaking students further have no difficulties with labels in English. DynaTraffic has not yet been tested with a substantial group of students.

8. **Economy at the user interface.** The (graphical) user interface of a learning environment should ideally be self-explanatory. After initial testing, it is important to undertake a review process. Each button, label, and other graphical element has to be critically rethought: Is this element really needed and is it at the right place?

   Our experience shows that through this critical analysis up to half of the original elements disappear. For professional development of learning environments, also scientific findings from multimedia learning should be taken into consideration, see for example the different principles Mayer [56] presents.

9. **Propagation of the learning environment.** The work is not finished when there is a first stable version of the learning environment. In order to have any impact on teaching, the learning environment must be available to students and instructors, for example through a popular web site. The possible channels for propagation should be cleared up before the development begins.

   Experience from the Swiss educational server `swisseduc.ch` shows that free didactic materials (e.g., introductory presentation, user instructions, exercise sheets, and didactic background information) must accompany learning environments.

10. **Securing maintenance and continuity.** After release, i.e., after the transition to operation mode, maintenance has to be ensured as is common for projects in computer science. This includes explicitly fixing any known bugs, the implementation of enhancements, and adaption to new versions of operating systems and other involved software. It is further important during this phase to gain trust. Lack of time is one of the main problems in most instructors' lives. An instructor therefore reasons carefully whether or not he should invest effort in preparing a topic with a new

learning environment. He will have to assess whether the software will still be available and operable within some years. Many interactive learning environments are built in universities as student activities and continuity is therefore not ensured. Here suitable solutions should be preponed.

Usually the developers of software receive requests for enhancement. For learning environments, the developers should be cautious: The quality of a learning environment is not solely measured by the number of features provided. Frequent revisions and enhancements make users uncertain. Typically related teaching materials have to be updated too, which is a tedious job. In addition, enhancements typically increase the complexity of the programming code and complicate its maintenance. "Less is more" is therefore the general motto for enhancing learning environments.

6.6 Conclusions

History shows that the integration of media into teaching and the development of high-quality interactive learning environments are tedious and difficult tasks. For the success of a learning environment, the teamwork of experienced instructors, well-versed software engineers, and specialists from educational science is indispensable and decisive.

The "usefulness" of computer-aided learning environments can hardly be evaluated with classic scientific criteria based on educational science, as will be elaborated in chapter 7. A promising pragmatic approach is to bring together knowledge from different areas, and thus to have partners with the willingness to cooperate interdisciplinarily. Such an approach takes courage because one is exposed to the almost certain criticism from the scientific community of educational science on the one hand and of computer science on the other hand. The scientific world might frown upon "Try everything out and follow your nose" from Nievergelt [58] but this statement is a promising engineering science approach for innovations in the field of e-learning.

7 Evaluation, Use and Experience

The presented work focuses on the development of interactive learning environments. InfoTraffic follows both an engineering science and a design-based research approach. During the process of designing and developing InfoTraffic we have gained insights that we can share. However, we did not evaluate InfoTraffic based on a classic experimental study. From the perspective of educational science, grounded on hypotheses and systematic experiments, it might seem essential that a doctoral project such as InfoTraffic provides a scientific evaluation of the "usefulness" of its outcome, i.e., numbers expressing how much better students learn the underlying topics, propositional logic, queuing theory, and Markov chains. On the other hand, many recent publications [12, 28, 39, 64, 69] criticize this classic approach to a so-called scientific evaluation by experiments. Schulmeister makes this point very clear and states in his renowned book [69]:

> There are thousands of reports by teachers on experiments in school, mostly with inadequate setups, but with sophisticated controlled test designs in part. Almost all of them report a learning increase in the end. Again and again, one is tempted as reader, if the subject is right, if the study agrees with one's own prejudices, to cite such results. But we do not need any of those "careful studies of the impact of ... on ...". What we need are teachers and lecturers who are highly motivated, who can fill their pupils and students with enthusiasm, and programs that are interesting, thrilling, highly interactive, and aesthetically designed.

In this chapter we elaborate on the difficulty of scientific evaluation of interactive learning environments and present different approaches to educational research. We then outline the engineering science and design-based research approach InfoTraffic follows and explain why we did not conduct a large-scale classic scientific evaluation. Finally we summarize our experience with InfoTraffic and give an overview of the use of the learning environments.

7.1 On the Difficulty of Scientific Evaluation of Interactive Learning Environments

From the scientific tradition of natural science it is clear that the usefulness, advantage or added value of a new method, approach or product must be measurable and reproducible in experiments. First a hypothesis is stated and then tested against systematic observations to see whether it holds or not. Hypotheses are stated by means of variables. So in order to state reasonable hypotheses, the relevant variables involved in the intended experiment must be identified. The complexity of real learning and teaching situations with its many effective variables and their manifold interactions with even other variables sets tight limits to such an experimental research. According to Reinmann [64], this leads to the case that most comparisons of teaching methods and teaching media show no significant results, and the few significant results contradict each other.

According to Schulmeister [69], difficulties in generalizing statements from evaluations regularly tempt methodologists into calling for further differentiation and control in the methodical design. This leads to the construction of utterly artificial learning environments, whose evidence thus loses its validity for real life situations, i.e., these learning environments are meaningless for teaching practice. It is thus not surprising that as Schulmeister [69] puts it, "the trivial nature of some results can hardly be surpassed."

In addition to the fundamental difficulty of scientific evaluation just mentioned, another issue is the appreciation of different efforts by the scientific community. As the title "Nur 'Forschung danach'? – Vom faktischen und potentiellen Beitrag der Forschung zu alltagstauglichen Innovationen beim E-Learning" of [64] suggests, Reinmann criticizes that currently only "research afterwards" is tolerated where a new concept or a new technology is only scientifically evaluated after it is developed and employed, and obviously no "science happens" during its development. Reinmann further asks for the goals of educational research in our society and whether this research should contribute to innovations in education.

7.2 Approaches to Educational Research

In recent years, many educational researchers [12, 28, 39, 64, 69] have observed a lack of applications of research findings in the field of education. This observation has started a large ongoing international debate, and some researchers like Reinmann [64] even speak of a crisis in innovation in (traditional) educational research. This discussion is not over yet, but so far it has certainly been helpful in fleshing out, comparing and discussing different approaches to educational research. In this section we summarize different approaches to the hard-to-do-science that educational researchers according to Berliner [12] do.

Following Reinmann [64] and Fischer et al. [39], we use a model by Stokes [74] to illustrate and compare different approaches to educational research. The quadrant model of scientific research by Stokes distinguishes the two dimensions "quest for fundamental understanding?" and "consideration of use?" which in this simple model can both only be answered with yes or no, see figure 7.1. To three of the resulting quadrants, Stokes assigns well-know scientific personalities as typical representatives, namely Niels Bohr, Louis Pasteur, and Thomas Edison. The bottom left quadrant remains empty.

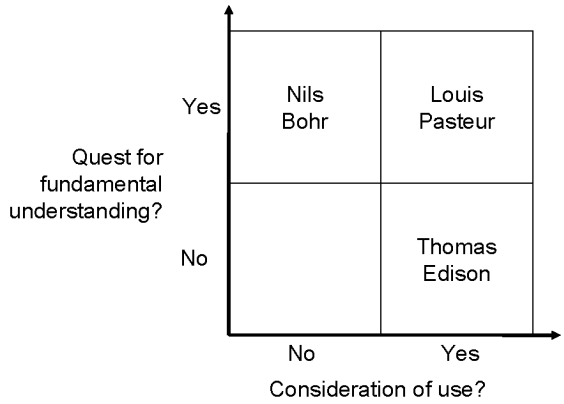

Figure 7.1: The quadrant model of scientific research by Stokes

Stokes' model serves as a reference for different approaches to doing research and to locate three proposed approaches to educational research. As an outcome of the international discussion in educational research described above, several solutions and reactions have been proposed and discussed. Below we list the three main approaches to educational research that according to [39, 64] emerged from this discussion.

Pure Basic Research. A first direction states that scientific standards should be applied more strictly in educational research and experiments should be conducted more rigorously. This approach claims to lead to new, replicable and generalizable insights that can be accumulated and directly transferred to practice. This approach can be located in the quadrant of Bohr in Stokes' model and claims that applicability will follow automatically if new insights are generated.

Use-Inspired Basic Research. A second direction emphasizes that in research the aspect of use is neglected, i.e., educational science produces results, but not the ones practice needs. This line of reasoning has led for example to the approach of *design-based research (DBR)*, which is located in the quadrant of Pasteur in Stokes' model. DBR is, according to Fischer et al. [39], a mixture of empirical-pedagogic research and the development of teaching and learning environments in a practical context, e.g., in the context of teaching in a classroom. Similar to DBR is the *integrative research* approach of Stark and Mandl as described in [51].

Engineering-Science-Inspired Research A third direction emphasizes an engineering science approach. According to Burkhardt and Schoenfeld [28], "the engineering approach to research is directly concerned with practical impact – understanding how the world works and helping it 'to work better' by designing and systematically developing high-quality solutions to practical problems." An engineering science approach is similar to DBR but requires a more radical rethinking, as the underlying logic extends from "understanding how the world works" to "helping it to work better" in the idealized and simplified words of Burkhardt and Schoenfeld just mentioned. The engineering approach is like use-inspired basic research located in the quadrant of Pasteur in Stokes' model and could according to Reinmann [64] be called *fundamental-oriented applied research* (grundlagenorientierte Anwendungsforschung), as it mainly focuses on developing new applications but at the same time seeks for basic and general knowledge.

There is no consensus yet in the international discussion on which approach is best. Reinmann [64] concludes that educational science as a hard-to-do-science cannot follow only one research tradition. Rein-

mann is further convinced that engineering science as a new pillar for educational science will lead to new impulses and innovations.

7.3 The Approach of InfoTraffic

InfoTraffic follows both an engineering-science-inspired and a design-based research approach. Our interdisciplinary engineering science approach is described in section 6.4. Table 7.1 lists the five characteristics for good DBR as proposed by the design-based research collective [37] with links of each characteristic to InfoTraffic.

Characteristic	**Link to InfoTraffic**
The central goals of designing learning environments and developing theories or "prototheories" of learning are intertwined.	See contributions of this thesis in section 1.4.
Development and research take place through continuous cycles of design, enactment, analysis, and redesign.	See our pragmatic recommendations in section 6.5, especially points 3, 4, 5, 7, and 10.
Research on designs must lead to sharable theories that help communicate relevant implications to practitioners and other educational designers.	See the contributions of this thesis in section 1.4 and our pragmatic recommendations in section 6.5.
Research must account for how designs function in authentic settings. It must not only document success or failure but also focus on interactions that refine our understanding of the learning issues involved.	See the report of uses of InfoTraffic in section 7.4.
The development of such accounts relies on methods that can document and connect processes of enactment to outcomes of interest.	See the contributions of this thesis in section 1.4 and the didactic concepts in chapter 2.

Table 7.1: InfoTraffic and the five characteristics for good design-based research

Intentionally we have not evaluated the InfoTraffic learning environments with traditional experiments. There are two reasons for this decision: Firstly severe systematic difficulties and problems with traditional experimental evaluation exist, i.e., conducting traditional experiments is a difficult and demanding task. Secondly, in addition to developing and iteratively improving the learning environments, conducting such experiments requires resources out of the scope of one single doctoral project.

We have applied an iterative process for the design, development and analysis of InfoTraffic, corresponding to the second characteristic for

good design-based research in table 7.1. Early paper-based prototypes of the learning environments, possible use-cases and types of exercises were analyzed and discussed. As soon as a first prototype was available, we conducted usability tests with individual users. Through this process the software and the concepts of InfoTraffic evolved and improved constantly, as is typically the case in engineering science.

The goal of InfoTraffic was to have an impact in teaching practice. Since engineering science is directly concerned with practical impact, it was obvious for us to choose such an approach for our research. As Burkhardt and Schoenfeld [28] put it: "The research-based development of tools and processes for use by practitioners, common in other applied fields, is largely missing in education." InfoTraffic contributes to research-based development of tools for practitioners, i.e., instructors.

7.4 InfoTraffic: Uses and Feedback

Our interdisciplinary engineering science approach requires and includes knowledge from different areas. The working concepts and designs should therefore be presented to different groups in order to get feedback and impulses. This section summarizes occasions where the InfoTraffic environments were used with or presented to larger audience. A detailed chronological list of uses and presentation can be found in appendix A. Note that we here exclusively consider uses by the authors, even though the download statistics in the next section suggest active uses by many other users.

InfoTraffic on SwissEduc

SwissEduc (`www.swisseduc.ch`) is a non-commercial, web-based and free offer of teaching materials for secondary education. The topics covered by one of Switzerland's most popular educational server include among others Chemistry, Latin, English, Physics, Geography, and Computer Science. Since May 2006 LogicTraffic has been available on SwissEduc along with an introductory presentation and exercises with sample solutions, see figure 7.2. QueueTraffic has been available since July 2006, and DynaTraffic since September 2007.

Table 7.2 shows access and download numbers for InfoTraffic on SwissEduc. Note that these numbers exclude web robots as well as downloads from within ETH Zurich. Further many files might be shared internally

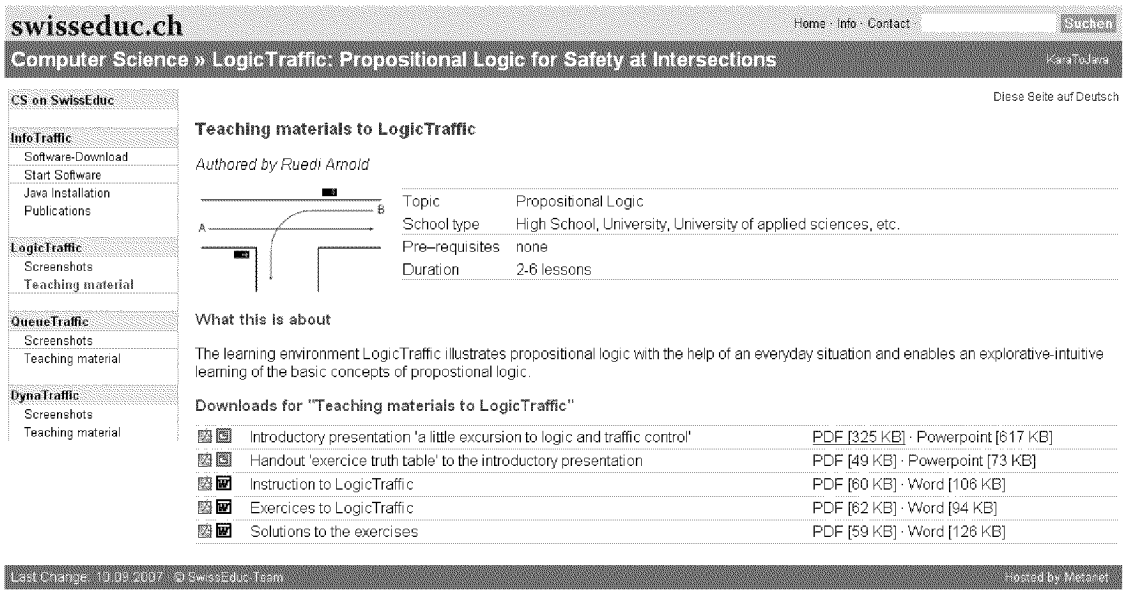

Figure 7.2: Teaching materials accompanying LogicTraffic on SwissEduc

in schools, thus leaving only one download in our statistics for use by potentially larger number of users. The numbers show that many people are interested in InfoTraffic.

Page views	3012
Downloads of InfoTraffic	108
Different visiting hosts	422

Table 7.2: InfoTraffic statistics on SwissEduc for September 2007

Secondary Education - High School

Early classroom trials force the development of practical assignments and accompanying teaching materials and thus help discovering weaknesses. Students use software often differently from what developers and instructors fancy. The student's feedback at these early trials was collected, analyzed, and discussed in the developer team. We used and tested LogicTraffic and QueueTraffic as soon as possible in class. DynaTraffic is also scheduled for class use on a broad basis.

Tertiary Education - Use in Academia

InfoTraffic can also be used in education at university level. In the winter semester 2006/07 and in the autumn semester 2007 LogicTraffic

was part of the compulsory logic course for first year computer science students at ETH Zurich. The students were given an introduction to LogicTraffic and had to solve exercises using LogicTraffic. The students' feedback were all positive.

Teacher Education

As described in chapter 6 one of the three groups involved in our interdisciplinary engineering science approach are practitioners, i.e., teachers and instructors. So we appreciated the possibility to receive their valuable feedback at various occasions.

Research Presentations

We presented the concepts and prototypes of InfoTraffic several times to the research community in computer science education. At all these occasions, we received valuable feedback and various design issues and development decisions were discussed.

Anecdotal Evidence

To round off this chapter, we here quote some exemplary feedback. More collected feedback can be found online at [3].

"Logik ist cool!" (Spontaneous statement of a high school student after having attended an introductory presentation to LogicTraffic.)

"Poissonverteilung ist viel zufälliger, nicht wie bei einem Fliessband, wenn jede Sekunde ein Teil aus der Maschine kommt." (High school student after having solved exercises with QueueTraffic.)

"Ich möchte einfach gratulieren zur Software LogicTraffic. [...] Mir persönlich hat das Programm wirklich gut gefallen, und ich habe damit im Bereich KNF, DNF usw. viel gelernt. Die Grafik finde ich auch super, und das ganze Interface ist wirklich total user-friendly. Sehr nützlich finde ich auch, die verschiedenen Formeln in Files speichern zu können." (CS student at ETH Zürich after having solved exercises with the help of LogicTraffic.)

"Ende März konnte ich Ihren Vortrag in Dresden erleben und habe fast den gesamten Inhalt in meinen Informatikunterricht integriert, das heisst, ich habe die Schüler (13-19 Jahre) auf die Möglichkeiten/Grenzen

von Simulationsprozessen hingewiesen und Ihre Simulationen testen lassen." (High school teacher after using QueueTraffic.)

"We met this summer at ITiCSE in Dundee. As I mentioned then, I liked your work on the teaching tools and I'm using LogicTraffic in the classroom this semester. [...] I really like the interactive style of LogicTraffic and it works well to let the student experiment with the models. [...] I really believe that interactive learning with devices such as LogicTraffic is the best way for today's generation of students to learn. They certainly do not learn from traditional lectures, at least in my experience." (American university professor.)

7.5 Conclusions

The focus of our work is the development of state-of-the-art interactive learning environments following an an engineering science and design-based research approach. InfoTraffic contributes to design-based development of educational tools for practitioners. The learning environments are readily available for use in class. Our interdisciplinary engineering science approach is described in section 6.4

The InfoTraffic learning environments intentionally were not evaluated with traditional learning experiments for two reasons: Firstly and as elaborated in section 7.1, conducting such evaluation is difficult and severe systematic criticism of this kind of evaluation exists, for example due to the large numbers of parameters influencing classroom teaching. – To use the clear words of Schulmeister [69] again: "we do not need any of those 'careful studies of the impact of ... on ...'." Secondly, conducting such experiments was out of the scope of a doctoral project in computer science.

The InfoTraffic interactive learning environments we have presented were iteratively tested on many occasions and accordingly improved. The current spread of InfoTraffic and the feedback we have received during this project lead us to conclude that InfoTraffic has reached its goals and found its way to teaching practice.

8 Results and Outlook

The goal of this thesis was to gain insight into how to teach abstract topics and into the development of high-quality interactive learning environments. This goal was reached by following an interdisciplinary engineering science approach and combining knowledge from teaching practice, educational science, computer science education and software engineering. The three new interactive learning environments LogicTraffic, QueueTraffic, and DynaTraffic have been used and presented on several occasions so far. They are readily available for use at `http://www.swisseduc.ch/compscience/infotraffic/` along with teaching materials.

Much research in computer science education traditionally focuses on classic natural scientific evaluation. As elaborated in chapter 7, Schulmeister [69] and others question the habit of putting the focus in educational research unilaterally on this kind of evaluation. Other approaches to educational research have also been discussed lately. InfoTraffic follows both an engineering-science-inspired and a design-based research approach.

A next reasonable development step for the InfoTraffic interactive learning environments might be to expand them for self study purposes. Such enhancement might include the integration into a suitable self study learning environment or an adequate blended learning framework as well as the development of automated self tests.

The current spread of the InfoTraffic environments and the feedback we have received so far lead us to conclude that InfoTraffic has found its way into teaching practice. We are convinced that the insights gained during the InfoTraffic project are of some help to other developers and that our findings as well as our interactive learning environments will spread further.

A Uses and Presentations of InfoTraffic

As mentioned in section 7.4, this appendix chronologically lists occasions where the InfoTraffic environments were used with or presented to larger audience, sorted by target groups. Note that we here exclusively consider uses by the authors.

Secondary Education - High School

- 19.01.2006: First use of LogicTraffic in two high school classes in Zurich.
- 15.03.2006: Use of LogicTraffic in a class of visiting female high school students at ETH Zurich.
- 05.07.2006: First use of QueueTraffic in a high school class in Baden.
- 30.08.2006: Use of QueueTraffic in a class of visiting female high school students at ETH Zurich.
- 26.10.2006: Use of LogicTraffic in a visiting high school class at ETH Zurich.
- 02.11.2006: Use of LogicTraffic in a visiting high school class at ETH Zurich.
- 14.03.2007: Use of QueueTraffic in a visiting class of female high school students at ETH Zurich.

Teacher Education

- 29.10.2005: Discovery learning workshop using LogicTraffic for high school teachers in Hamburg.
- 13.12.2005: LogicTraffic workshop in a computer science didactics class at the pedagogical university of Bern.

- 02.03.2007: Presentation of InfoTraffic in a continuing education course at the pedagogical institute of Linz.
- 31.03.2007: Presentation and use in a workshop of LogicTraffic and QueueTraffic at the annually "Absolvententreffen der Informatiklehrer" at TU Dresden.

Research Presentations

- 29.09.2005: First public demo of LogicTraffic during a presentation at the GI-INFOS'05 conference in Dresden, Germany.
- 21.08.2006: Presentation of LogicTraffic and QueueTraffic at the international doctoral colloquium of CS didactics in Paderborn, Germany.
- 23.08.2006: Presentation of LogicTraffic and QueueTraffic in the research group for "algorithms, data structures, and applications" at ETH Zurich.
- 07.03.2007: Presentation of InfoTraffic in the doctoral consortium at the ACM SIGCSE technical symposium in Covington KY, USA.
- 08.03.2007: Presentation of LogicTraffic and QueueTraffic at the ACM SIGCSE technical symposium in Covington, Kentucky KY, USA.
- 31.03.2007: Presentation of InfoTraffic at the international doctoral colloquium of CS didactics at ETH Zurich.
- 27.06.2007: Demonstration of InfoTraffic at the ACM ITiCSE conference in Dundee, Scotland.

B System Design and Implementation Issues

This appendix covers details of the implementation of InfoTraffic. We first give an overview of the system and then present selected interesting algorithms. Finally we show some of the data formats of InfoTraffic and list software libraries we have used.

B.1 Acknowledgements

The author of this thesis was the project leader and wrote most of the LogicTraffic code. QueueTraffic was mainly implemented by Nicolas Born as part of his master's thesis [19]. DynaTraffic was implemented by Anna-Nina Simonetto as part of her master's thesis [71]. The following students additionally contributed to InfoTraffic as part of their semester's thesis: Marc Bühler, Hasan Karahan, Anna-Nina Simonetto, and Xiaoping Yin.

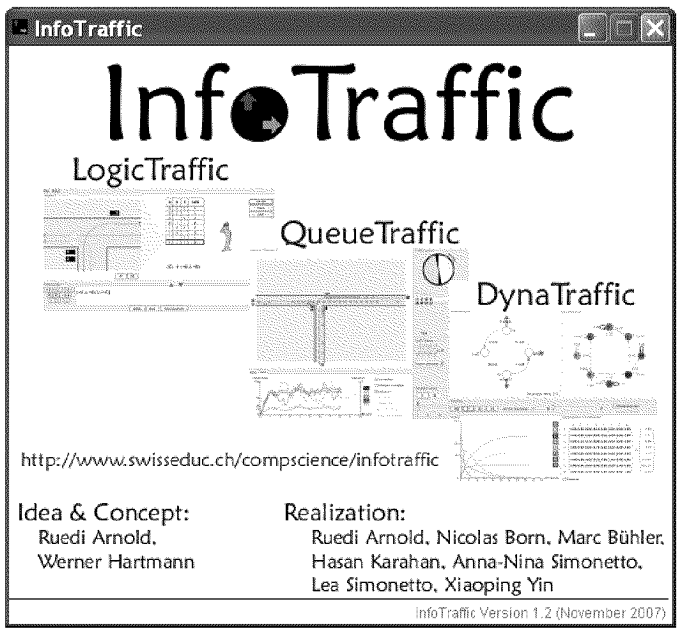

Figure B.1: The start window of InfoTraffic

Figure B.1 shows the start window of InfoTraffic where the user can choose which of the three environments to start by clicking on the corresponding image.

B.2 Overall System Architecture

InfoTraffic is written in the programming language Java. Java was chosen because it is platform independent, an important issue for use in schools. The software consists of about 26000 lines of code split up in 186 classes. These classes are grouped again into 36 packages (figure B.2).

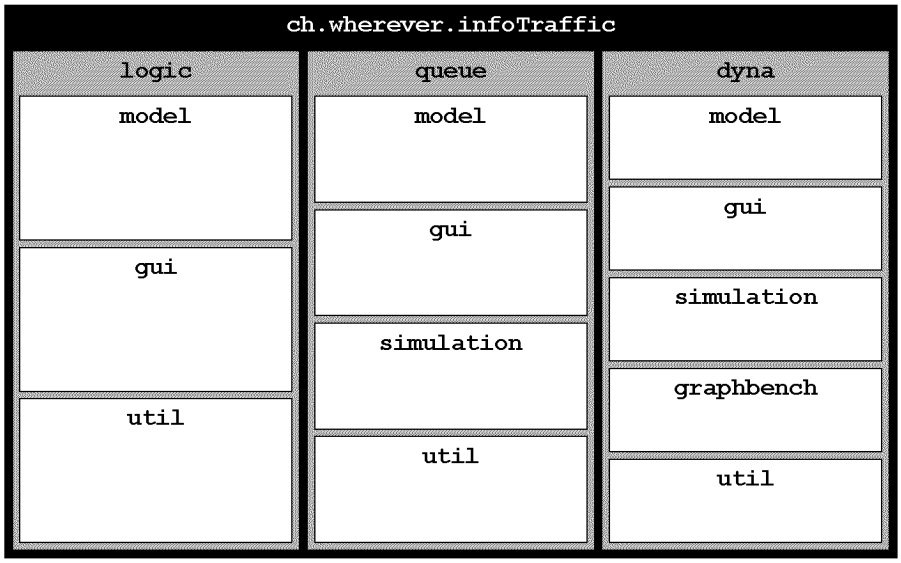

Figure B.2: The first three levels of the InfoTraffic package hierarchy

When InfoTraffic is started, the `main` method of the class `ch.wherever.infoTraffic.InfoTrafficApplication` is invoked. According to the user's choice, one of the three subprograms is started by invoking the `main` method of one of the following classes:

- `ch.wherever.infoTraffic.logic.LogicTraffic`
- `ch.wherever.infoTraffic.queue.QueueTraffic`
- `ch.wherever.infoTraffic.dyna.DynaTraffic`

The three programs are placed in different packages, as can be seen in figure B.2. LogicTraffic and QueueTraffic share code to load, display, and simulate traffic situations, as they use the same data format for

specifying these traffic situations (see section B.4). The two programs also share some traffic situation files, i.e., they display traffic intersections based on the same situation data. DynaTraffic does not share code with LogicTraffic and QueueTraffic.

InfoTraffic uses the Java Swing toolkit and follows in general the *Model-View-Controller* (MVC) and *Observer* design pattern [40]. Figure B.3 shows an UML class diagram of LogicTraffic implementing the MVC design pattern. QueueTraffic and DynaTraffic use analogous classes.

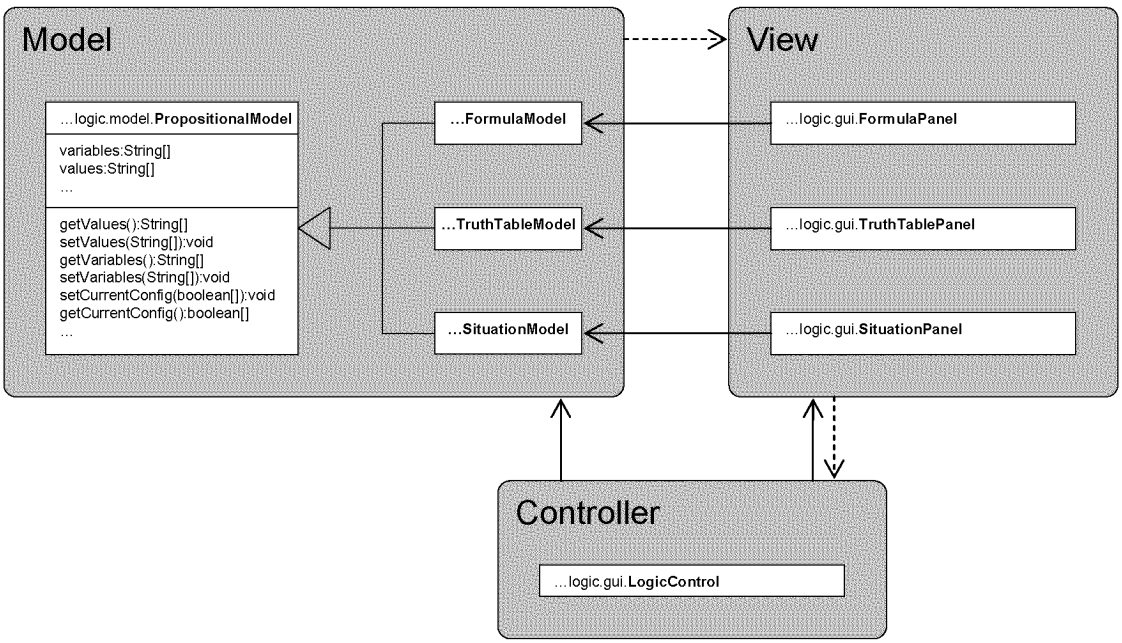

Figure B.3: LogicTraffic implementing the *Model-View-Controller* design pattern

For learning environments the visual appearance is important. We followed the design principles as presented by Mayer [56], especially the redundancy principle, the spatial contiguity principle, and the coherence principle when designing the *graphical user interface* (GUI) of InfoTraffic.

InfoTraffic configuration is implemented with the help of the `java.util.Properties` framework. Parameters and settings such as colors and sizes of GUI elements, the default situation at startup, or the simulation speed are stored in so-called property files. General properties of InfoTraffic like the language or the version of the program are stored in the main property file `data/logicConfig.properties`.

InfoTraffic currently provides a German and an English version. All language specific elements like labels or messages are specified in pro-

perty files. The English texts for QueueTraffic are for example stored in the property file `queue/util/lang/lang_en.properties` and the German texts accordingly in `lang_de.properties`. Providing additional language only requires the creation of the according language files, e.g., `lang_es.properties` for Spanish and to change the `Language` property in the main property file to `es`. No modification and recompilation of code is needed to support additional languages.

B.3 Selected Algorithms

In this section we present selected interesting algorithms of the InfoTraffic environments in pseudo code.

Choice of Formulas in LogicTraffic

As described in section 3.3, LogicTraffic implements the Quine-McCluskey algorithm [46] to obtain optimized formulas in *conjunctive normal form* (CNF) and *disjunctive normal form* (DNF). To clarify how the `Implication` and `Simplest` forms are generated, we here present their generation algorithms.

Generation of Implication Formula

```
formula = DNF as provided by Quine-McCluskey algorithm
sort clauses in ascending order by number of literals
foreach two clauses A, B of the form A AND B of formula
  replace A AND B with (NOT A) IMPLICATION B
end foreach
foreach clause A of the form NOT(A)
  eliminate NOT outside the clause by applying De Morgan's law
end foreach
```

Generation of Simplest Formula

```
foreach formula provided by the logictraffic
  sumOpsVarFormula = #literals + #operators in clauses +
    2*(#operators between clauses)
end foreach
simplestFormula = formula with min(sumOpsVarFormula)
```

Calculation of Parameters in QueueTraffic Simulations

The calculation of the key parameters of queuing theory in QueueTraffic is done according to the following formulas. The other parameters are calculated analogously, see [19] for details.

Calculation of the Arrival Rate

```
cars = #cars arrived within the last round
arrivalRate = cars / roundTime
```

Calculation of the Average Waiting Time

```
time = sum of waiting times of all cars on lane
averageWaitingTime = time / #cars on lane
```

The Upper Limit Mode in DynaTraffic

The algorithms below describe the action when the upper limit mode in DynaTraffic is invoked. For details see [71].

Set a Lane Closed in Upper Limit Mode

```
foreach column of transitionMatrix except laneToLock
  forach row of column
    if (row == laneToLock) then
      probability := 0
    end if
  end foreach
end foreach
foreach column of transitionMatrix except laneToLock
  columnSum = sum of probabilities of this column
  if (columnSum != 1.0) then
    ratio = 1 / columnSum
    foreach element of column
      probability = currentProbability * ratio
    end foreach
  end if
  if (columnSum == 0.0) then
    cycleEdge = 1.0
  end if
end foreach
```

Reopen a Lane in Upper Limit Mode

```
retrieve original row of transitionmatrix(laneToUnLock)
foreach column of transitionMatrix except laneToUnlock
```

```
  foreach element of column
    probability = probability * ((1 - probability of cyclic edge)
      / sum of all edges except cyclic edge)
  end foreach
end foreach
```

B.4 Data Formats

The different traffic situations of InfoTraffic are specified and stored in XML files. We here list the two most relevant XML specification along with an example of each one.

Traffic Situation Format for LogicTraffic and QueueTraffic

LogicTraffic and QueueTraffic share the same data format for the specification of their traffic situations. In LogicTraffic traffic situations are displayed in the size specified in the XML file. In QueueTraffic the traffic situations are scaled down by factor of three, centered and extended with straight lanes and green space, respectively. The following *Document Type Definition* (DTD) specifies the format of traffic situations in LogicTraffic and QueueTraffic.

DTD for Traffic Situations

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT trafficsimulation (situationName, trafficLogic)>
<!ELEMENT situationName (#PCDATA)>
<!ELEMENT trafficLogic (tracks, roadsides, cars?)>
<!ELEMENT tracks (track*)>
<!ELEMENT track (sections, stopPoint, direction, trafficLight)>
<!ATTLIST track name ID #REQUIRED>
<!ELEMENT stopPoint EMPTY>
<!ATTLIST stopPoint x NMTOKEN #REQUIRED y NMTOKEN #REQUIRED>
<!ELEMENT direction EMPTY>
<!ATTLIST direction
  start (north | south | east | west) #REQUIRED
  end (north | south | east | west) #REQUIRED
>
<!ELEMENT trafficLight EMPTY>
<!ATTLIST trafficLight
  x CDATA #REQUIRED
  y CDATA #REQUIRED
>
<!ELEMENT cars (car*)>
<!ELEMENT roadsides (roadside*)>
```

```
<!ELEMENT roadside (shape)>
<!ELEMENT car (center, length, width, speed?, color?)>
<!ATTLIST car
  carid ID #REQUIRED
>
<!ELEMENT center (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT width (#PCDATA)>
<!ELEMENT speed (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT sections (shape+)>
<!ELEMENT shape EMPTY>
<!ATTLIST shape
  type (line | curve) #REQUIRED
  x1 CDATA #IMPLIED
  y1 CDATA #IMPLIED
  x2 CDATA #IMPLIED
  y2 CDATA #IMPLIED
  x CDATA #IMPLIED
  y CDATA #IMPLIED
  radius CDATA #IMPLIED
  startAngle CDATA #IMPLIED
  endAngle CDATA #IMPLIED
>
```

An Example File

The following XML file specifies situation 3 in LogicTraffic and QueueTraffic, see figure B.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<trafficSituation>
  <situationName>Situation 3</situationName>
  <situationElement>
    <tracks>
      <track name="A">
        <sections>
          <shape type="line" x1="0" y1="152" x2="400" y2="152"/>
        </sections>
        <stopPoint x="140" y="152"></stopPoint>
        <direction start="west" end="east"/>
        <trafficLight x="90" y="210"/>
      </track>
      <track name="B">
        <sections>
          <shape type="line" x1="0" y1="176" x2="146" y2="176"/>
          <shape type="curve" x="146" y="224" radius="48"
            startAngle="0" endAngle="270"/>
```

```
          <shape type="line" x1="194" y1="220" x2="194" y2="300"/>
        </sections>
        <stopPoint x="140" y="176"/>
        <direction start="west" end="south"/>
        <trafficLight x="90" y="240"/>
      </track>
      <track name="C">
        <sections>
          <shape type="line" x1="400" y1="104" x2="260" y2="104"/>
          <shape type="curve" x="260" y="200" radius="96"
            startAngle="180" endAngle="270"/>
          <shape type="line" x1="164" y1="200" x2="164" y2="300"/>
        </sections>
        <stopPoint x="260" y="104"></stopPoint>
        <direction start="east" end="south"/>
        <trafficLight x="270" y="50"/>
      </track>
    </tracks>
    <roadsides>
      <roadside>
        <shape type="line" x1="0" y1="80" x2="400" y2="80"/>
      </roadside>
      <roadside>
        <shape type="line" x1="0" y1="200" x2="140" y2="200"/>
        <shape type="line" x1="140" y1="200" x2="140" y2="300"/>
      </roadside>
      <roadside>
        <shape type="line" x1="260" y1="200" x2="400" y2="200"/>
        <shape type="line" x1="260" y1="200" x2="260" y2="300"/>
      </roadside>
    </roadsides>
  </situationElement>
</trafficSituation>
```

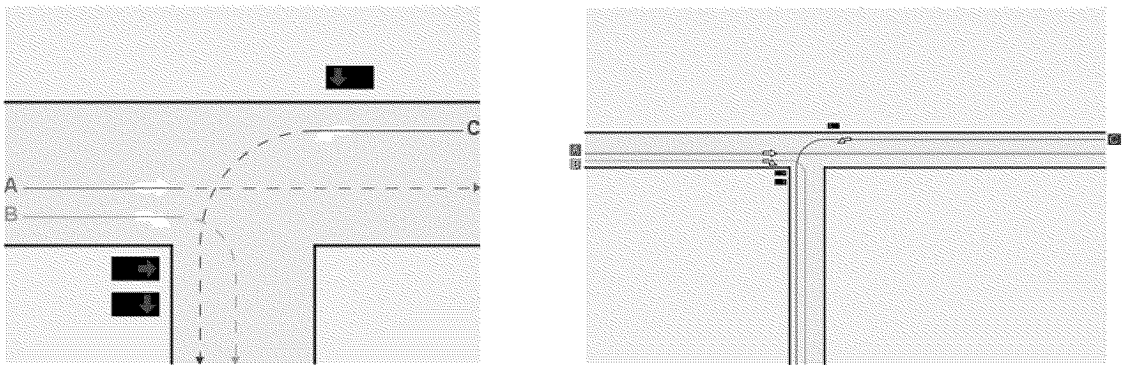

Figure B.4: Situation 3 as displayed in LogicTraffic (left) and QueueTraffic (right)

Graph Format for DynaTraffic

The following XML schema specifies the format of graphs in DynaTraffic. It is an extension of the graph XML schema of GraphBench [22].

XML Schema for Graphs in DynaTraffic

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="XmlNode">
    <xsd:attribute name="node_id" type="xsd:int" use="required"/>
    <xsd:attribute name="index" type="xsd:int" use="required"/>
    <xsd:attribute name="type" type="xsd:int"/>
    <xsd:attribute name="value" type="xsd:double"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="x" type="xsd:int" use="required"/>
    <xsd:attribute name="y" type="xsd:int" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="XmlEdge">
    <xsd:attribute name="edge_id" type="xsd:int" use="required"/>
    <xsd:attribute name="index" type="xsd:int" use="required"/>
    <xsd:attribute name="type" type="xsd:int"/>
    <xsd:attribute name="hasSibling" type="xsd:boolean"
      use="required"/>
    <xsd:attribute name="value" type="xsd:double"/>
    <xsd:attribute name="from" type="xsd:int" use="required"/>
    <xsd:attribute name="to" type="xsd:int" use="required"/>
    <xsd:attribute name="cycleType" type="xsd:int"/>
    <xsd:attribute name="color" type="xsd:int"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="threshold" type="xsd:int"/>
  </xsd:complexType>

  <xsd:complexType name="XmlGraph">
    <xsd:sequence>
    <xsd:element name="Node" type="XmlNode" minOccurs="1"
      maxOccurs="unbounded"/>
    <xsd:element name="Edge" type="XmlEdge" minOccurs="0"
      maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
    <xsd:attribute name="index" type="xsd:int" use="required"/>
    <xsd:attribute name="graph_id" type="xsd:int" use="required"/>
    <xsd:attribute name="value" type="xsd:double"/>
  </xsd:complexType>
</xsd:schema>
```

An Example File

The following XML file specifies situation 1 in DynaTraffic, see figure B.5.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xmlGraph value="0.0" type="standard" index="1" graph_id="1">
  <Node y="150" x="100" type="1" index="0" name="0" node_id="0"/>
  <Node y="150" x="400" type="1" index="1" name="1" node_id="1"/>
  <Edge value="100.0" type="1" to="0" index="0" from="1"
    name="A" edge_id="0" hasSibling="true"/>
  <Edge value="0.0" type="1" to="1" index="1" from="0"
    name="B" edge_id="1" hasSibling="true"/>
</xmlGraph>
```

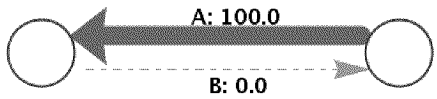

Figure B.5: Situation 1 as displayed in DynaTraffic

B.5 Used Libraries

InfoTraffic employs software libraries and tools that are not part of the standard Java JDK distribution. To implement the LogicTraffic propositional logic parser we applied the JavaCC [33] parser/scanner generator. QueueTraffic uses the TableLayout [34] layout manager. DynaTraffic finally utilizes and extends the GraphBench API [22] for handling graphs and the Jama [54] Java matrix package for handling matrices and vectors.

Bibliography

[1] J. R. Anderson, L. M. Reder, and H. A. Simon. Situated Learning and Education. *Educational Researcher*, 25(4):5–11, Mai 1996.

[2] R. Arnold. Demonstration abstract: Introducing Propositional Logic and Queueing Theory with the InfoTraffic Interactive Learning Environments. In *Proceedings of ACM ITiCSE 2007*, Dundee, Scotland, June 2007.

[3] R. Arnold. InfoTraffic: Interactive Learning Environments, with Teaching Material, German Edition. `http://swisseduc.ch/informatik/infotraffic`, last visited September 2007.

[4] R. Arnold and W. Hartmann. LogicTraffic – Logik in der Allgemeinbildung. *Hauptbeitrag Informatik-Spektrum*, 30(1):19–26, 2 2007.

[5] R. Arnold and W. Hartmann. Pragmatische Empfehlungen zur Entwicklung von interaktiven Lernumgebungen. In *Proceedings INFOS 2007, 12. GI-Fachtagung Informatik und Schule*, Siegen, Germany, September 2007.

[6] R. Arnold, W. Hartmann, and R. Reichert. Entdeckendes Lernen im Informatik-Unterricht. In *Proceedings of INFOS 2005, 11. GI-Fachtagung Informatik und Schule*, pages 197–205, Dresden, Germany, September 2005.

[7] R. Arnold, M. Langheinrich, and W. Hartmann. InfoTraffic - Teaching Important Concepts of Computer Science and Math through Real-World Examples. In *Proceedings ACM SIGCSE Technical Symposium*, pages 105–109, Covington, Kentucky, USA, March 2007.

[8] D. P. Ausubel. The use of advance organizers in the learning and retention of meaningful verbal material. *Journal of Educational Psychology*, 51:267–272, 1960.

[9] J. Barwise and J. Etchemendy. *Language, Proof and Logic.* CSLI Publications, Stanford, 1999.

[10] M. Ben-Ari. Constructivism in computer science education. In *SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, pages 257–261, New York, NY, USA, 1998. ACM Press.

[11] G. A. Berg. The Big Questions. *International Journal on E-Learning*, 1(2):5–6, 2002.

[12] D.C. Berliner. Educational Research: The Hardest Science of All. *Educational Researcher*, 31(8):18–20, 2002.

[13] D. Bertsekas and R. Gallager. *Data Networks, second edition.* Prentice-Hall Int., New Jersey, 1992.

[14] J. Bewersdorff. *Glück, Logik und Bluff. Mathematik im Spiel – Methoden, Ergebnisse und Grenzen.* Vieweg, Wiesbaden, 1998.

[15] J. Bewersdorff. Monopoly im Blickwinkel der Mathematik. `http://www.bewersdorff-online.de/monopoly/`, last visited September 2007.

[16] D. A. Bligh. *What is the Use of Lectures?* Penguin Books, Harmondsworth, England, 1972.

[17] B. S. Bloom, M. B. Engelhart, E. J. Furst, W. H. Hill, and D. R. Kratwohl. *Taxonomy of educational objectives. Handbook 1: The cognitive domain.* Longsmans Green, London, 1956.

[18] A. Blumstengel. *Entwicklung hypermedialer Lernsysteme.* Wissenschaftlicher Verlag, Berlin, 1998.

[19] N. Born. *QueueTraffic – Eine Lernumgebung rund um die Warteschlangentheorie.* Masterarbeit ETH Zürich, 2006.

[20] B. S. Borowski. Truth Table Constructor Applet. `http://www.brian-borowski.com/Truth/TruthTableConstructor.html`, last visited September 2007.

[21] M. Brändle. *GraphBench: Exploring the Limits of Complexity with Educational Software.* Dissertation Nr. 16392, ETH Zurich, 2006.

[22] M. Brändle. GraphBench API on SwissEduc. `http://www.swisseduc.ch/informatik/graphbench/programming.html`, last visited September 2007.

[23] M. Brändle and J. Nievergelt. Tackling complexity: A case study on educational software. *World Conference on E-Learning in Corp., Govt., Health., and Higher Ed. (ELEARN), Volume 2004, Issue 1*, pages 1794–1799, 2004.

[24] J. D. Bransford, R. D. Sherwood, T. S. Hasselbring, C. K. Kinzer, and S. M. Williams. Anchored Instructions: Why we need it and how technology can help. In D. Nix and R. Spiro, editors, *Cognition, Education and Multimedia: Exploring ideas in high technology*, pages 163–205. Erlbaum, Hillsdale, NJ, 1990.

[25] T. Brinda. *Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sek. II.* PhD thesis, Universität Siegen, Siegen, 2004.

[26] J. S. Bruner. *The Process of Education.* Harvard University Press, 1960.

[27] J. S. Bruner, R. R. Oliver, and P. M Greenfield. *Studies in Cognitive Growth.* John Wiley and Sons, New York, 1966.

[28] H. Burkhardt and A. H. Schoenfeld. Improving Educational Research: Toward a More Useful, More Influential, and Better-Funded Enterprise. *Educational Researcher*, 32(9):3–14, 2003.

[29] D. Carlson, M. Guzdial, C. Kehoe, V. Shah, and J. Stasko. WWW interactive learning environments for computer science education. In *SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education*, pages 290–294, New York, NY, USA, 1996. ACM Press.

[30] J. Christen. Department of Traffic Control, City Police Zurich. Personal Communication, March 2005.

[31] J. Clark and D. A. Holton. *Graphentheorie. Grundlagen und Anwendungen.* Spektrum Akademischer Verlag, 1994.

[32] R. E. Clark. Media Will Never Influence Learning. *Educational Technology Research and Development*, 42(2):21–29, 1994.

[33] CollabNet. JavaCC: A Parser/Scanner Generator for Java. `https://javacc.dev.java.net/`, last visited September 2007.

[34] CollabNet. TableLayout: A Java Layout Manager. `https://tablelayout.dev.java.net/`, last visited September 2007.

[35] T. Crews, G. Biswas, S. Goldman, and J. Bransford. Anchored interactive learning environments. *International Journal of AI in Education*, 8, 1997.

[36] P. J. Denning. Great principles of computing. *Communications of the ACM*, 46(11):15–20, 2003.

[37] Design-Based Research Collective. Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher*, 32(1):5–8, 2003.

[38] A. Engel. *Wahrscheinlichkeitsrechnung und Statistik, Band 2.* Ernst Klett, Stuttgart, 1978.

[39] F. Fischer, M. Waibel, and C. Wecker. Nutzenorientierte Forschung im Bildungsbereich: Argumente einer internationalen Diskussion. *Zeitschrift für Erziehungswissenschaft*, 8(3):427–442, 2005.

[40] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley, Massachusetts, 1994.

[41] M. Guzdial and E. Soloway. Teaching the Nintendo Generation to Program. *Communications of the ACM*, 45(4):17–21, 2002.

[42] W. Hartmann, M. Näf, and R. Reichert. *Informatikunterricht planen und durchführen.* Springer, Berlin, 2006.

[43] P. J. Hurley. *A Consice Introduction to Logic.* Wadswoth / Thomson Learning, eight edition, 2003.

[44] ISEE Systems. Stella. Systems Thinking Software for Education and Research. `http://www.iseesystems.com/`, last visited September 2007.

[45] B. J. Jansen, A. Spink, and T. Saracevic. Failure analysis in query construction: Data and analysis from a large sample of web queries.

In *Proceedings of the 3rd ACM International Conference on Digital Libraries*, pages 289–290, Pittsburgh, USA, 1998. ACM.

[46] R. Katz. *Contemporary Logic Design.* Benjamin-Cummings, Redwood City, California, 1994.

[47] A. C. Kay. Computers, Networks and Education. *Scientific American. Special Issue 3(265)*, pages 100–107, 1991.

[48] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, Princeton and Oxford, 2006.

[49] B. Laurel. *Computers as Theatre.* Addison-Wesley Longman, 1993.

[50] H.P. Lindenmann. Netzmodelle und Simulation (Teil 2). Vorlesungsunterlagen Institut für Verkehrsplanung und Transportsysteme, ETH Zürich, September 2003.

[51] H. Mandl and B. Kopp. Blended Learning: Forschungsfragen und Perspektiven. *Research report No. 182, Ludwig-Maximilians-Universität München, Lehrstuhl für Empirische Pädagogik und Pädagogische Psychologie*, 2006.

[52] MathDemos Projects. Animated Matrix-Vector Multiplication. `http://mathdemos.gcsu.edu/mathdemos/matvec/matvec.html`, last visited September 2007.

[53] EPA Mathematik. *Einheitliche Prüfungsanforderungen in der Abiturprüfung Mathematik.* Beschluss der Kultusministerkonferenz vom 01.12.1989 i.d.F. vom 24.05.2002. Siehe auch `http://www.kmk.org/doc/beschl/EPA-Mathematik.pdf`, last visited September 2007.

[54] MathWorks, NIST. JAMA: A Java Matrix Package. `http://math.nist.gov/javanumerics/jama/`, last visited September 2007.

[55] F. Mattern. Hundert Jahre Zukunft: Visionen zum Computer- und Informationszeitalter. In F. Mattern, editor, *Die Informatisierung des Alltags – Leben in smarten Umgebungen.* Springer, Berlin Heidelberg New York, 2007.

[56] R. Mayer. *Multimedia Learning.* Cambridge University Press, 2001.

[57] W. J. Meyer. *Concepts of Mathematical Modeling.* McGraw-Hill Book Company, 1984.

[58] J. Nievergelt. Interactive Systems for Education: The New Look of CAI. *Proceedings IFIP Conference on Computers in Education*, 53, No. 4:465–472, 1975.

[59] J. Piaget. *Das Erwachen der Intelligenz beim Kinde.* Klett-Cotta, Stuttgart, 3 edition, 1991.

[60] PTV Planung Transport Verkehr AG. VISSIM. `http://www.ptv.de/`, last visited September 2007.

[61] R. Reichert. *Theory of Computation as a Vehicle for Teaching Fundamental Concepts of Computer Science.* Dissertation Nr. 15035, ETH Zurich, 2003.

[62] R. Reichert and W. Hartmann. On the Learning in E-Learning. In *Proceedings of EDMEDIA 2004 - World Conference on Education Multimedia, Hypermedia and Telecommunications*, pages 1590–1595, Lugano, Switzerland, June 2004.

[63] R. Reichert, J. Nievergelt, and W. Hartmann. *Programmieren mit Kara. Ein spielerischer Zugang zur Informatik, (ergänzte Neuauflage).* Springer, Berlin, Dezember 2004.

[64] G. Reinmann. Nur "Forschung danach"? Vom faktischen und potentiellen Beitrag der Forschung zu alltagstauglichen Innovationen beim E-Learning. *Arbeitsbericht Universität Augsburg*, Nr. 14, 2006.

[65] M. Schaefer. Crossroad – Simple Traffic Simulation. `http://paginas.fe.up.pt/~eol/schaefer/crossroadApplet/index.htm`, last visited September 2007.

[66] T. Schickinger and A. Steger. *Diskrete Strukturen. Band 2: Wahrscheinlichkeitstheorie und Statistik.* Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[67] U. Schöning. *Logik für Informatiker.* Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford, 1995.

[68] R. Schulmeister. Taxonomy of Multimedia Component Interactivity. A Contribution to the Current Metadata Debate. *Studies in*

Communication Sciences. Studi di scienze della communicazione., 3(1):61–80, 2003.

[69] R. Schulmeister. *Grundlagen hypermedialer Lernsysteme: Theorie - Didaktik - Design, 4., überarbeitete und aktualisierte Auflage.* Oldenbourg Wissenschaftsverlag, 2007.

[70] A. Schwill. Fundamental ideas of computer science. *EATCS-Bulletin*, 53:274–295, 1994.

[71] A. Simonetto. *DynaTraffic – Eine Lernumgebung zu Markov-Ketten.* Masterarbeit ETH Zürich, 2007.

[72] C. Snyder. *Paper Prototyping - The Fast and Easy Way to Design and Refine User Interfaces.* Elsevier Science, 2003.

[73] Software Systems Ampel and Knobel. BPS GmbH. `http://www.bps-verkehr.de/`, last visited September 2007.

[74] D. E. Stokes. *Pasteur's Quadrant. Basic Science and Technological Innovation.* Brookings Institution Press, Washington DC, 1997.

[75] M. Treiber. Microsimulation of road traffic, online Applet. `http://vwisb7.vkw.tu-dresden.de/~treiber/MicroApplet/`, last visited September 2007.

[76] E. Trichina. Didactic instructional tool for topics in computer science. In *ITiCSE '99: Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education*, pages 95–98, New York, NY, USA, 1999. ACM Press.

[77] V. Tscherter. *Exorciser: Automatic Generation and Interactive Grading of Structured Exercises in the Theory of Computation.* Dissertation Nr. 15654, ETH Zurich, 2004.

[78] University of Berne. The Logics Workbench. `http://www.lwb.unibe.ch`, last visited September 2007.

[79] University of Berne. ViLoLa - A Virtual Logic Laboratory. `http://www.vilola.unibe.ch`, last visited September 2007.

[80] R. Wattenhofer. *Computer Networks, Chapter 3.* Compulsory course for 2nd year computer science students, ETH Zurich, 2007.

[81] H. Wedekind, R. Inhetveen, and E. Ortner. Informatik als Grundbildung - Teil VI: Logik und Geltungssicherung. *Informatik-Spektrum*, 28(1):48–52, 2005.

[82] D. B. West. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, Upper Saddle River, 2001.

[83] S. Wolf. *Logic*. Compulsory course for 1st year computer science students, ETH Zurich, 2006.

[84] P. Wülser. *Unterricht fürs Ohr - Podcasting in der Schule*. Unveröffentlichtes Manuskript, 2006.

Curriculum Vitae

Ruedi Arnold

Personal Data

June 23, 1976	Date of birth
Bürglen UR	Swiss citizenship

Education

1989–1996	Gymnasium in Altdorf UR, Switzerland
1996	Matura, type C
1996–1997	Serving the Swiss army. Working at UBS in Zurich, Switzerland. Language courses and traveling in the USA
1997–2002	Studies of computer science, ETH Zurich, Zurich, Switzerland
1999–2000	Exchange year, Strathclyde University, Glasgow, United Kingdom
2002	Master's degree in computer science (Dipl. Informatik Ing. ETH)
2000–2003	Studies of education sciences, ETH Zurich
2003	Didaktischer Ausweis ETH in computer science
2002–2007	Ph. D. studies in computer science, ETH Zurich Advisor: Prof. Dr. Friedemann Mattern Co-referee: Prof. Dr. Werner Hartmann Co-referee: Prof. Dr. Carl August Zehnder