Explaining behaviour: using qualitative simulation in interactive learning environments

Bouwer, A.J.

Publication date
2005

Document Version
Final published version

Link to publication

Citation for published version (APA):
Bouwer, A. J. (2005). *Explaining behaviour: using qualitative simulation in interactive learning environments*. [Thesis, fully internal, Universiteit van Amsterdam]. SIKS.

Explaining Behaviour

Using Qualitative Simulation in
Interactive Learning Environments

Explaining Behaviour

Using Qualitative Simulation in
Interactive Learning Environments

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. mr. P. F. van der Heijden
ten overstaan van een door het college
voor promoties ingestelde commissie,
in het openbaar te verdedigen
in de Aula der Universiteit
op woensdag 6 juli 2005, te 10.00 uur

door

Anders Jan Bouwer

geboren te Dordrecht

Promotiecommissie

Promotores:	Prof. dr. B. J. Wielinga
	Prof. dr. J. A. P. J. Breuker
Co-promotor:	Dr. B. Bredeweg
Overige leden:	Prof. dr. P. Brna
	Prof. dr. A. L. Ellermeijer
	Prof. dr. K. D. Forbus
	Prof. dr. ir. R. J. H. Scha
	Prof. dr. J. L. Top
	Dr. R. G. F. Winkels

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

SIKS Dissertation Series No. 2005-10

Printed by Ponsen & Looijen, Wageningen
ISBN 90-6464-770-4

Contents

Acknowledgements

To everybody who was waiting for this thesis to be finished, thank you for your patience! Many of you deserve some credit for this little feat, which I try to acknowledge here.

First of all, I want to thank Bert for his support and constructive criticism throughout the years. I admire your endless drive in making QR work, and your enthusiasm in communicating knowledge. I have come to respect your stubbornness as well (or was it mine?). Bob, thank you for hiring me on several occasions, also when times were tough, and your ability to pinpoint the weak spots in my writing. Joost, thank you for your expertise. I enjoyed the academic blizzards when brainstorming with you.

Thanks to the members of the promotion committee for proofreading my thesis, and to numerous reviewers in the QR and AIED communities for their comments on my work at various stages of development. Special thanks go out to Ken Forbus in this respect. I take full responsibility for any remaining suboptimalities...

Thanks to Floor Goddijn for the design and implementation of the QUAGS question generation module. Thanks to Peggy Tjaris for carrying out the VisiGarp evaluation studies. Thanks to the Master Meetings and JvT people for their feedback and enthusiasm in related projects. Thanks to the participants in the VisiGarp evaluation studies (and other users) for their comments.

Thanks to Vania, my academic *twin sister*, for her friendliness, as well as for helping out with the layout, and showing me Brazil for the first time. Thanks to Roland for being sharp, discussing all things graphical, and solving some of my teXiest problems. Thanks to Paulo Salles for building the example ecological models, thoroughly testing VisiGarp, and showing me more of Brazil.

Thanks to my SWI/HCS-colleagues over the years for being so friendly. I am happy to have (had) very pleasant roommates: Richard, thanks for the chair! Anne-Marie, it was nice to hear you whistle those Charlie Parker solos. Jan Wielemaker, I was amazed how you could solve many of my prolog programming problems as fast as opening a bottle. Noor, *wonderful person extraordinaire*, thanks for the map–scientific quests may not be unlike off-the-road travelling after all...

To the people who finished before me (Floortje, Rens, Yuri, Jos, Hedderik, among others) and those who will follow shortly (you know who you are!), thanks for sharing tips, ideas, highlights and obstacles along the way–I enjoyed your company.

Thanks to Gaston and Patrick for being my *paranymphs*. Gaston, I very much appreciated your effort in creating (an awareness of the importance of) the social context on many Friday evenings. That certainly helped to put problems into perspective. Pat, thanks for being my best friend since elementary school!

Many friends helped me to keep a welcome balance between work and play. I want to thank you all for the time and energy you invested in me, especially when I didn't have much of it. To those with musical instruments, thanks for sharing the fun(k)! To those with wheels on their feet (my fellow Survivors and Hot Wheels, among others), thanks for sharing the floor! Robert and Que, in particular, thanks for your support in difficult times.

Thanks to my family (Bouwer, Schenkel, etc.) for being there, especially during the last two years. Rob, thanks for your work on the cover design, and the many tasty meals. Finally, thanks to Jopi (in memoriam) for naming me Anders, and always believing in me.

1 Introduction

"Common sense is not so common"

Voltaire (1694–1778), French author and philosopher

One of the things that makes us human is the desire to understand the world around us. This has led to the development of science, which has evolved into different disciplines, or domains, such as physics, biology and ecology. Each of these domains includes theories and models which predict or explain certain phenomena, such as the behaviour of an apple falling from a tree, the working of the heart in the blood circulation, and the effects of fire management on a vegetation, respectively. While each domain has its own laws, often expressed as mathematical formulae, every domain requires *conceptual understanding*, which is usually expressed in a qualitative, more general way. Explanations are intended to support people in building such a conceptual understanding. Producing good explanations requires a great deal of effort and expertise, however. Explaining dynamic phenomena is particularly hard, because it involves describing how a system changes over time. The expertise required to generate explanations about behaviour includes knowledge about the structure of the system under investigation, knowledge of the variables of interest and how they relate to each other, and knowledge of the processes that apply to the situation at hand, and how they change the state of the system. These kinds of knowledge closely match the knowledge represented in qualitative models. Qualitative reasoning (QR) is an area of research, combining artificial intelligence and cognitive science, which deals with such models, and how to reason with them [Weld and de Kleer, 1990, Bredeweg and Struss, 2003]. A qualitative model can be used to generate predictions given an initial situation. This process, as well as its result, is called *qualitative simulation* [Kuipers, 1986, Bredeweg, 1992]. Because qualitative simulations explicitly represent system behaviour, they are regarded as a good basis for *interactive learning environments* [Winkels and Bredeweg, 1998, Bredeweg and Forbus, 2003]. These are computer software systems which provide interactive access to the knowledge they contain, which allows people to learn interactively, in absence of a human expert to talk to.

This thesis addresses interactive learning environments for learning about dynamic phenomena, and focusses on how information from qualitative simulations can be used to generate interactive explanations. The following sections discuss the background in the philosophy of explanation, interactive learning environments, qualitative simulation in education, and explanation generation and visualization.

1.1 The Philosophy of Explanation

Explanations are supposed to clarify, but the term itself needs some clarification as well, as it is used in many ways in the literature. The philosophy of explanation concerns itself with the question 'what is explanation?'. This involves thinking about what can be explained (the *explanandum*), and what may explain it (the *explanans*, or actual explanation). With respect to the explanandum, the notion of *event* is important, as behaviour can be seen as a collection of events. According to Hempel's theory of explanation, an event is explained 'when we can show that it was, in the circumstances, to be expected' [Hempel, 1993]. In other words, there is a symmetry between explanation and prediction. There is some disagreement among philosophers about which, and how much information should be included in the explanans of an explanation [Ruben, 1993].

There is no controversy about the importance of causal relations for explanations, an idea which dates back at least to Aristotle, although the exact role which it plays varies among researchers [Salmon, 1993]. Laplace regards 'the present state of the universe as the effect of its anterior state and as the cause of the one which is to follow' [1951, p. 4]. Achinstein [1993] argues that there are also explanations which use other kinds of relations, such as *identity* relations. For example, one can explain why ice is water on the grounds that it is H_2O. Kim further adds 'dependency or determination relations, along with temporal and spatial ones' [Kim, 1974, cited by Ruben [1993], p. 12]. Hempel [1993] stresses the need for general laws or principles to be mentioned in an explanation, which are tied to the explanatory facts or events in the particular situation. Lewis [1993] defines a *full explanation* of an event as including a complete causal history, ultimately stretching back to the big bang, but realizes that this is an idealization; in real-life, explanations only refer to parts of this causal history. Van Fraassen also takes a pragmatic view of explanation as a process, by noting that 'an explanation is an answer to a why-question' in a particular context [1993, p. 287]. The context may refine a question from 'Why this' to 'Why this rather than that', leading to the notion of *contrastive explanation* [Lipton, 1993]. To choose an appropriate contrast, he proposes the use of the *difference condition*: 'to explain why P rather than Q, we must use a causal difference between P and not-Q, consisting of a cause of P and the absence of a corresponding event in the history of not-Q' [Lipton, 1993, p. 217]. Achinstein [1993] also includes who-, what-, when-, and where-questions, thereby broadening the scope of explanation to answering any question that attempts to gain understanding.

In this thesis, the topic is how to explain the behaviour of systems automatically, for different domains. This amounts to answering questions such as what happens, when, and why. Although the philosophical literature referred to above includes requirements for what to include in an explanation, it does not adequately specify how an explanation can be generated automatically.

1.2 Interactive Learning Environments

Computers have always been regarded as powerful tools for educational purposes, because they allow a high degree of interaction with large libraries of domain materials. However, traditional ways of using computers in educational settings are often limited to supporting the creation, sharing and presentation of information in forms which have been available for a long time as printed media. The research area called *artificial intelligence in education* aims to unleash the full potential of computers in education, by providing a richer, and more personalized experience for learners (and sometimes, teachers). This is done by (1) using more articulate representations of the domain knowledge, so that the computer can reason about the knowledge it incorporates, besides merely presenting it, (2) encoding didactical knowledge, so that the computer can reason about how it should communicate the appropriate information, (3) including capabilities to model learner behaviour, for the purpose of monitoring, diagnosing, and curriculum planning, and (4) providing an adaptive interface, which may include capabilities for language processing or graphical communication. A system which incorporates these characteristics is called an *intelligent tutoring system* (ITS, *e.g.*, see Sleeman and Brown [1982], Wenger [1987]), or *interactive learning environment* (ILE, *e.g.*, see Lajoie and Vivet [1999]).

ITS have been shown to accomplish learning benefits by using strategies inspired by human tutors [Corbett and Anderson, 1995, Graesser et al., 1999]. An important didactic strategy employed in ITS is *model tracing*. This entails close monitoring of a learner's behaviour by comparing it to a correct model of reasoning, and remediating as soon as a deviation occurs [Anderson et al., 1990, Corbett and Anderson, 1995]. Other examples of strategies employed in ITS are socratic tutoring [Collins and Stevens, 1991], help, and coaching [Winkels, 1992].

Moving away from the idea that the system should be in control (as is the case with many ITS), ILE reflect a more open view on learning, where students are more responsible for their own learning. This is exemplified by the strategy of (collaborative) scientific inquiry [Suthers et al., 1997, Joolingen, 1998], which is rooted in discovery learning [Bruner, 1961]. Scientific inquiry entails a learning process consisting of activities for the learner such as postulating hypotheses, making predictions, performing experiments, evaluating results, and generating explanations.

Curriculum planning involves the choice and sequencing of topics and exercises. To this end, Goldstein [1982] defines a *genetic graph* of the domain material, incorporating four evolutionary relations between pieces of knowledge (explanation, generalization, analogy, and refinement). These relations are used to select the next topic, thereby extending the learner's knowledge frontier. In the context of electronics, White and Frederiksen [1990] specify how learning is supported in their system QUEST, by progressing along a sequence of models of increasing complexity, according to a number of dimensions. A related, but more flexible approach is the didactic goal generator, which uses a domain knowledge representation to generate links (allowing possible learning routes) automatically [Winkels and Breuker, 1993]. A problem with most current approaches to curriculum planning is that they are either tied to a specific domain, or do not account for the complexity of qualitative simulations. It is not clear how didactic goals and links should be defined in the context of qualitative simulations, although some suggestions are given by Salles and Bredeweg [2001a].

In this thesis, the goal is to develop an interactive learning environment based on qualitative simulations, which can exploit the benefits of different didactic strategies, ranging from student-initiated exploration, to system-directed tutoring.

1.3 Qualitative Simulation in Education

One of the earliest examples of the use of qualitative simulations in computer-based learning is the Meteorology Tutor [Brown et al., 1973]. It used an inference tree based on qualitative states and transitions to generate answers to questions about meteorological processes, such as precipitation and evaporation. Brown et al. continued their efforts, which resulted in the SOPHIE systems. SOPHIE I & II included numerical simulation capabilities to create an artificial lab, or *reactive learning environment*, in which a learner can perform experiments safely and easily, and receive informed feedback, for the domain of troubleshooting of electronic circuits [Brown et al., 1982]. With SOPHIE III, its designers incorporated a *qualitative* simulator, in an attempt to move towards more humanlike reasoning and explanation capabilities. A remarkable feature of the SOPHIE systems was the robust natural language interface, which could handle a broad range of queries from a user. A similar influential project involving interactive simulations is the STEAMER project [Hollan et al., 1984, 1987]. Whereas the SOPHIE systems used natural language dialogue as the most important medium of communication, the STEAMER system tried to provide a *continuous explanation* of the domain (a steam propulsion plant) in the form of animated icons. Interestingly, while the simulation was driven by a mathematical model, the graphical representations were (qualitative) abstractions that were conceptually faithful to an expert's mental model.

These early projects spawned much work on qualitative reasoning, such as Envision theory [de Kleer and Brown, 1983, 1984], and Qualitative Process Theory (QPT) [Forbus, 1984, 1988]. Qualitative simulations have great potential for education and training, because they show which behaviour can be expected, as well as how this behaviour is linked to the initial situation, using relevant knowledge about the domain and generic rules of prediction. To make optimal use of this potential, an articulate representation is necessary, in which the role of all different bits of information is made explicit [Forbus, 1990, 1997, Bredeweg and Forbus, 2003]. The QPT framework includes such an articulate knowledge representation and has been employed in educational software of various kinds. First, in the form of the self-explanatory simulation software, which incorporates simulations of liquids cooling and evaporating from cups of different materials to help students understand the principles of thermodynamics [Forbus, 1996a]. Second, in the form of the CyclePad Articulate Virtual Laboratory software, which supports learning about thermodynamical engineering by the design of thermodynamic cycles [Forbus et al., 1999]. Third, in the form of VModel, a qualitative modelling environment for middle-school students which includes qualitative simulation within one state [Forbus et al., 2001]. Related work concerns an educational tool based on QPT aimed at secondary school level chemistry, which supports learning of chemistry principles as well as laboratory procedures [Mustapha et al., 2002].

Another example of a qualitative reasoning framework which includes such an articulate representation, is GARP [Bredeweg, 1992]. The GARP framework includes (1) a simulation engine, with access to the model and results via a command-line interface, (2) a number of domain libraries with scenarios and model fragments about various situations and processes, and (3) model-building tools, such as MOBUM [Bessa Machado, 2004, Groen, 2003, Bouwer et al., 2002] and HOMER [Jellema, 2000, Bessa Machado and Bredeweg, 2002, 2003].

Despite the fact that qualitative simulations are often meant to be intuitive models of behaviour, they can be complex, because they incorporate many states, and/or because there is a great amount of information specified within a state. Therefore, simplifying and communicating simulation results is considered a major requirement for fulfilling the potential of qualitative reasoning [Mallory, 1998, de Koning et al., 2000]. Another important consideration is that learners need adequate support while using computer simulations, to guarantee a useful learning experience [van der Hulst, 1996]. The SimQuest software, which incorporates simulations based on quantitative models, offers support in the form of visualizations of simulation results in graphs, a hypothesis scratchpad, and a monitoring tool, which gives feedback on the choice of experiments that a learner has made [van Joolingen, 1999]. Another advantage of the SimQuest software is that its modular architecture can be (and has been) adapted to work with different domain modules.

The work described in this thesis combines strengths of these various approaches, including the use of simulations for education, the use of qualitative models to simulate system behaviour and generate causal explanations, the abstraction of simulation results for communication, and domain-independence.

1.4 Explanation Generation and Visualization

Explanation requires a medium of communication, such as natural language in speech, written or typed form, or some kind of visualization. Traditionally, an explanation is considered to be an exposition or interactive dialogue in natural language, and much work on explanation generation reflects this [McKeown, 1985, Woolf and Murray, 1987, Acker et al., 1991, Winkels, 1992, Suthers, 1992, Moore, 1995]. There are good reasons for using textual representations, as it is a fast medium for communicating propositional information which can be presented sequentially [Arens et al., 1993]. Natural language dialogue is a good medium for abstraction, and for checking and remediating misunderstandings [Collins and Stevens, 1991, Cawsey, 1993, Fox, 1993]. Questions play an important role, in this respect, from both teachers [Sinclair and Coulthard, 1975, Collins and Stevens, 1991] and learners [Pilkington, 1992, Moore, 1996]. However, the preference for text as a medium for explanation may also have been rooted in social and technological developments; with the arrival of modern printing, computer, and networking technology it has become much easier to produce and disseminate graphical material during the last few decades, and there is an increasing popularity of graphics to communicate information [Tufte, 1990, 1997, Engelhardt, 2002]. Now, information visualization and diagrammatic reasoning are active areas of research. Information visualization deals with issues related to the automatic presentation of information graphically (e.g., see Roth et al. [1997]), while diagrammatic reasoning focuses on the logical aspects of graphical representations in human and computer reasoning [Kulpa, 1994, Glasgow et al., 1995]. Researchers who take a cognitive or educational viewpoint often use the phrase *external representations* [Cox and Brna, 1995], to distinguish them from mental representations [Johnson-Laird, 1983]. Advantages of graphical representations include a direct (or *analogical*) mapping from information structure to representation structure, which supports search and inference processes [Larkin and Simon, 1987, Kulpa, 1994]. Because there is no need to present information sequentially, graphics allow a meaningful use of space [Tversky, 1995, Engelhardt, 2002]. Diagrams can easily be extended to incorporate or derive extra information, e.g., by drawing relationships or adding annotations [Norman, 1993, Suthers, 2003]. Different types of diagrams may put emphasis on different information elements and support different cognitive operations [Cheng et al., 2001, Suthers, 2003]. From empirical studies, it has become clear that reading and creating external representations is a non-trivial skill, which may require learning, just as learning a language [Cox and Brna, 1995]. Some researchers argue for the use of *multiple representations* concurrently, to allow integration of different viewpoints on the same, or related information [van Someren et al., 1998]. However, there are additional costs associated with the use of multiple representations: they all have to be learned, as well as translated into each other to be effective [Ainsworth, 1999]. It is also possible to generate natural language descriptions alongside graphical representations [Mittal et al., 1995]. These may help to clarify how the diagrams should be understood in context, and avoid misinterpretations.

Much of the work in information visualization and diagrammatic reasoning is geared towards numerical information and is highly domain-specific; therefore its applicability is often limited to a particular domain, such as electronics, or software modelling. There is a need for a visual language to illustrate the behaviour of complex systems in conceptual terms, which can be employed across various domains.

In this thesis, qualitative simulations are used as a basis for diagrammatic visualizations of system behaviour. To support the interpretation of the visualizations, and to communicate abstract concepts which are hard to represent graphically, the visualizations in this work are complemented by elementary textual explanatory dialogue, consisting of fact and event descriptions, queries, and tutoring questions.

1.5 Research Goals

Reflecting on the current state of the art, what is still missing are domain-independent learning environments which can take models from different domains as input to generate parsimonious explanations of dynamic phenomena in both textual and graphical form. The overall research goal of this work is to investigate how qualitative simulations can be utilized for this purpose. The GARP framework [Bredeweg, 1992] is chosen here as the basis for this work, because it incorporates a rich ontology, a large set of models in various domains, and a rich set of tools for model-building. The overall research goal can be broken down into a number of research questions:

- how can qualitative simulations be used to support learning about dynamic phenomena?
- how can qualitative models and simulations be visualized in a domain-independent manner?
- how can the complexity of qualitative simulations be reduced to facilitate communication?
- how can textual explanations and questions be generated from qualitative simulations?
- how can graphical and textual means of communication be effectively combined?
- how can different didactic styles be employed in a simulation-based learning environment?

These questions are addressed in this research by a combination of theoretical analysis, system design and empirical evaluation.

1.6 The Structure of this Thesis

This thesis is structured as follows. Chapter 2 describes the design, implementation, and evaluation of a system called VisiGarp, which allows interactive access to visualizations of GARP models and simulations. In chapter 3, techniques are introduced for simplifying qualitative simulations, by aggregating the data into events on different levels of abstraction. Chapter 4 goes into the automatic generation of textual explanations, queries, and questions, as well as didactic strategies which combine these didactic means. This culminates into the design of WiziGarp, which combines the techniques for visualization, aggregation and explanation generation. In chapter 5, the possibilities of WiziGarp are illustrated by two interaction scenarios which exemplify different didactic strategies. Chapter 6 concludes this thesis, with a discussion of the advantages and disadvantages of the chosen approach, as well as directions for future research.

2 Visualization of Qualitative Simulations

"If you think that the Truth can be known from words, if you think that the Sun and Ocean can pass through that tiny opening called the mouth, O someone should start laughing!"

Hafiz (1320-1388), Persian Poet

This chapter deals with the automatic visualization of qualitative models and simulation results. A tool, named VisiGarp, was developed for this purpose. Section 2.1 describes GARP, the underlying qualitative reasoning framework. Sections 2.2 and 2.3 describe the design of VisiGarp, section 2.4 illustrates its possible use, and section 2.5 discusses its evaluation by students and teachers. Section 2.6 compares VisiGarp to related work on visualizing qualitative models. Section 2.7 concludes with a discussion of the advantages and limitations of VisiGarp as well as indications for possible improvements and additions which may contribute to a full-fledged interactive simulation-based learning environment.

2.1 The GARP Framework for Qualitative Reasoning

The GARP framework (Generic Architecture for Reasoning about Physics) combines elements of three approaches to qualitative reasoning: the component centered approach, the process oriented approach, and the contraint centered approach [Bredeweg, 1992]. Therefore, the GARP framework includes a rich ontology, consisting of the following modelling primitives:

- Entities, attributes, and structural relationships. The entities represent the (physical) parts of the system. Each entity is of a certain type. The entity types are organized hierarchically by subtype relationships in the entity is-a hierarchy. Entities can have attributes, which consist of an attribute relation and an attribute value. The structural relationships are defined between entities which are related, and consist of the relationship and two entity references. In most cases, the entities, attributes and structural relationships represent the *physical* structure of the system. In some cases, however, these primitives are also used to represent abstract conceptual ideas, such as assumptions, about the system.

- Quantities with a quantity space, a qualitative value and a derivative. A *quantity* belongs to an entity, and represents a property which can vary dynamically. Each quantity has a domain of possible values associated with it, called its *quantity space*. In a particular situation, or *state*, a quantity has a qualitative *value* and *derivative*. The value is either a value from the associated quantity space, or unknown. The derivative can be either min (decreasing), zero (steady), plus (increasing), or unknown.

- Causal and mathematical dependencies between quantities. Dependencies can be specified between quantities and their values and derivatives, indicating causal or mathematical relationships. Causal dependencies are: positive influence ($I+$), negative influence ($I-$), positive proportionality ($P+$), and negative proportionality ($P-$).[1] An influence is a relationship between a value of one quantity and a derivative of another quantity, *e.g.*, a flow being plus causes some amount of fluid to increase ($I+$). A proportionality is a relationship between two derivatives, *e.g.*, the increase of the amount propagates to an increase in pressure ($P+$). Mathematical dependencies include (in)equalities between quantity values ($<$, $<=$, $=$, $>=$, $>$), (in)equalities between derivatives (*d*$>$, *d*$>=$, *d*$=$, *d*$<=$, *d*$<$), undirected and directed value correspondences (V and $V\hat{}$, respectively), and undirected and directed quantity correspondences (Q and $Q\hat{}$, respectively). A value correspondence specifies that the value of a quantity may be derived from the value of another quantity, (and vice versa, in the case of the undirected version). A quantity correspondence specifies a value correspondence between all values in the quantity spaces of two quantities.

These modelling primitives are used to specify scenarios, model fragments, and termination rules. A scenario is a (partial) description of an initial situation, which functions as input for the simulation. Model fragments are generic pieces of knowledge which apply under certain conditions, and specify processes, actions, situations, assumptions, or combinations of these. A model fragment has conditions and can introduce results. The conditions may include a particular structural constellation, consisting of zero or more entities, attributes and structural relationships, as well as particular quantities, values, derivatives, and dependencies. They may even include other model fragments, to allow hierarchical modelling. The results may introduce new entities, attributes, or structural relationships, new quantities which become important, values or derivatives that become known under these circumstances, or new dependencies which become applicable. The model fragments are organized hierarchically in two ways. The model fragment is-a hierarchy specifies subtype relationships between model fragments. The model fragment applies-to hierarchy specifies which model fragments are conditional for which other ones. A collection of model fragments, scenarios, and transition rules in a particular domain constitutes a domain model library.

A simulation starts with the choice of a scenario: the initial situation description. This is interpreted by the simulation engine (or simulator) by finding applicable model fragments, which result in the begin state: the qualitative state that the system is in, initially. Model fragments are applied to a particular state when their conditions are true (or can be assumed) and their results do not lead to contradictions with information already known to hold in that state. If the scenario is underspecified, there may be multiple interpretations of it, resulting in multiple begin states. Then, the simulator tries to find termination rules. Most of the termination rules are generic, and specify what may happen to quantities given their current value and derivative. Domain-specific termination rules are sometimes added to specify possible transitions which include other modelling primitives as well, *e.g.*, structural entities or relationships. There may be multiple termination rules applicable to a certain state, *e.g.*, when multiple quantities are changing at the same time. Because some changes occur before others, the applicable termination rules must be ordered to find out which changes will occur first, and which rules can be combined together. Finally, the consequences of the termination rules must be determined and checked for consistency to see which of the rules result in valid transitions to another (new, or already existing) state. Depending on how far the simulation has progressed, the status of a state is thus new, terminated, ordered, or closed.

[1]There are many views on causality in the literature, not all of which agree with the characterization of influences and proportionalities as *causal* dependencies [Lehmann, 2003]. In GARP, the cause should be seen as a sufficient condition for the effect, *ceteris paribus*. It is not a necessary condition, because there may be multiple influences or proportionalities operating on a particular quantity.

The processes of finding applicable model fragments to build states from partial situation descriptions and finding termination rules to generate state transitions are repeated, until no more transitions are possible. The resulting collection of states and transitions can be regarded as a graph, the *state-transition graph*. This is sometimes called behaviour graph, or (attainable) envisionment, as each path in the graph represents a possible behaviour that the system may exhibit in the simulation, according to the model that is used [de Kleer and Brown, 1984, Kuipers, 1986, Forbus, 1990].

2.2 The Design of VisiGarp

GARP processes its input (a scenario, model fragments, and transition rules), and generates its output (the possible states of behaviour) in a predicate logic (Prolog) format. Although concise, this format is not easily read (especially by novices), and it is hard to search for particular kinds of information (even for qualitative reasoning experts) as soon as a model becomes moderately large. Furthermore, the mainly text-based GARP interface does not exploit the potential of explicitly visualizing the different ontological categories of information captured in a simulation. The assumption is that appropriate visualizations can make the model and simulation results more insightful for students.

This has motivated the design of VisiGarp, a tool that provides users (*i.e.*, students without expertise in qualitative reasoning) with interactive access to automatically generated graphical representations of qualitative models and simulation results. VisiGarp may serve as the basis for an interactive simulation-based learning environment which helps students to understand how a system in the world behaves by interacting with simulations based on qualitative models of that system.

VisiGarp is based on the GARP framework and is implemented in SWI-Prolog/XPCE.[2] The relationship between GARP and VisiGarp is depicted schematically in figure 2.1. The figure shows a user interacting with VisiGarp, which receives its input from the domain library and a simulation. The user interacts with VisiGarp by means of navigation options and simulation controls. With the navigation options, the user can select and manipulate VisiGarp's *library views* and *simulation views*. The library views contain generic information from the domain library, including the scenarios, model fragments, and rules for the transitions. The simulation views contain information for a specific simulation, which consists of states (including instantiated model fragments) and the transitions between them. With the simulation controls, the GARP simulation engine can be controlled in various ways. The user can select a scenario to start a new simulation, open a saved simulation, run a simulation completely, or constrain the simulation process in the direction of particular states.

The design of VisiGarp is based on the assumption that appropriate visualizations make the model and simulation results more insightful for students. The research questions involved in the design process are the following:

- How to generate insightful visualizations of qualitative models and simulations?
- How to organize the interface to allow easy navigation between the different visualizations?

In order to answer these questions, visual primitives and visualization principles are presented as the basis for the visualizations in VisiGarp.

[2]Information about SWI-Prolog, and the graphical user interface toolkit XPCE is available from www.swi-prolog.org.

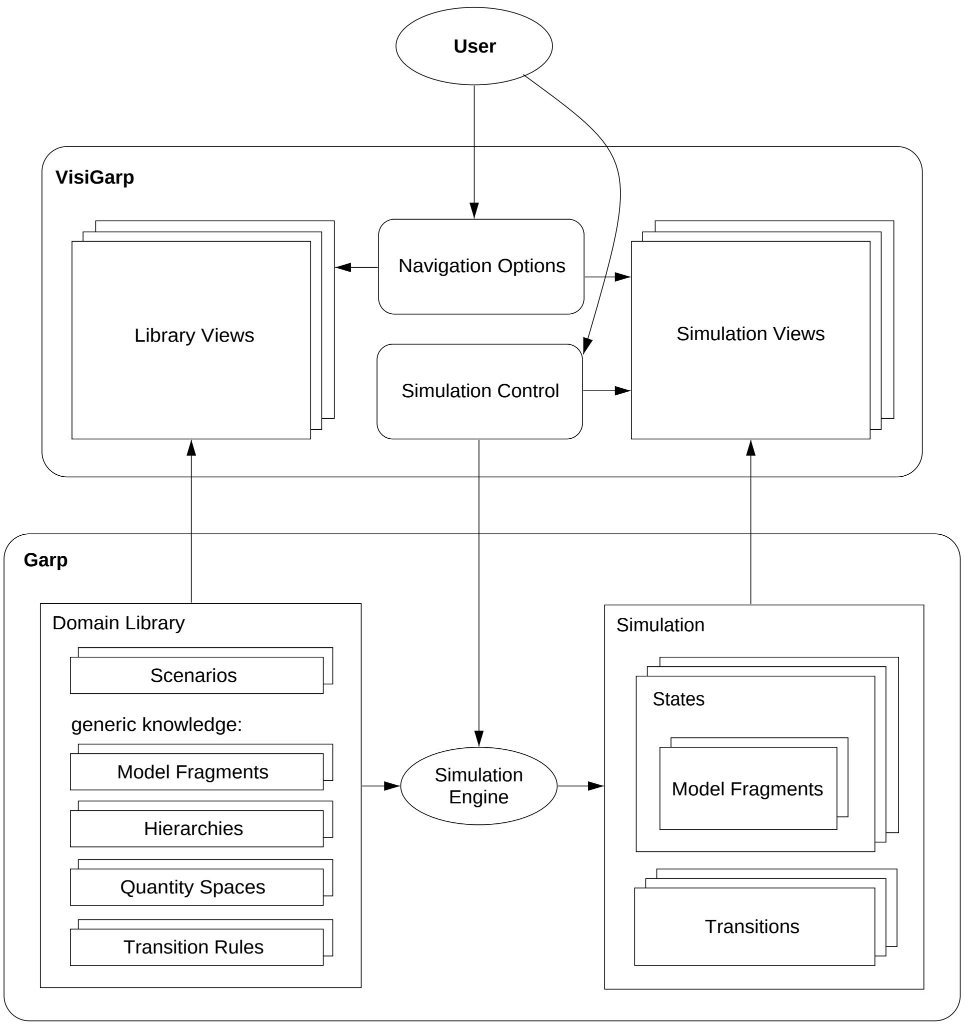


Figure 2.1
The relationship between GARP and VisiGarp.

2.2.1 Visual Primitives

Visual primitives, or ’elementary graphical objects’ [Engelhardt, 2002, p. 23], are ’the smallest meaningful components’ [Richards, 2002, p. 93] in a visualization. These visual primitives can have characteristic visual attributes to distinguish them: they can differ in shape, size, colour, position, and direction. The visualisation ontology used in this thesis consists of circular, rectangular and oval shapes of variable sizes, as well as text labels for different kinds of concepts in the model, and lines, arrows, inclusion, ordering, and indentation for different kinds of relations between concepts in the model. Figure 2.2 presents the mapping from the main GARP modelling primitives to graphical elements in the visualization. Where necessary, the visual primitives are shown within their visual context. The principles governing how this is done are described in section 2.2.2. A state is displayed either as a coloured circle labeled with an identification number, or the number only. A termination is shown as a small coloured circle connected to the originating state by a small line, or as the identification label, followed by the name of the transition rule that the termination contains. A transition

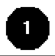
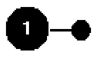
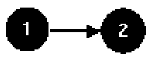
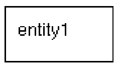
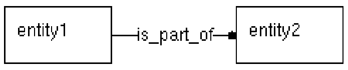
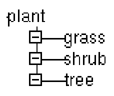
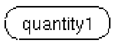
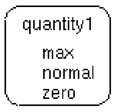
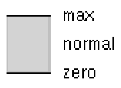
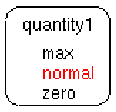
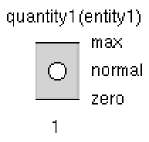
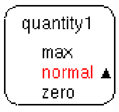
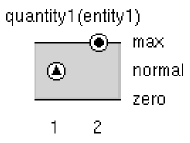
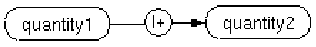
Modelling primitive	**Visual primitive(s)**
State	 1
Termination	 1a: to_point_above(quantity1)
Transition	 1 -> 2: to_point_above(quantity1)
Entity	 entity1
Relation	
Attribute	material: kryptonite
Is-a relation	
Quantity	 quantity1 (entity1) entity1: quantity1
Quantity space	 
Quantity value	 
Quantity derivative	 
Dependency	

Figure 2.2
The mapping from GARP modelling primitives to their corresponding visual primitives.

is displayed as an arrow from the originating state to the resulting state, in graphics, or text format. In the latter case, it is followed by the name of the transition rule that resulted in the transition. An entity is displayed as a labeled box, or just by the label itself. A relation is displayed as a labeled line between two entities, with a dot indicating the reading direction. In textual format, the line and the dot are left out. An attribute with its attribute value is displayed as ’Attribute: Value’. An is-a relation between entity types (*e.g.*, grass, shrub, tree) and their supertype (*e.g.*, plant) is displayed graphically by indentation, with small lines and boxes to connect the types. A quantity is displayed graphically as a labeled box with round edges. In text format, a quantity is denoted by its name, which may be followed or preceded by the name of the entity it belongs to. A quantity space is shown by vertically listing the values it contains, ordered from low values at the bottom to high values at the top. A quantity space is shown either within the quantity it belongs to, or separately, with the value labels next to a graphical space. In the latter case, the difference between points and intervals is shown as lines and grey areas (*i.e.,* zero and max are points, and normal is an interval value). A quantity value is displayed graphically as a highlighted value label in the context of a quantity space, or as a small white circle on a line or between lines within a quantity space. A quantity derivative is displayed graphically as a small black triangle pointing upwards (plus) or downwards (min), or a small black circle (zero). This is shown besides the value label, or within the value circle. A dependency is shown graphically where possible as a labeled arrow (or line, in the case of an undirected dependency) between two quantities or values.

As noted above, for many modelling primitives (state, termination, transition, entity, quantity, quantity space, value and derivative), multiple visualizations are used. These visualizations differ in the use of visual attributes and in interface behaviour (*e.g.,* whether they can be selected, opened, moved, etc.). Which mapping is used in a particular view depends on the context (*e.g.,* is it a state-based, path-based, or generic view?), and the role of that particular graphical element in the visualization as a whole (*e.g.,* does it show detailed information, does it refer to, or provide interactive access to information in another view?). These view-specific considerations are discussed in section 2.3, which describes the various views of VisiGarp.

Generic and abstract vs. domain-specific and pictorial. In related work on simulation-based learning environments (e.g., Hollan et al. [1987], Forbus et al. [1999]), the interface is often centered around a pictorial, or schematic representation of the system that is modelled. In contrast to such an approach, the visualization approach taken in this thesis is generic and does not use domain-specific pictures or symbols. Instead, only non-pictorial elements (abstract shapes, words and numbers) are used to denote specific kinds of model primitives. The motivation for this is threefold. First, the abstract kind of representations used in VisiGarp are useful for a student to learn, as they make explicit important modelling concepts which can be used across various domains (e.g., see Gentner [1983], Falkenhainer et al. [1989]). Second, to avoid becoming dependent on the software, it would be good if students were able to draw similar diagrams themselves. Using only geometric shapes, words and numbers (and avoiding too much use of visual attributes such as brightness and texture) ensures that the visualizations remain simple enough for students to draw by hand. Third, pictorial graphics would require domain-specific knowledge to be added to the system, which would make it hard to reuse the system for different domains.

In some domains (*e.g.,* electronics) standardized visual languages exist which include domain-specific symbols; the generic diagrams presented here are not intended to replace these specialized visualizations. The usefulness of (semi-)realistic figures in educational settings should not be ignored, as they can help in drawing attention, increasing motivation, or creating context. Nevertheless, this study investigates what can be achieved using only non-pictorial, domain-independent graphics. The popularity of generic visual formalisms such as E-R diagrams and state-charts in various domains suggests that such an approach has potential [Harel, 1987, 1995].

2.2.2 Visualization Principles

Visualization principles specify how visual primitives are combined to form composite graphical objects, representing larger units of information. In VisiGarp, the largest units are views: diagrams which provide an overview of multiple elements, or a detailed presentation of a particular model element. Since the principles for visualization design from the existing literature, as mentioned in section 1.4, do not prescribe how to generate diagrams for qualitative simulations, a more extensive set of principles is introduced here for this purpose.

- Information related to a specific model concept is displayed, if possible, within the visual element representing that concept (*i.e.*, using a visual *container metaphor* [Lakoff and Johnson, 1980]. This eliminates the need for repetitious labeling or linking, supports hierarchical search strategies, and constrains the layout of the display in a systematic way.

- Information which relates multiple model concepts is displayed as a separate visual element, connecting the graphical objects which represent the model concepts involved, as much as possible. Encapsulating the graphical objects is impractical in this case, since they might be large, or far from each other, thereby creating large and awkward container shapes. Another alternative is labeling all graphical objects involved with information about the relationship, but this would generate needless repetition.

- If an information element has more detailed information associated with it, it should be displayed as a separate visual element, large enough to be selected and to incorporate connections. If an information element has no further information associated with it, it may be displayed smaller, and/or in the context of another visual element, in the form of a textual label, or a visual attribute. This ensures that rich information elements (*i.e.*, with lots of related information) are assigned more space on the screen than more basic information elements, thereby increasing the data-ink ratio [Tufte, 1983] and minimizing the amount of space necessary. Since it is easier to recognize and select large graphical objects, this also supports navigation for information in a natural way.

- Information which is only meaningful in a context should be displayed within that context as much as possible. This should reduce the chance of misconceptions. Highlighting can be used to make the information stand out from its context, if necessary.

- When multiple types of relationships are combined in one view, labeled links can be used to distinguish them; if all relationships to be displayed in a view are of the same type, unlabeled links, or indentation (for relationships between textual elements) may be sufficient.

- Textual labels are shown horizontally only (not rotated), and are placed in the middle or top-left position of the objects they identify, to allow easy reading.

- When labels (potentially) have to be represented many times on the same display, abbreviations or icons should be used to save space. In other cases, the use of abbreviations and icons is kept to a minimum as it may not be worth the user's effort to learn their meaning.

- Information which indicates that something is unknown is only shown when leaving it out might cause confusion.

- Temporally ordered information is displayed with time progressing along the x-axis as much as possible.[3]

[3] Because qualitative reasoning often results in multiple possible behaviours, which may also be cyclic, this is not a strict requirement - states can occur multiple times within a single behaviour, making a complete linear ordering impossible.

Composition and meaningful space. To show how certain model concepts structurally fit together, visual primitives are combined graphically by positioning them besides, or inside each other (see figure 2.3). For example, all attributes and quantities belonging to the same entity are grouped together within the block representing that entity. Furthermore, within a quantity node, the quantity space can be shown as a vertical list of all possible values. The current value in the selected state is highlighted in red, and flanked by a derivative symbol indicating increase, steadiness, or decrease.

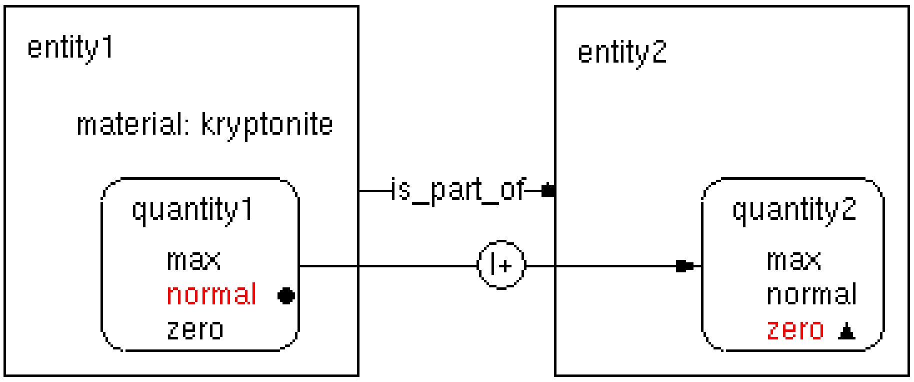


Figure 2.3
Entities, attributes, quantities, quantity spaces, values, derivatives, and dependencies combined.

When also the relationships between entities and the dependencies between quantities and between quantity values are shown, the collection of nodes, subnodes and the various kinds of links forms a kind of hierarchical graph [Harel, 1995]. This facilitates recognition of dependencies within subsystems, and dependencies crossing subsystem borders. In VisiGarp, links occur only between nodes of the same type, *i.e.*, from entity to entity, from quantity to quantity, or from value to value (not shown in the figure), however - this is not always the case in hierarchical graphs.

Besides the shape of a visual element, the position of a visual element with respect to its graphical context may also be used to communicate information–this is known as the use of meaningful space [Engelhardt, 2002, p. 54]. The positioning of the highlighted value and derivative symbol on the vertical dimension along which the quantity space values are ordered is an example of the use of meaningful space. A slightly different use of meaningful space is necessary when information from

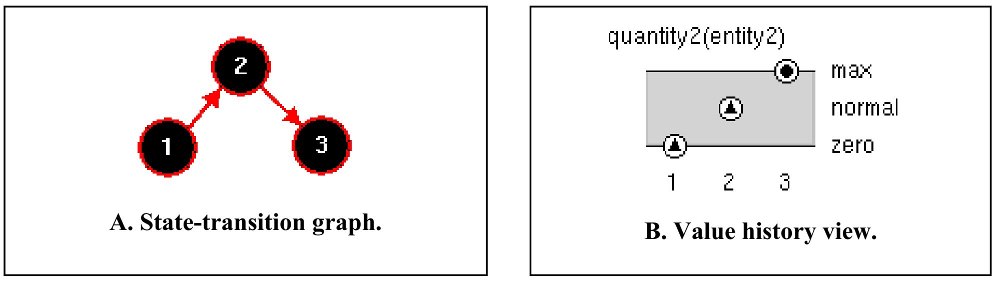


Figure 2.4
Two different visualizations of paths.

multiple states is shown, *e.g.*, the values of a particular quantity along a path of successive states. In this case, the context is not specified by the structure of the system, as before, but rather by the path of states. This path often has an angular shape in the *State-transition graph* (see figure 2.4A, on the left-hand side), but in the *Quantity value history view*, the path is straightened out and simply represented as a horizontal linear sequence of state numbers (see figure 2.4B, on the right-hand side). The value of a quantity in a certain state is not represented by a highlighted text label, but by a small circle. This circle is horizontally aligned to the corresponding value label. The derivative symbol (if the derivative is known) is displayed inside the value circle. Since this view most explicitly focuses on quantity values, all available detail is shown in the representation of the quantity space. Point values are indicated by lines in the quantity space. Interval values do not have a line, to make clear that the value can be anywhere above/below/in between the point values around it.

2.3 The Different Views in VisiGarp

VisiGarp offers a number of views, each of which contains a particular combination of information elements from the simulation. An overview of all the views available in VisiGarp (version 1.0) is shown in table 2.1. Each view focuses on certain kinds of information, shows context information where needed, and provides links to more detailed information if necessary.

The dependencies view, entities and relations view, and the model fragments views are *state-based* visualizations, while the value history and transition history view are *path-based* visualizations. Most of the views are directly accessible by selecting a particular (set or path of) state(s) in the main window and clicking on the appropriate view-button, or option in the view-menu. The views marked with a * display generic knowledge in the domain model library - they do not contain information specific for a simulation. As such, they are only accessible from the view menu. The transition details view, and the model fragment graphics and text views are only accessible through other views (the transition history and model fragments view, respectively), to structure the navigation process from overview to more detailed information. The set of views covers all of the model primitives present in the GARP framework. Multiple views can be opened simultaneously, allowing users to compare different views of the same situation (or different situations), or to combine global overviews with more detailed descriptions.

For each view, a choice has been made (from the options shown in figure 2.2) for the mapping from model primitives to visualization primitives, to facilitate specific reasoning tasks (identifying system structure, controlling the simulation, characterizing system dynamics, navigating between and within windows, following value changes, tracing back causal chains, etc.) and interaction types (reading, selecting, dragging, clicking, resizing, etc). In the next subsection, these choices are discussed for each of the views, as well as possible alternatives which have been considered.

View	**Information, visualization and interface characteristics**
Main Window	Includes the file, view, help and display menus, as well as the buttons for opening other views. Once a simulation is opened, the main window displays the *state-transition diagram*. States, transitions, and state-transition paths can be selected and investigated in detail using one of the views below.
Transition history	Shows a high-level tabular overview of all transitions in a selected path, with access to further details of individual transitions.
Transition details	Shows the name of the termination rule, the conditions leading to it, its results, and the status of the termination (open, terminated, ordered, or closed).
Entities and relations	Block diagram showing the elementary structure of the system in terms of entities, attributes and relations.
Entity is-a hierarchy*	Subtype-relationships between different kinds of entities, indicated by horizontal indentation to allow textual labels to be read accurately, using collapsible nodes to allow various levels of detail.
Dependencies	Shows all quantities, grouped within their entities, and the causal and mathematical relationships between different quantities, displayed as a higraph.
Quantity values	Shows quantity values and their derivatives for a specific state in tabular format.
Quantity value history	Displays a graph for the values of a selected quantity changing over time during a selected path in the state-transition graph.
Model fragments	A list of all model fragments which are applicable in a selected state, with access to further details in either text or graphics format.
Model fragment: text	Structured text display of the contents of a particular model fragment, with conditions and results separated vertically, and indentation for the different categories of knowledge types.
Model fragment: graphics	Graphical display based on the dependencies view, with the contents of the particular model fragment highlighted in colour (red for conditions, blue for results) and the rest in light-grey to show the context.
Model fragment is-a hierarchy*	Subtype-relationships between different model fragments (processes, agents, situations, and combinations), indicated by horizontal indentation to allow textual labels to be read accurately, using collapsible nodes to allow various levels of detail.
Model fragment applies-to hierarchy*	Similar to the is-a hierarchy, this view shows which model fragments are conditional for which other model fragments.
Scenarios*	Shows the contents of all input models (scenarios) available in the domain model library, displayed in the textual GARP format. This view was added for completeness; a graphical visualization of this information was not deemed necessary.
Transition rules*	Shows the details of all transition rules in the domain model library in the textual GARP format. This view was added for completeness; graphical visualization was not considered necessary.

Table 2.1
Description of all views available in VisiGarp 1.0. Library views are marked by a *.

2.3.1 The main window: opening a simulation

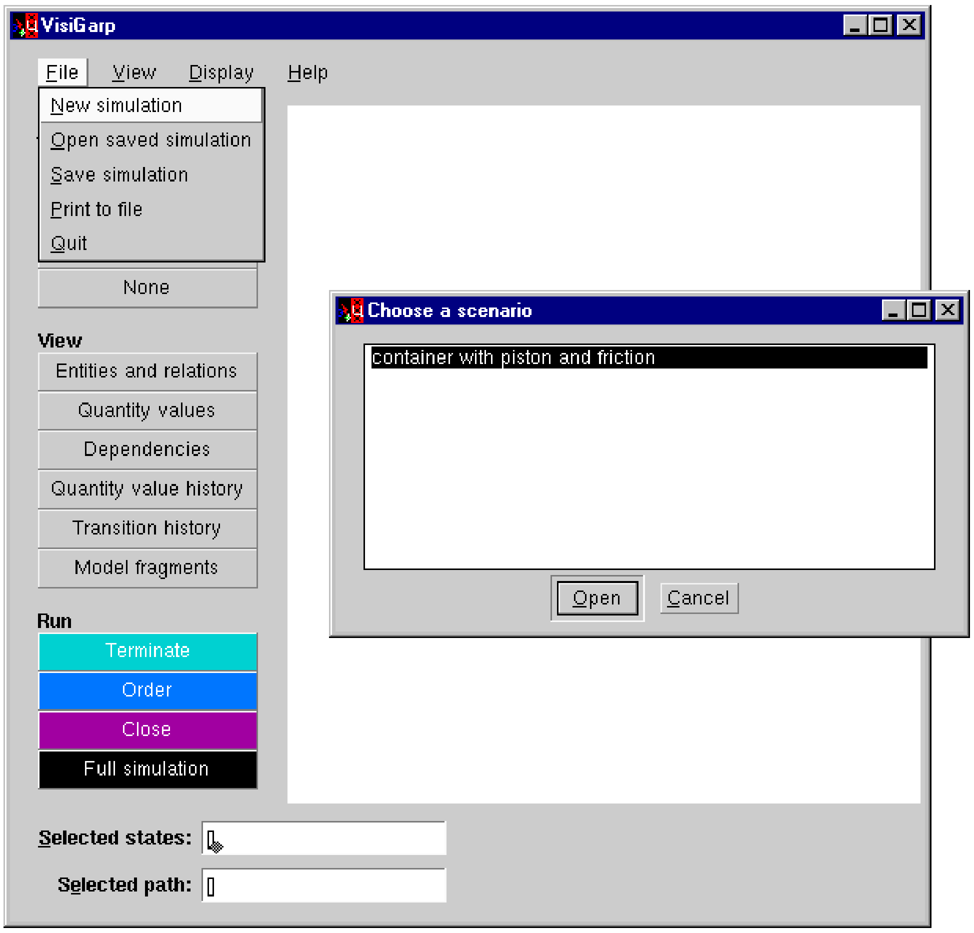


Figure 2.5
Opening a new simulation in VisiGarp.

The main window, shown in figure 2.5, serves to control the simulation and gives access to the various views on the simulation and model library.

Choosing a simulation scenario

In VisiGarp, the user can either start a new simulation, or open a saved simulation by choosing the desired option from the file menu in the main window. When the user decides to start a new simulation, a list of available scenarios is displayed, from which the user can select one to open. As displayed in figure 2.5, a scenario is selected called 'container with piston and friction'. This simulation scenario, from the domain of physics, is used as a running example in the remainder of this section. The simulation features a container and a piston, which moves outwards when the gas in the container is heated. A schematic picture (created by hand) of the most important elements in the simulation is shown in figure 2.6.

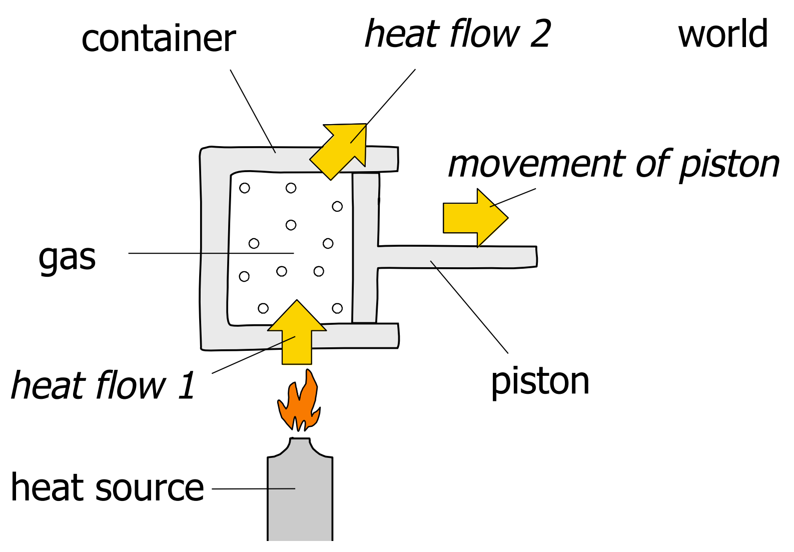


Figure 2.6
A schematic overview of the most important entities and processes in the container-piston simulation.

Menus and Buttons

Besides opening a new or saved simulation, the file menu also offers options to save a simulation, to print the state-transition graph to a postscript file, or to quit the program.

The view menu offers the choice between all the views in VisiGarp, except the Transition details, and Model fragment details views, which can be opened only from views showing transitions and model fragments, respectively (see figure 2.7). Above the line are all *simulation views*, which are specific for a state or path in a simulation, below the line are all *library views*, which show generic information from the domain model library. The simulation views (which are used most often) are also available by pressing one of the buttons on the left side of the main window. The display menu allows actions to modify the layout and some visualization preferences. The help menu currently does not do anything, but should present a legend to the currently active view, as well as an index to help information about other views.

2.3.2 The main window: the state-transition diagram

When a simulation is opened (new or saved), the main window contains the state-transition graph. This shows an overview of the progress of the simulation in terms of states and state transitions (see figure 2.8). This can be compared to the kinds of diagrams generated by the QSIM software [Kuipers, 1994]. An interesting difference is that in their approach it is not the individual states which are numbered, but the possible paths of states (called *behavior*, in the QSIM framework). This can be useful, when the simulation consists of several distinct sequences of states, but not when there are many branches in the simulation, because that makes the number of paths grow exponentially. Furthermore, for the target users of VisiGarp (students), it is important to be able to refer to specific states, because a lot of conceptual information is associated with each state. Another advantage of the VisiGarp

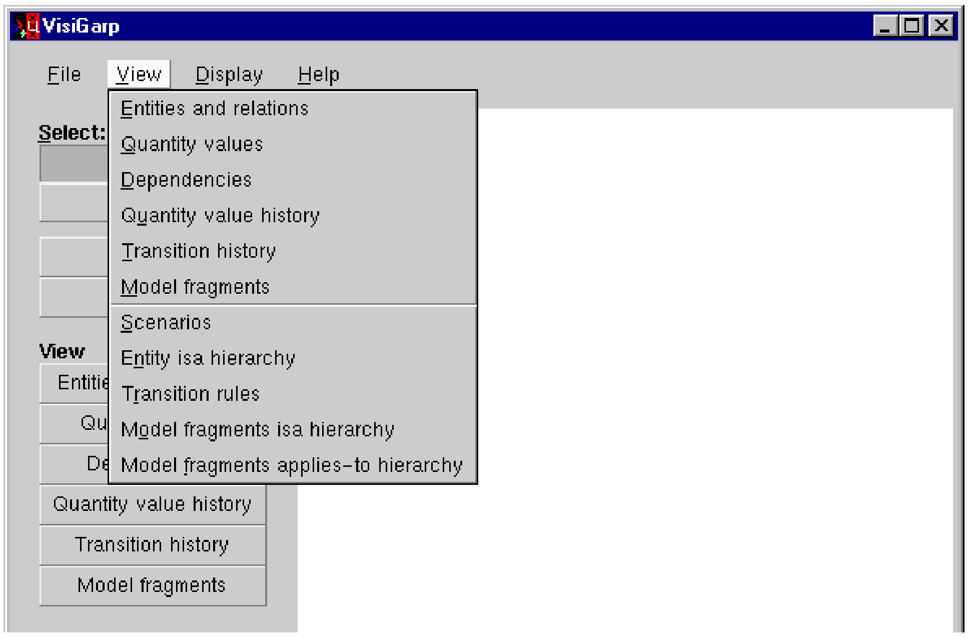


Figure 2.7
The View menu, showing the state-specific views above, and the generic library views below the lineOpening a new simulation in VisiGarp.

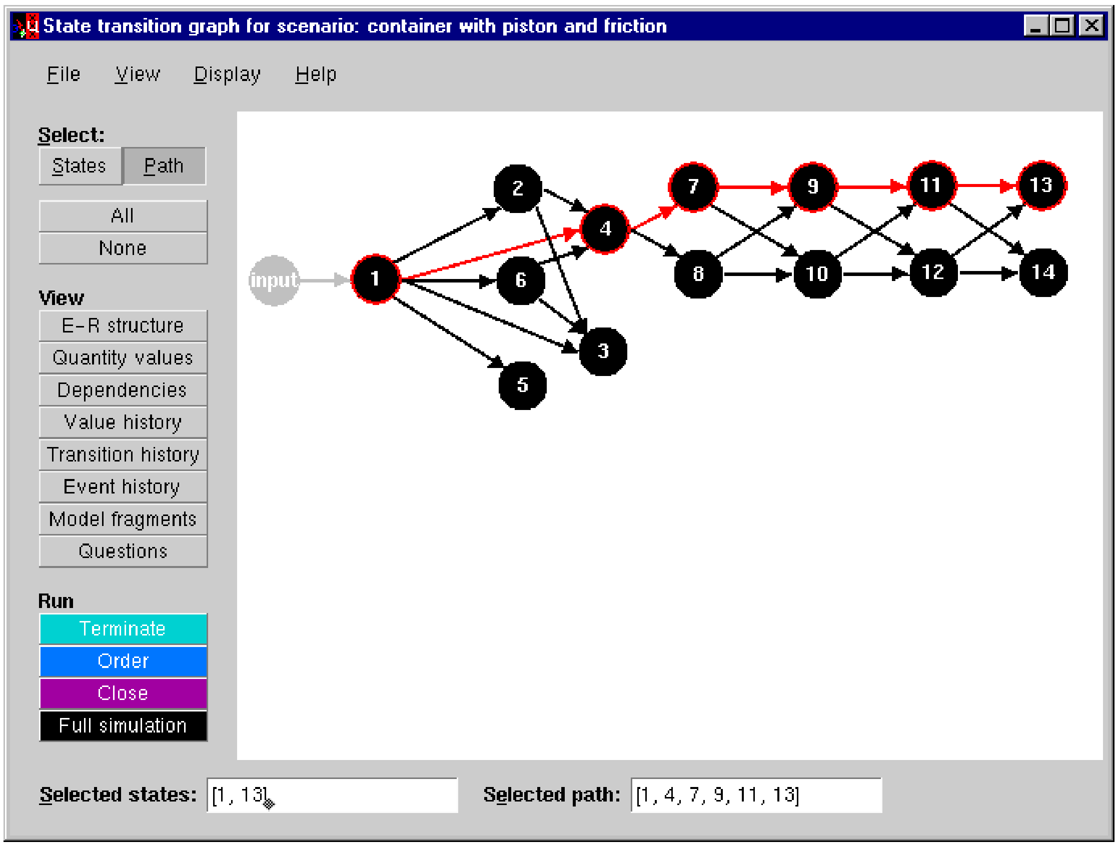
representation is that certain characteristics of states (e.g., end states) and paths (e.g., cycles) in the simulation are easy to recognize.

State: Black or coloured circle with white textual label, centered in the middle.

To represent a state, a circle was chosen because it takes up the least amount of space. The grey circle represents the scenario, or input model, which is not a real state, since it contains only a partial situation description. Nevertheless, it is displayed in this view because it shows which states are possible interpretations of the input model (the ones connected to the input model via a grey arrow - only state 1 in the example). More information about the input model can be viewed by selecting the input model 'state' and opening the Entities and relations, Quantity values, or Dependencies view as if it were a normal state. Because the input model contains uninstantiated variables, the name labels for these may differ from the instantiated ones in views for normal states (uninstantiated variable names start with a capital).

In figure 2.8, a full simulation is shown, in which all states are closed, and all unsuccessful terminations are hidden from view. Figure 2.9 shows the same simulation in progress, with terminations, and states of different status.

Because the number of states can generally grow to more than 100, the size is set such that numbers up to 999 still fit the circle. When states are selected by a mouse-click, a shift-mouse-click (to select multiple states) or by entering their numbers in the *selected states* entry field at the bottom of the main window, a red selection boundary is added around them. The fill colour of a state denotes its status, which is related to the terminations originating in that state:



Selected states: [1, 13]   Selected path: [1, 4, 7, 9, 11, 13]

Figure 2.8
State-transition graph for a simulated scenario involving a container with a piston and friction. A path has been selected between states 1 and 13, consisting of states 1, 4, 7, 9, 11 and 13.

- Aqua: new state, yet to be terminated
- Blue: terminated, yet to be ordered
- Purple: ordered, yet to be closed
- Black: closed

The logical sequence in status values, from new to closed, is mapped visually to a sequence of colours, from light aqua, via blue and purple, to black. Other colours, as well as different shades of grey were found to be less clearly distinct, or to lead to confusion with the red selection boundary. States of different status could also have been visualized using slightly different shapes, or an extra label, but the basic shape of a circle is preferred because it best allows arrows to be attached.

Termination: Small black circle.

Terminations are transition rules which have fired, but have not (yet) resulted in a state transition. In GARP, terminations are labeled (albeit randomly), and they can be ordered and closed individually. In practice, it is usually convenient to treat all terminations of a state the same way. This implies that there is no real need to distinguish terminations, until they lead to a state-transition. For this reason,

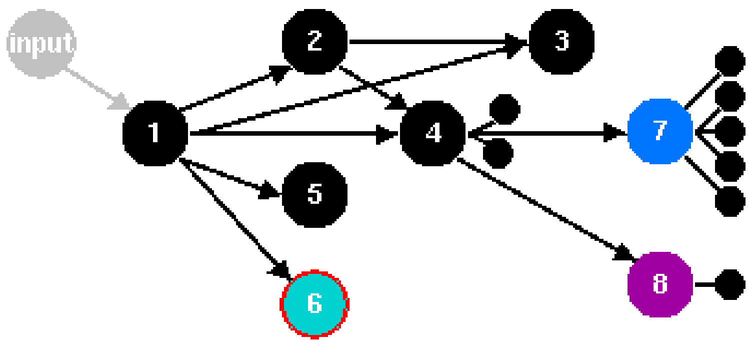


Figure 2.9
State-transition graph for a simulation in progress, showing states with different status, and terminations.

VisiGarp does not show terminations by default. They can be shown, however, as small black circles connected to their originating state (see figure 2.9), by choosing the Display-menu setting 'Show Termination Nodes'.

When a state with terminations is closed, any of the terminations may result in a transition to another state. When this happens, the corresponding termination node is removed and replaced by an arrow to the new state. Termination nodes which remain after closing a state represent termination rules which did not lead to a transition to a new state.

Transition: an unlabeled arrow.

A transition indicates an ordering between two states. A transition is visualized as an arrow between the two states involved. In GARP, transitions are labeled with a randomly generated label, but since different transitions in principle lead to different states, it is considered superfluous to display such a label in the visualization.[4]

Simulation control

At the bottom left alongside the diagram in figure 2.8, under the 'Run' heading, four coloured buttons are provided to control the simulation. The simulation can be run step-by-step, using the top three buttons: Terminate (aqua) checks whether there are termination rules applicable to the selected state(s); Order (blue) combines termination rules if possible, and rules out termination rules which are preceded by other ones; Close (purple) processes the results of the remaining termination rules, and generates new states for succesful transitions. The colour of these buttons corresponds to the colour of the states, indicating their status. This way, users can see which control action should be chosen to proceed to the next termination status. Instead of proceeding step by step, the user can also click on the button Full Simulation (black), to run the complete simulation at once; all branches will then be pursued until no further progress is possible.

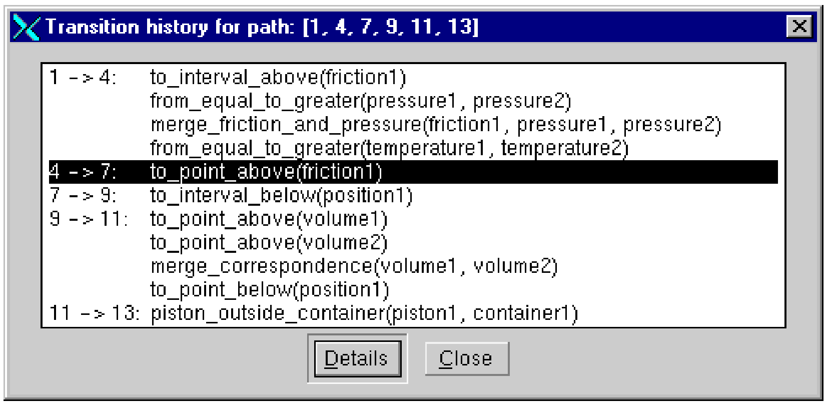
States, terminations and transitions can be investigated in more detail by selecting them and clicking one of the other view buttons. If one is interested in a particular behaviour of the system represented by a path of states, the Select-mode can be set to Path instead of States (the default) using

[4]In fact, it is possible to include multiple alternative transition rules to create the same successor state (which would require the need for a label), but this is considered bad modelling practice.

the Select choice buttons on the top left of the main window. When two or more states are selected in Path-mode, the shortest path through these states (if one exists) is automatically selected. This is especially useful for the transition history and the quantity value history views, since they show behavioural aspects of multiple transitions/states in a single screen.

2.3.3 The transition history view

The transition history view (see figure 2.10) shows a short textual description of all transitions for a selected path, or all terminations/transitions for selected states. This gives a brief overview of all transition rules which triggered a state transition. Note that the selected path is visualized in two ways, which are both different from the state-transition graph. On the left, the transitions are listed below each other for easy search for both from- and to-states. Each transition is visualized as an arrow between two state numbers, and the transition rules are indented to show which transition they belong to. In the bar on the top of the window, the whole path is just presented horizontally as a list of states, as a brief identifier to distinguish the path from other ones if multiple transition history views are opened for different paths at the same time. A more detailed description of individual terminations is available by selecting one and clicking on the Details button.


Transition history for path: [1, 4, 7, 9, 11, 13]

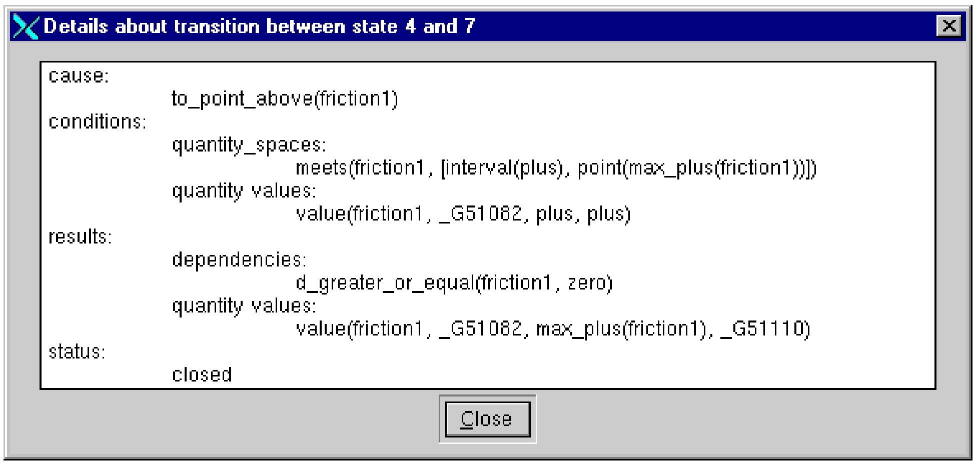
```
1 -> 4:    to_interval_above(friction1)
           from_equal_to_greater(pressure1, pressure2)
           merge_friction_and_pressure(friction1, pressure1, pressure2)
           from_equal_to_greater(temperature1, temperature2)
4 -> 7:    to_point_above(friction1)
7 -> 9:    to_interval_below(position1)
9 -> 11:   to_point_above(volume1)
           to_point_above(volume2)
           merge_correspondence(volume1, volume2)
           to_point_below(position1)
11 -> 13:  piston_outside_container(piston1, container1)
```

Figure 2.10
The transition history for the path between state 1 and 13.

2.3.4 The transition details view

The transition details view, shown in figure 2.11, presents the details of a transition in a textual format closely matching that of the original GARP representation: a rule consisting of a Cause (the trigger for the transition), Conditions (necessary for the rule to fire), Results (the consequences), and a Status (open, terminated, ordered or closed, depending on how far the simulation has progressed). The conditions and results can contain statements about the entity-relationship structure, quantity spaces, values, derivatives, dependencies, and (only for the conditions) model fragments.

In the figure, the termination rule 'to_point_above' applies to the quantity *friction1* (denoting the friction between the piston and the container) in state 4 because this quantity has the value and derivative plus (is increasing), and the quantity space of friction1 contains a higher point value, max-plus. This has resulted in a transition to state 7, in which friction1 has reached this new value, while the derivative is constrained to be greater or equal to zero (i.e., it can keep increasing or become steady, not suddenly decrease).



```
cause:
            to_point_above(friction1)
conditions:
            quantity_spaces:
                        meets(friction1, [interval(plus), point(max_plus(friction1))])
            quantity values:
                        value(friction1, _G51082, plus, plus)
results:
            dependencies:
                        d_greater_or_equal(friction1, zero)
            quantity values:
                        value(friction1, _G51082, max_plus(friction1), _G51110)
status:
            closed
```

Figure 2.11
Transition details for the transition from state 4 to state 7, showing fragments of the GARP code.

The current visualization, which still includes fragments of the propositional GARP code, with many brackets and uninstantiated variables, is not ideal. It is readable for GARP model builders, but for the target user group of students, a more graphical format such as that used in the dependencies view (described in section 2.3.7), or a description in natural language would probably be better.

2.3.5 The entities and relations view

The entities and relations view, shown in figure 2.12, combines entities, attributes and relationships, which describe the parts of the system modelled, as well as their structural characteristics in a particular state. The behavioural properties (i.e., quantities, values, etc.) are deliberately left out of this view, to focus on the structure rather than the behaviour.

Entity: Rectangular box with text label positioned in the top left corner.

An entity is a part of the system under investigation, either a physical part, or a conceptual entity. An entity is displayed as a box with a label displaying its name in the top-left corner, a natural place to start reading. A possible alternative position would be in the middle of the box, but this would be inconvient for reasons discussed in section 2.3.7.

Attribute: Text label positioned inside the entity involved.

An entity may have certain characteristics (i.e., structural property, operation mode, colour) which do not change, or change only in a discrete way. For modelling these characteristics, attribute-value pairs can be used, where the value specification is just a textual label (which has no quantity space attached, as quantities do). Attribute specifications are displayed as text labels inside the entity they apply to, indented just below the name of the entity, as shown in figure 2.12 for the material (*steel*) of *container1*.

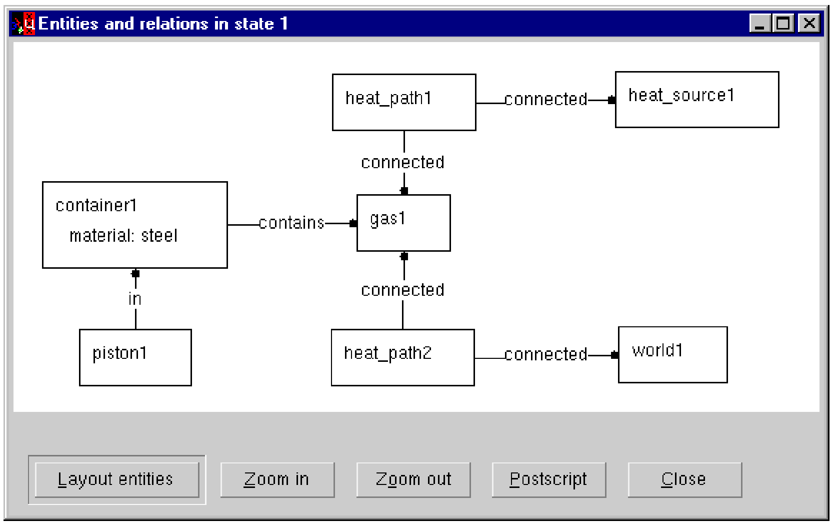


Figure 2.12
Entities and relations.

In GARP, attribute specifications are represented in the same way as relationships between entities, but in VisiGarp they are visualized differently to clarify that attributes belong to a certain entity, while relationships hold between two entities.

Relationship: Labeled line with a dot on one end.

The model builder is free to create relationships between two entities and define labels for them. Therefore, the exact meaning of a relationship is not known to the system - it has to be inferred by the user from the label. From experience with model builders in several domains, most relationships specify structural relations between entities. Because the order of entities often matters, a relationship between two entities is considered a directed relationship in VisiGarp, although the GARP representation does not specify this explicitly. This directedness is not indicated by an arrow because that has been reserved for indicating time sequence and causality; instead, the arrowhead at the end of the line has been replaced by a round dot. The text label is added on top of the middle of the line.

An alternative way to visualize a relationship, based on the containment-principle, would be to visualize it as a separate node, large enough to contain the relationship name label inside it. However, this label can be rather long, since it is created by the model builder (unlike the dependencies, which come in predefined categories only). Therefore, a container shape around it would be of unknown width, and could become rather large. It was felt that introducing a third type of node (besides entity and quantity nodes) to display relationships as separate nodes could be confusing, would take more space, and would draw more attention than necessary.

The layout and zoom options and mechanisms for this view are similar to those for the dependencies view, which are discussed in section 2.3.7.

2.3.6 The entity is-a hierarchy view

The entity is-a hierarchy is a view on generic knowledge in the domain model library and is only accessible from the view menu. It shows the subtype-relationships which exist between different kinds of entities (see figure 2.13).

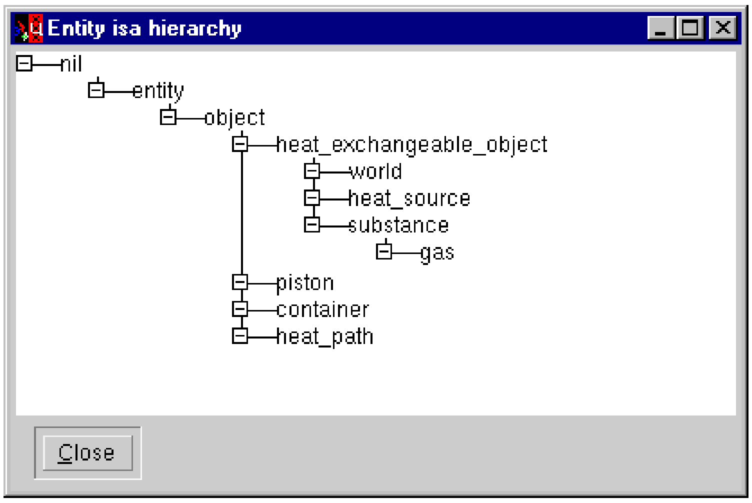


Figure 2.13
Entity is-a hierarchy.

The entity types are visualized as textual labels, indented and connected by lines where an is-a relation applies, using the XPCE indented list format. Is-a hierarchies, or *taxonomies*, are often displayed in a tree format, *i.e.*, vertically growing from the top category into various subtype-branches to the bottom leaves, or horizontally, growing from left to right. The indented list format is preferred above such trees, however, because trees would require more space vertically or horizontally to accomodate the same number of labels on the screen. Furthermore, the indented list format in XPCE has the feature that parts of the hierarchy can be hidden, thereby allowing exploration of large hierarchies on different levels of detail. This is done by clicking on the minus sign, which causes the part below it to collapse; in this case, the minus sign changes into a plus.

2.3.7 The dependencies view

The dependencies view shows all quantities and the dependencies between them, which allows investigation of the causes and effects of changes to quantities in the simulation. It incorporates the entities (and optionally, also the relations) from the Entity-relations view to relate the quantities and dependencies to the system structure, and also to enforce some organization in the layout. The layout can also be adjusted by hand, by dragging entities or quantities. When more than a handful of quantities or entities are present, some modifications by the user are usually necessary to arrive at an adequate layout as depicted in figure 2.14.

Toggle-buttons are supplied alongside the diagram which can be turned on or off to show or hide specific types of information, such as the type of dependencies (for an overview of the meaning of the different kinds of dependencies, see section 2.1). This way, also the value, the quantity space and the derivative of quantities can be shown. In addition, it is possible to show or hide the entities and attribute relations to clarify or disregard the system structure. A variation of the same view, in which all options all turned on, is shown in figure 2.15.

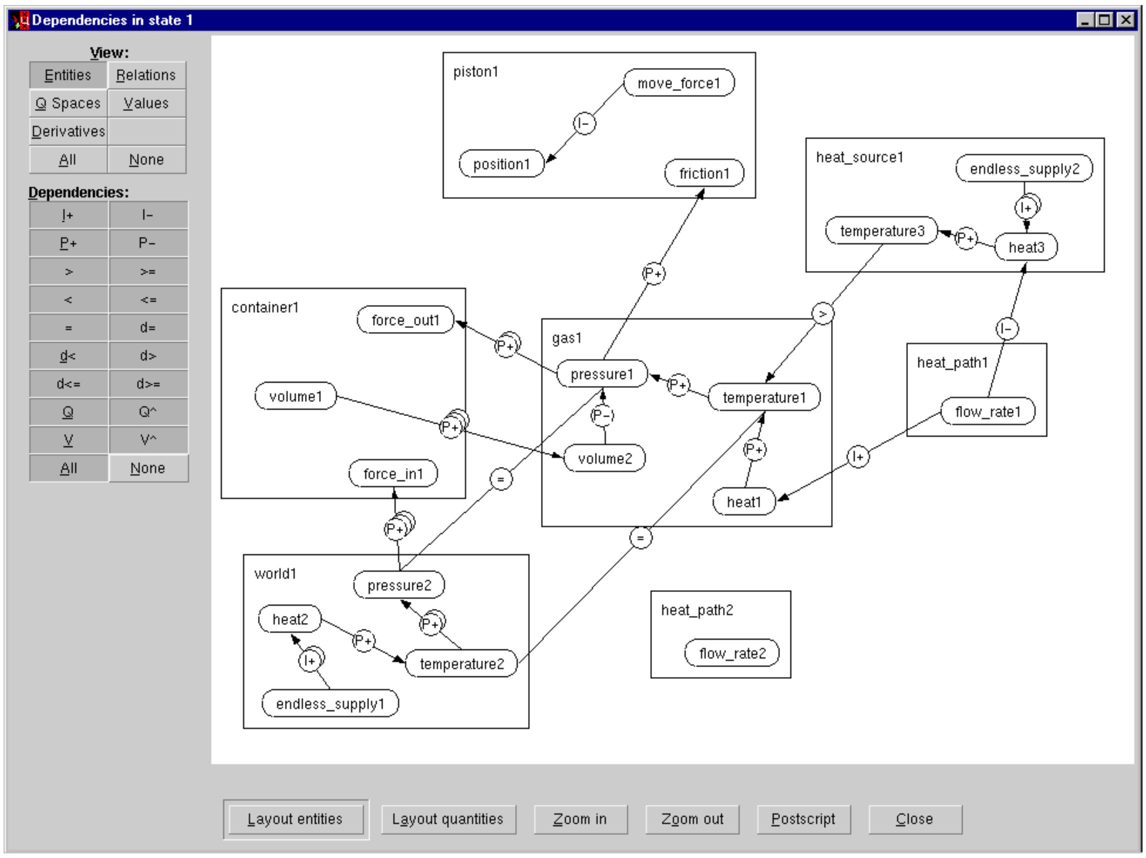


Figure 2.14
Dependencies view.

Entity: Rectangular box with text label positioned in the top left corner.

The entities are basically shown as in the entities and relations view discussed before. In the dependencies view, there are additional concerns, however. Because an entity often has multiple quantities attached to it, and containment is used to show this association, an entity node must be able to contain multiple quantity nodes. A rectangular shape, which can stretch out to fit the quantities contained, is most efficient for this purpose. Now, also the reason becomes apparent for the positioning of the text label in the top left corner (cf. section 2.3.5). If it would be in the middle of the entity box, it would get in the way of the quantities which might be displayed there.

The entities can be hidden from view by releasing the Entities button. This automatically releases the relationships button too, since it would be strange to see the relations without the entities being present.

Relationship: Labeled line with a dot on one end. See the entities and relations view.

In this view, they can be hidden by releasing the Relations button. Because the relationships do not make sense without entities present, pressing the Relations button automatically triggers the Entities button too.

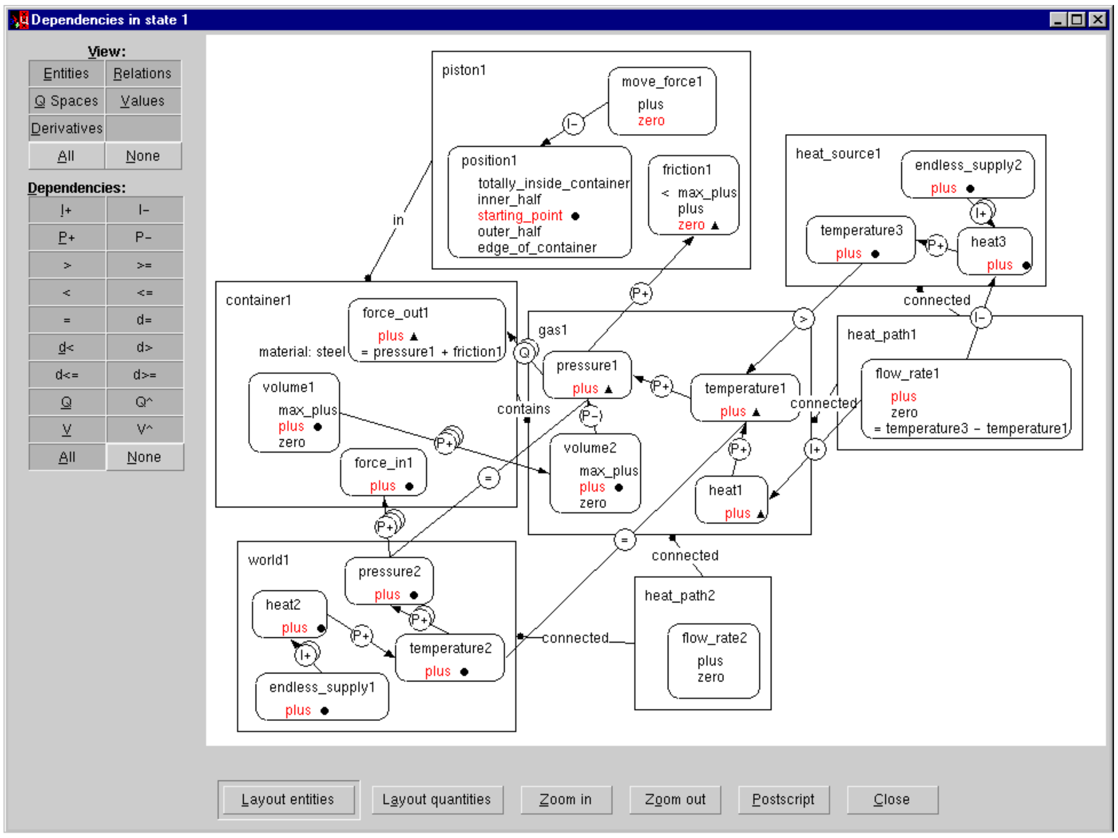


Figure 2.15
Dependencies view, with all display options turned on.

Quantity: Rounded rectangular shape.

Because there are often a lot more quantities than entities present, the shape of a quantity is made as small as possible, while still fitting the name label. However, the quantity node is enlarged (stretched downwards) when the quantity spaces are shown, as shown in figure 2.15. The edges of the quantity node's shape are rounded to distinguish quantity nodes from entity nodes.

Quantity Space: Vertically oriented list of value labels.

A quantity space often contains multiple values. These are always strictly ordered, usually from low to high values. These values are displayed ordered from bottom to top, so that low values are positioned low, and high values high along the vertical orientation. This corresponds to a natural interpretation of ’high and low’ in a two-dimensional space, although it contrasts with the western vertical reading direction fom top to bottom. Note that a horizontally oriented list would also be less practical, because that would require a much wider display, unless the value labels are displayed much smaller, or rotated, both of which would hamper readability. There is a distinction between point and interval values, as present in the GARP representation. This disctinction is lost in the vertical list representation, but it is still present in the representation of Quantity value histories. Using different colours, or emphasis for one of the types of values would have been possible, but this was considered too distracting in this type of view. The display of quantity spaces can be turned on and off by pressing/releasing the

Q Spaces button. Hiding the quantity spaces hides the quantity values and derivatives too, as well as value correspondences.

Quantity Value: Value label coloured red.

The current value of a quantity in a state, if known, is highlighted in red. When the value is unknown, no value is highlighted. More explicit ways of showing unknown values were considered, *e.g.*, by displaying a question mark (as in the quantity values view), but there does not seem to be a logical place to display it in this view without potentially creating confusion. The display of quantity values can be turned on and off by pressing/releasing the Values button. Turning the values on automatically turns on the quantity spaces as well.

Derivative: small black triangle (up/down) or circle.

To indicate the direction of change in the quantity space, a pointing symbol was desired. A triangle is the simplest pointy shape imaginable. Small arrows were also an option, but the triangles were preferred for aesthetic reasons. To indicate no change, *i.e.*, a zero-derivative, a circle is used, which does not point anywhere, and in fact, visually resembles the 0 for ’zero’. One could argue that ’no change’ could also be left implicit by not showing a derivative symbol, but this could create confusion: If there are no derivative symbols visible, would that mean that they are hidden, or are they all zero? Together with the circle, the triangles form an easily identifiable, but still relatively uniform set of shapes, which makes it clear that all three visualize the same kind of information.

Dependencies: lines or arrows connecting quantities with round symbolic labels.

Causal dependencies are directed relationships between two quantities and are therefore depicted as labeled arrows. The I-relations denote influences, P-relations denote proportional relationships. There are two versions of each (+ and -), specifying a positive or negative (inverse) relation. Together, they specify the causal model in the selected state of the simulation.

The other kinds of dependencies are mathematical in character: the equalities (undirected) and inequalities (directed) between values ($<, <=, =, >=, >$) and derivatives ($d <, d <=, d =, d >=, d >$) and the (directed or undirected) quantity and value correspondences (Q and V, respectively, where the addition of the ˆ-symbol indicates directedness). The directed relationships are depicted as labeled arrows, while the undirected dependencies are depicted as labeled lines between the quantities or values involved.

Exceptions are formed by d-(in)equalities involving a specific derivative, such as $dQ >= zero$, and (in-)equalities involving formulae, such as $Q1 = Q2 + Q3$. These are not shown as lines or arrows, as there are no nodes for specific derivatives or mathematical expressions. Instead, these dependencies are shown within (at the bottom, below the quantity space of) the quantity node they refer to as the result. For example, the formula $force\_out1 = pressure1 + friction1$ is displayed literally inside the quantity node $force\_out1$. This, and other examples of mathematical dependencies are shown in figure 2.16. An alternative representation for such formulas was considered, displaying it explicitly as a ternary relationship, by connecting a label ’=’ to $force\_out1$ on the one hand, and pressure1 and friction1 on the other, as shown in figure 2.17, taken from Arnbak et al. [2000]. A colour difference between the lines to the left and right argument is used to encode directional information (to distinguish $x - y$ from $y - x$).

Because it would not be intuitively clear how, and in which direction to read such a representation, this option was discarded in favour of the standard textual representation. To show that this textual

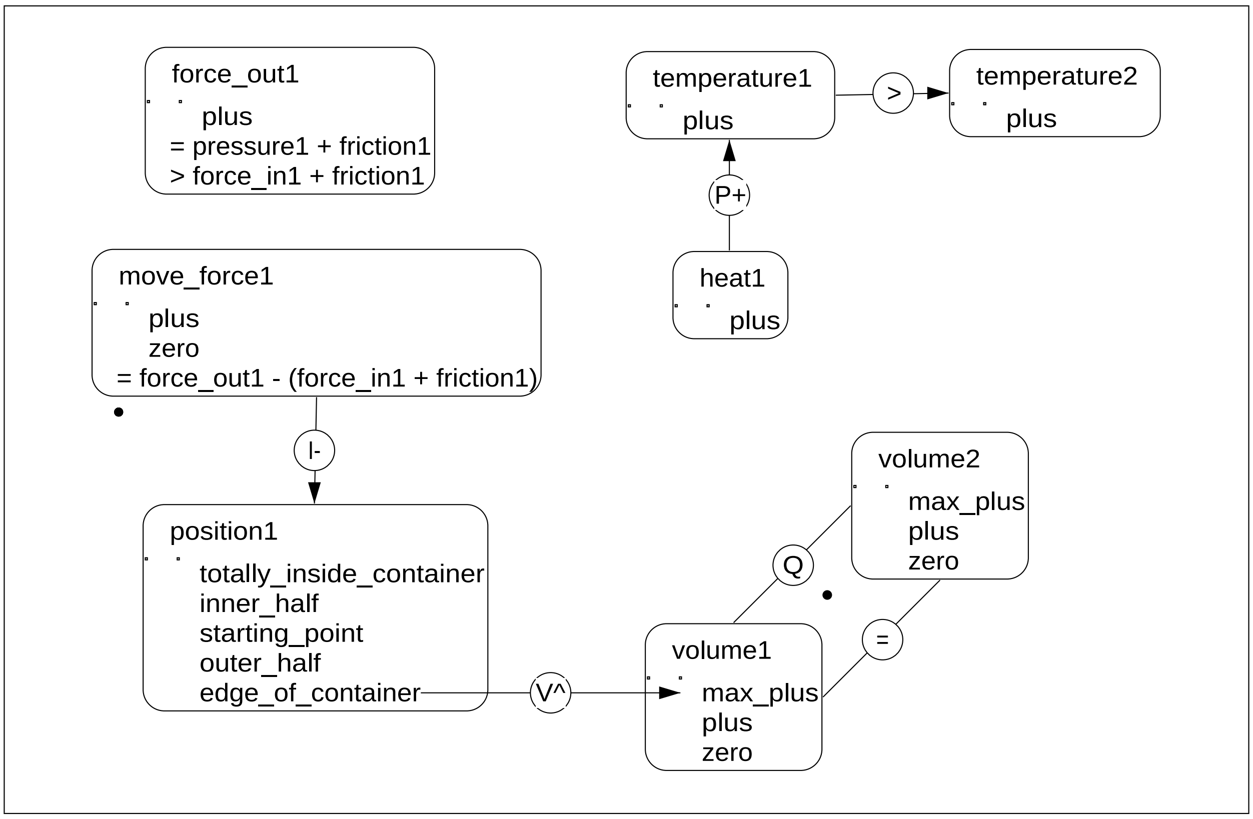


Figure 2.16
Examples of mathematical dependencies.

representation is still quite different from the original GARP propositional logic format, consider the original GARP code for some of the dependencies discussed above:

```
inf_neg_by( position1, move_force1 )
prop_pos( temperature1, heat1 )
greater( temperature1, temperature2 )
equal( move_force1, min( force_out1, plus( force_in1, friction1 ) ) )
dir_v_correspondence( volume1, max_plus(volume1),
                            position1, edge_of_container(position1) )
```

To read these original statements in GARP code, it is necessary to peel off the brackets to find the arguments inside, and to know the meaning imposed on the order of the arguments (which is reversed, in the case of the causal relationships).

Layout mechanism

The layout mechanism for the dependencies view deserves special attention. First, the entity nodes are created, with their quantities inside, just below each other. The entities are positioned using a spring-based layout mechanism [Battista et al., 1999] in XPCE, using the relations between entities (as in the Entities-relations view). Second, the dependencies are created. Another spring-based layout mechanism comes into action, which tries to shorten dependency links by moving quantity nodes

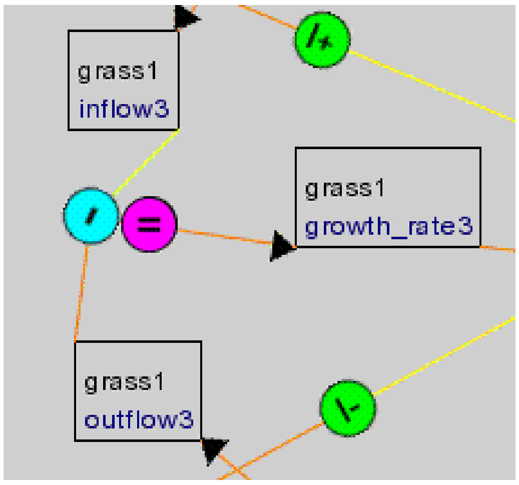


Figure 2.17
An alternative representation for an equality statement, in this case $growth\text{-}rate = inflow - outflow$, from Arnbak et al. [2000].

slightly. Dependencies between entities are considered as weaker links than dependencies within an entity. In this phase, the entities move only when all their quantities move. As the mechanism is non-deterministic (i.e., the result can be different each time), the result is often not ideal. Therefore, VisiGarp offers the user several ways to modify the layout further. The two steps performed by the algorithm can be repeated separately by clicking on the buttons *Layout entities* (to move the entities, without changing their internal quantity layout), or *Layout quantities* (which moves the quantities, mainly). Furthermore, the layout can also be adjusted by hand, as mentioned earlier, by dragging quantity and entity nodes. When moving entities, their quantities move with them. When moving quantities, their dependency links will follow - the encapsulating entity node will stretch if necessary.

Zoom mechanism

When a model contains many entities or quantities, it may become impossible to show them all on the screen at the same time without overlap. The scroll bars can be used to focus on a particular subset, but because users sometimes miss information on the screen, a zoom mechanism was added. By default, it checks whether there are graphical elements outside of view; if so, the distances between all nodes are compressed (without decreasing font size - to ensure readability at all times) to make everything fit within the screen. This may cause overlap, but by zooming in on a particular entity, the distances are increased again causing the overlap to disappear.

2.3.8 The quantity values view

The quantity values view shows all quantity values and derivatives for a selected state in a plain table format. This takes up less space than the dependencies view and can therefore be convenient when the user is not so much interested in the dependencies. Unknown values and derivatives are displayed as a ’?’ in this view. In the table format adopted for this view, there is not much danger that the ’?’ would create confusion, and leaving a cell empty would be unusual.

2.3.9 The quantity value history view

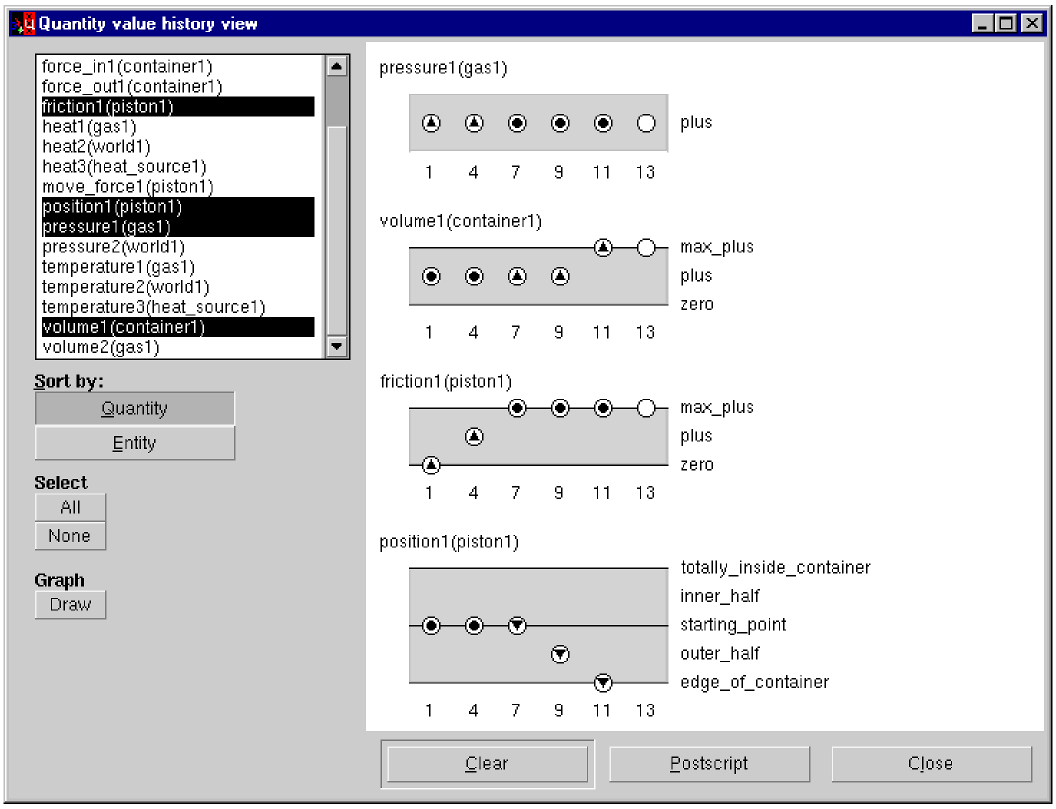


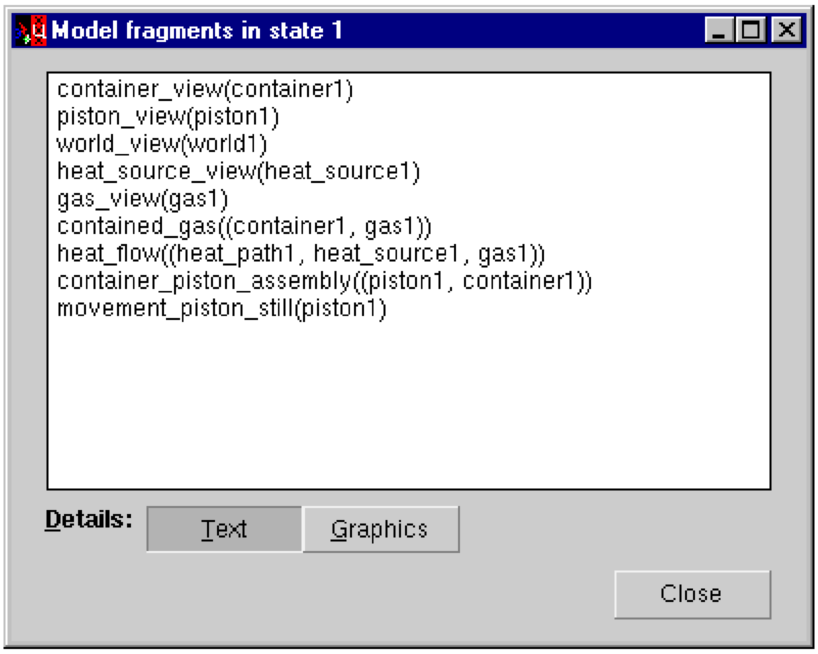
Figure 2.18
Quantity value history view.

In the quantity value history view (see figure 2.18), quantities can be selected from a list on the left, which is sorted by quantity, or entity. When the Draw button is pressed, the values of the selected quantities will be plotted over time (following the sequence of states selected in the behaviour graph). A graph format is used with the quantity space displayed as the grey value space with black lines, with the quantity space value names listed along the vertical axis. Along the horizontal axis, the states are displayed, ordered according their occurrence in a path, if a path was selected.

Values are represented by the circles on a line, or between lines, to denote a point or interval value, respectively. Derivatives are shown using the derivative symbols, as in the dependencies view (see 2.3.7), but here, they are displayed inside the value circles. Theoretically, there would be a problem when the derivative is known while the value is unknown, but this should not occur in practice.

Note that the value points are not connected. Connecting them would create a kind of line graph, but it was felt that this would suggest too much, because the slope of ascent/descent within intervals is unknown in the GARP qualitative simulation framework. In QSIM, straight dotted lines *are* added to the visualization of value histories, but with the comment that ’dots connecting symbols have no significance’ [Kuipers, 1994, p. 23]. Leaving the dots unconnected is more in line with the principle of not showing information which is unknown.

2.3.10 The model fragments view



```
container_view(container1)
piston_view(piston1)
world_view(world1)
heat_source_view(heat_source1)
gas_view(gas1)
contained_gas((container1, gas1))
heat_flow((heat_path1, heat_source1, gas1))
container_piston_assembly((piston1, container1))
movement_piston_still(piston1)
```

Figure 2.19
Model fragments applicable in state 1.

To investigate the role of the different model fragments during a simulation, a view is supplied which lists all model fragments which apply in a particular state (see figure 2.19). Such a list gives a high-level overview of what is true, and what is happening in that state. A model fragment can be selected, and more details can be requested in a structured text format, or a graphical format. These views are only accessible from the Model fragments view.

2.3.11 The model fragment graphics view

The grapical format is based on the causal model view (shown in figure 2.20), with colour used to highlight the contents of the specific model fragment (red for conditions, and blue for the results).[5]

This gives an overview of the context while drawing special attention to the specific knowledge introduced by that model fragment. Note that the graphical format alleviates the need for repetition of information elements, as is the case in the text format. Second, the graphics format includes also contextual information in the background (in grey), to support integration of the specific information of the particular model fragment into the model as a whole. The figure shows the model fragment *heat-flow*, which applies when there is a temperature difference between two heat exchanging entities and a heat-path connecting the two. It introduces a positive flow-rate, specified as the temperature difference, which positively influences the heat of the colder object (gas1) and negatively influences the heat of the hotter object (heat-source1). Two things are missing in this view, however: conditional,

[5]The use of these colours has changed during the development of VisiGarp. In VisiGarp 0.9, blue was used for the conditions and red for the results, but VisiGarp 1.0 and later versions use red for the conditions and blue for the results, to make them correspond to the use of the colours in the qualitative modelling tool Homer [Jellema, 2000].

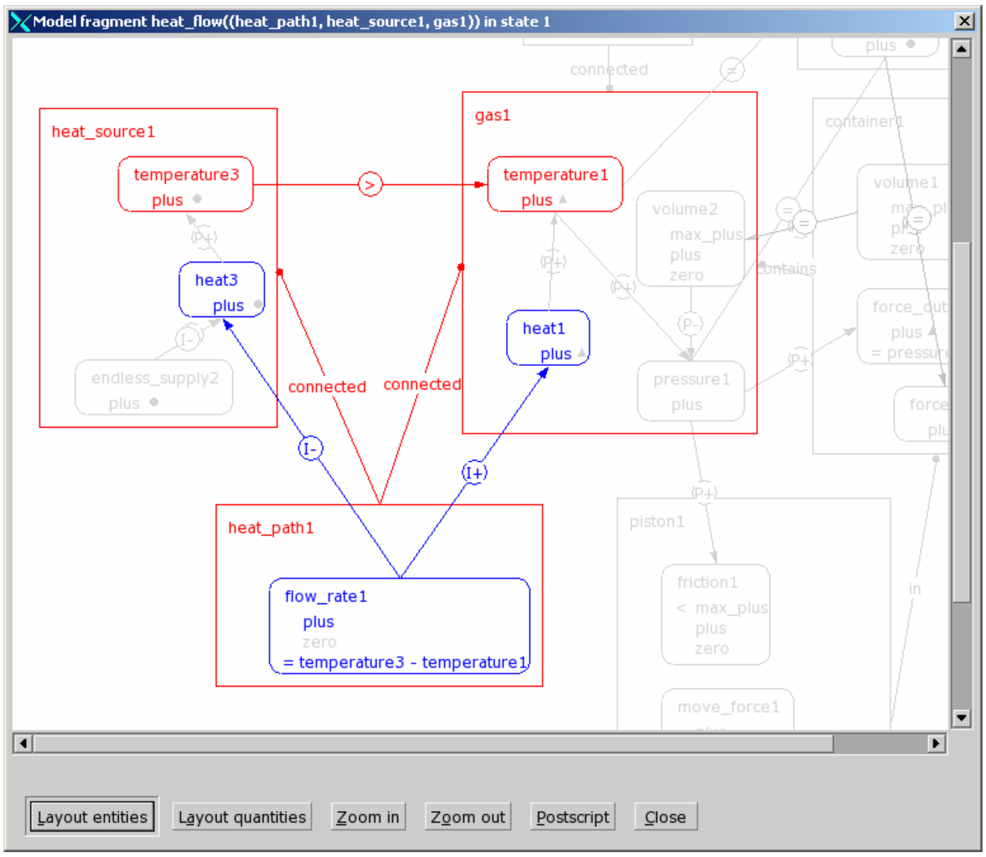


Figure 2.20
Model fragments graphics view.

and supertype model fragments are not displayed. They were deliberately left out, because it is unclear how to display the contents of these other model fragments (which often overlap), while keeping the focus on the selected model fragment.

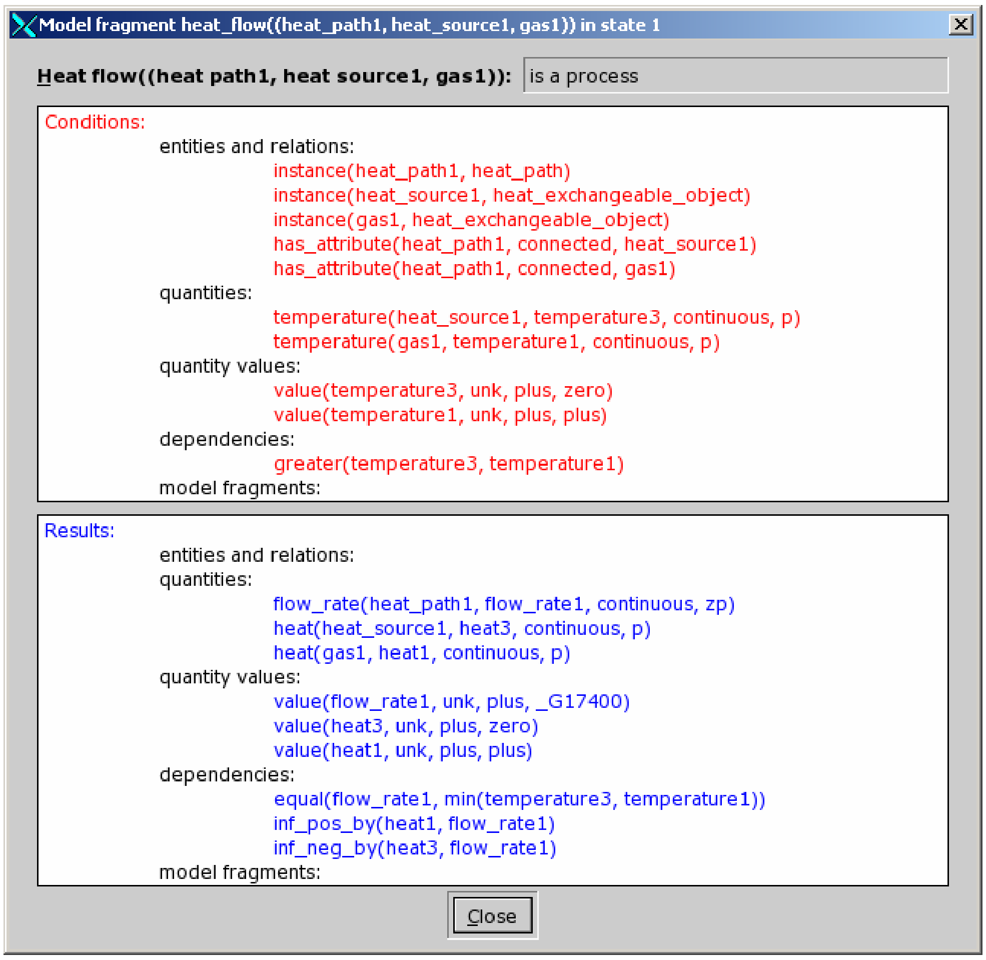
2.3.12 The model fragment text view

The textual format, shown in figure 2.21, is more compact than the graphical format and gives a concise overview of what types of knowledge are introduced. The text format vertically distinguishes the conditions (shown at the top, in red) from the results (shown at the bottom, in blue). For both conditions and results, the different knowledge categories are shown, with the actual content indented.

In contrast to the model fragment graphics view, the text view does include the conditional and supertype model fragments, by listing the name labels of the model fragments involved behind the ’model fragments’ category indicator. In the example, however, there are no conditional or supertype model fragments.

2.3.13 The model fragment is-a hierarchy view

It is also possible to view how the different model fragments are structurally related to each other in the model library. The is-a hierarchy of model fragments (see figure 2.22) shows the hierarchical subtype-relationships between model fragments, in the same horizontal format with collapsible nodes as the


Model fragment heat_flow((heat_path1, heat_source1, gas1)) in state 1

Heat flow((heat path1, heat source1, gas1)): is a process

```
Conditions:
          entities and relations:
                    instance(heat_path1, heat_path)
                    instance(heat_source1, heat_exchangeable_object)
                    instance(gas1, heat_exchangeable_object)
                    has_attribute(heat_path1, connected, heat_source1)
                    has_attribute(heat_path1, connected, gas1)
          quantities:
                    temperature(heat_source1, temperature3, continuous, p)
                    temperature(gas1, temperature1, continuous, p)
          quantity values:
                    value(temperature3, unk, plus, zero)
                    value(temperature1, unk, plus, plus)
          dependencies:
                    greater(temperature3, temperature1)
          model fragments:

Results:
          entities and relations:
          quantities:
                    flow_rate(heat_path1, flow_rate1, continuous, zp)
                    heat(heat_source1, heat3, continuous, p)
                    heat(gas1, heat1, continuous, p)
          quantity values:
                    value(flow_rate1, unk, plus, _G17400)
                    value(heat3, unk, plus, zero)
                    value(heat1, unk, plus, plus)
          dependencies:
                    equal(flow_rate1, min(temperature3, temperature1))
                    inf_pos_by(heat1, flow_rate1)
                    inf_neg_by(heat3, flow_rate1)
          model fragments:
```

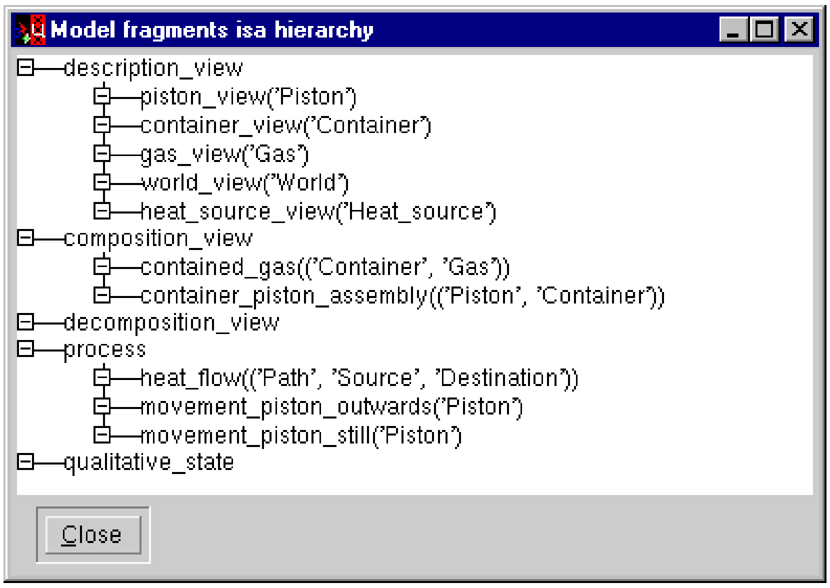
Figure 2.21
Model fragments text view.

entity is-a hierarchy. The figure shows that in this domain library, there are five description view model fragments specifying information about singular entities, two composition views specifying information about multiple entities, and three process model fragments.[6] A model fragment can be selected for further inspection.

2.3.14 The model fragment applies-to hierarchy view

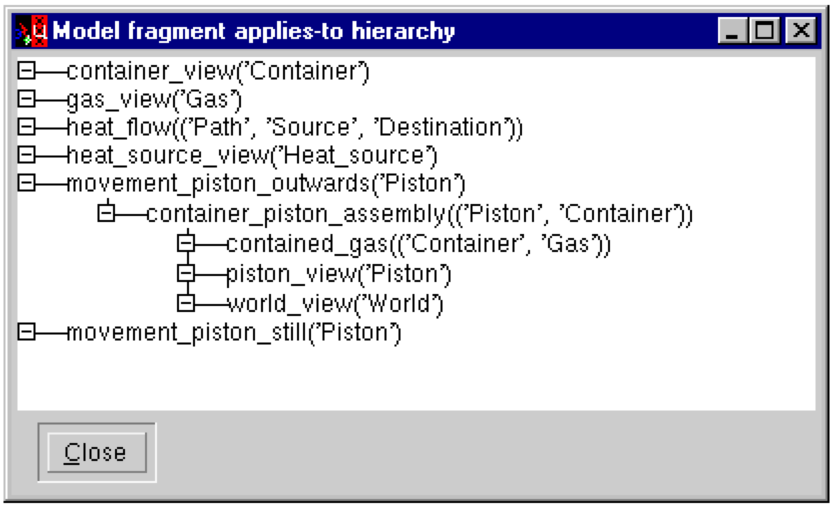
The applies-to hierarchy (shown in figure 2.23) shows which model fragments are conditional for which other model fragments, in the same way as the entity- and model fragments is-a hierarchies. The figure shows that the model fragment movement_piston_outwards applies to situations in which there is a model fragment container_piston_assembly, which in turn requires three other model fragments to hold. Here too, a model fragment can be selected for further inspection.

[6]Because this kind of knowledge is more abstract than the kinds described previously, it should not be used before students have had some experience with the more concrete aspects of the simulation model. It is especially useful, however, for the system modeller, or students learning to model.



```
⊟—description_view
        ⊟—piston_view('Piston')
        ⊟—container_view('Container')
        ⊟—gas_view('Gas')
        ⊟—world_view('World')
        ⊟—heat_source_view('Heat_source')
⊟—composition_view
        ⊟—contained_gas(('Container', 'Gas'))
        ⊟—container_piston_assembly(('Piston', 'Container'))
⊟—decomposition_view
⊟—process
        ⊟—heat_flow(('Path', 'Source', 'Destination'))
        ⊟—movement_piston_outwards('Piston')
        ⊟—movement_piston_still('Piston')
⊟—qualitative_state
```

Figure 2.22
Model fragments is-a hierarchy.

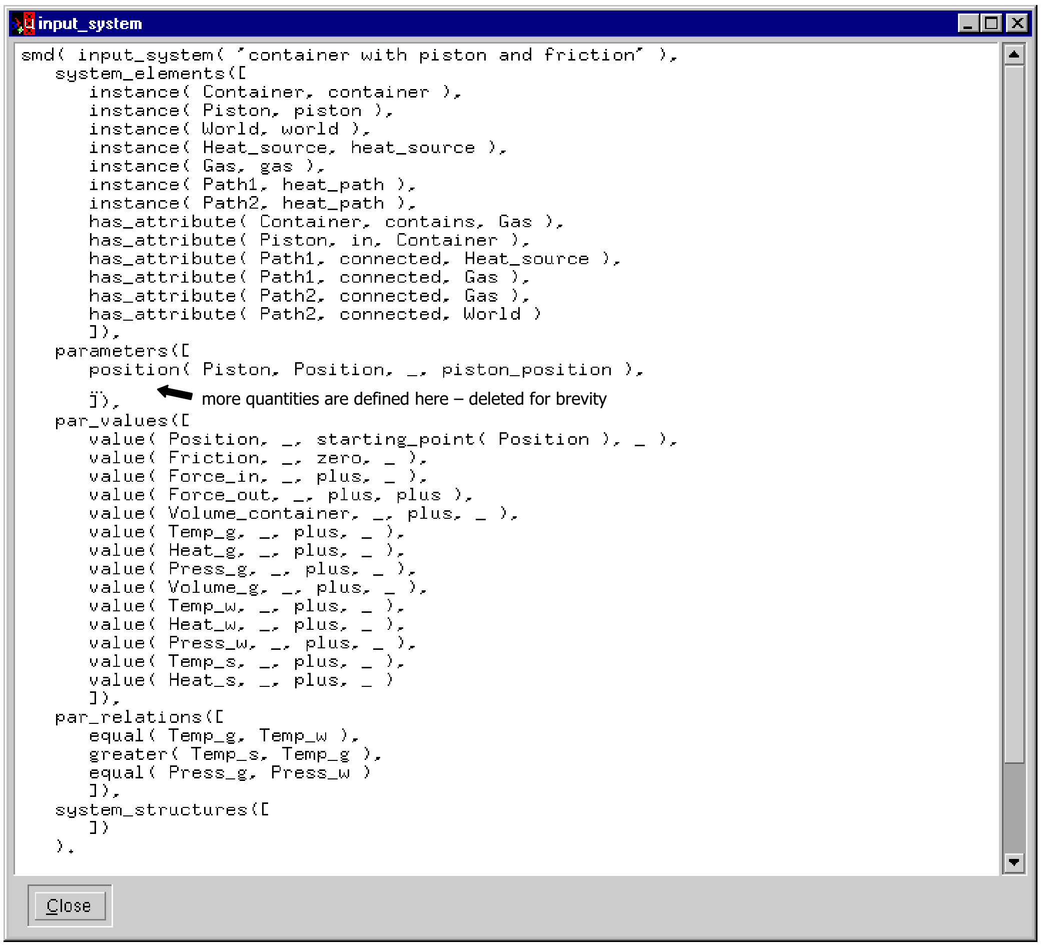


```
⊟—container_view('Container')
⊟—gas_view('Gas')
⊟—heat_flow(('Path', 'Source', 'Destination'))
⊟—heat_source_view('Heat_source')
⊟—movement_piston_outwards('Piston')
        ⊟—container_piston_assembly(('Piston', 'Container'))
                ⊟—contained_gas(('Container', 'Gas'))
                ⊟—piston_view('Piston')
                ⊟—world_view('World')
⊟—movement_piston_still('Piston')
```

Figure 2.23
Model fragments applies-to hierarchy.

2.3.15 The scenarios view

The scenarios view displays all scenarios available in the domain model library. No graphical visualization was designed because it was not deemed necessary for the target users (learners). It is considered sufficient to show the names of scenarios to choose from when opening a new simulation, and it is possible to view more details of the chosen scenario by clicking on the input model (scenario) 'state' in the state-transition graph and opening the Entities and relations, Quantity values, or Dependencies view. For completeness, the scenario view is available, which simply shows the original code file with all scenarios in the propositional GARP format, including comments from the model builder.



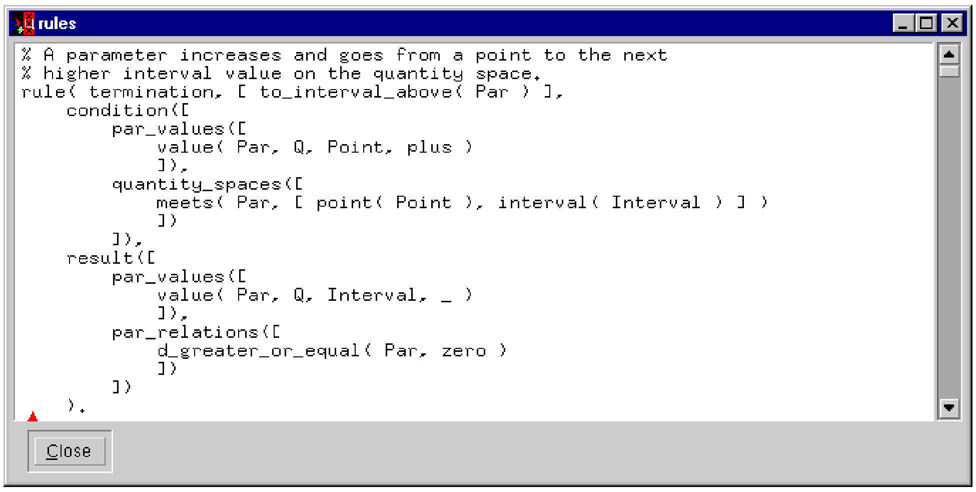
```
smd( input_system( 'container with piston and friction' ),
   system_elements([
      instance( Container, container ),
      instance( Piston, piston ),
      instance( World, world ),
      instance( Heat_source, heat_source ),
      instance( Gas, gas ),
      instance( Path1, heat_path ),
      instance( Path2, heat_path ),
      has_attribute( Container, contains, Gas ),
      has_attribute( Piston, in, Container ),
      has_attribute( Path1, connected, Heat_source ),
      has_attribute( Path1, connected, Gas ),
      has_attribute( Path2, connected, Gas ),
      has_attribute( Path2, connected, World )
      ]),
   parameters([
      position( Piston, Position, _, piston_position ),
      ]),
   par_values([
      value( Position, _, starting_point( Position ), _ ),
      value( Friction, _, zero, _ ),
      value( Force_in, _, plus, _ ),
      value( Force_out, _, plus, plus ),
      value( Volume_container, _, plus, _ ),
      value( Temp_g, _, plus, _ ),
      value( Heat_g, _, plus, _ ),
      value( Press_g, _, plus, _ ),
      value( Volume_g, _, plus, _ ),
      value( Temp_w, _, plus, _ ),
      value( Heat_w, _, plus, _ ),
      value( Press_w, _, plus, _ ),
      value( Temp_s, _, plus, _ ),
      value( Heat_s, _, plus, _ )
      ]),
   par_relations([
      equal( Temp_g, Temp_w ),
      greater( Temp_s, Temp_g ),
      equal( Press_g, Press_w )
      ]),
   system_structures([
      ])
   ).
```

Figure 2.24
Scenarios view.

For an example of the scenarios view showing a fragment of the scenarios file, see figure 2.24.

2.3.16 The transition rules view

The transition rules view shows all possible transition rules in the domain model library. Because this is not considered very useful information for the target audience, no effort was made to design a visualization. A view of the original GARP code file is included (see figure 2.25 for a fragment of this file) for more experienced GARP users. When transition rules are used in a simulation, the instantiated versions are available for inspection using the Transition history and details views.



```
% A parameter increases and goes from a point to the next
% higher interval value on the quantity space.
rule( termination, [ to_interval_above( Par ) ],
    condition([
        par_values([
            value( Par, Q, Point, plus )
            ]),
        quantity_spaces([
            meets( Par, [ point( Point ), interval( Interval ) ] )
            ])
        ]),
    result([
        par_values([
            value( Par, Q, Interval, _ )
            ]),
        par_relations([
            d_greater_or_equal( Par, zero )
            ])
        ])
    ).
```

Figure 2.25
Transition rules.

2.4 Using VisiGarp in an educational context

VisiGarp is a simulation model-inspection tool designed for students without expertise in qualitative reasoning. After a learning phase in which students should get acquainted with VisiGarp, it is expected that they are able to learn about the structure and behavioural characteristics of any system modelled in VisiGarp, provided that it is not too complex, and at least some of the concepts used are grounded in their experience. The fact that the visualizations do not make use of domain-specific graphics has the advantage that the system can be used in many different domains. Currently, there are models available in the domains of thermodynamics, elementary physics, fluid dynamics, climate and weather modelling, biology and ecology [Bredeweg, 1992, Salles et al., 1997, Salles and Bredeweg, 2001a, Dubbeldam, 2003, van der Werf, 2003, Koeman, 2003].

One of the educational goals that VisiGarp can support, is learning to make qualitative predictions and to test their accuracy. Given a particular scenario, a student may be asked to predict what will happen to a certain quantity of interest. Using VisiGarp, he or she can then run the simulation and check the hypothesis by looking at the value histories for paths leading to the end state(s) in the simulation. A common mistake is to consider only one option, when in fact multiple alternatives are possible. A second educational goal is to explain results occurring in a simulation. For example, a student may be given an assignment to explain why a certain quantity is increasing in a particular state of the simulation. After an initial guess, this requires opening the dependencies view in VisiGarp to investigate which other quantities have affected the quantity in question. There may be a whole chain of causal relationships which has to be traced back to the beginning to find the original cause. When students have a good understanding of the basics (e.g., states, entities, quantities, dependencies), they may be stimulated to investigate the role of model fragments in the simulation and consequently, the structure of the model as a whole. As model fragments are responsible for introducing new quantities, dependencies, etc., it can be useful to know under which conditions a certain model fragment applies. When a model is designed in a structured manner, with educational use in mind, learning goals may correspond to one or more model fragments containing particular chunks of information, *e.g.*, a process or particular situation description.

Besides learning about the behaviour of systems in a certain domain, VisiGarp may also support higher-level educational goals. By investigating simulation models in different domains, students may learn general principles of causality, as well as skills related to prediction and conceptual modelling of system behaviour. Using VisiGarp, students may also learn to make effective use of diagrams as external reasoning aids. Some characteristics of system behaviour can be directly read off from the diagrams. For example, by scanning for simple visual patterns, students can find out whether there is ambiguity in the simulation, or a feedback loop in the causal model. An overall estimation of complexity and relative importance can be given by glancing at the diagram layout: a diagram with lots of arrows going back and forth suggests a complex system; clutter around a certain quantity may suggest that it serves an important role in the system. These kinds of reasoning with diagrams can prove very useful because they go beyond the possibilities of text as a medium for knowledge communication. It is also important to note that the diagrams in VisiGarp are created at the moment a student wants to see them, in the context he or she has chosen. They can be extended and modified at the student's will in various ways: adding extra details, extending behaviour paths, highlighting fragments, modifying the layout. Hopefully, this will lead students to see that diagrams are not necessarily static figures (as is the case in books), but flexible external representations which you can interact with to facilitate reasoning.

As a note of caution, a flexible tool such as VisiGarp, with many views, menu choices and buttons to choose from, carries the danger of the 'art museum' problem [Foss, 1989]–students may be overwhelmed by the amount of information presented to them, and keep clicking on things without real understanding of what they are doing or what they are looking at. Therefore, support is necessary to keep students on track and create a useful learning experience [van der Hulst, 1996, Veermans, 2002]. There are several ways to implement such support, *e.g.*, general advice, personalized advice based on student modelling and diagnosis, system-initiated explanations, and exercises. Ideally, the visualization capabilities of VisiGarp should be integrated in an intelligent learning environment which incorporates mixed-initiative dialogue which includes instruction and assessment, following a curriculum of learning goals. Chapter 4 deals with the issues related to this kind of functionality. In the next section, however, VisiGarp is evaluated as a standalone learning environment, with a set of *exercises* to guide students' explorations through a set of simulations.

2.5 Evaluation of VisiGarp

As VisiGarp has been fully implemented, it allows testing whether it fulfills the goals it was designed for. Ideally, the questions to be answered by evaluation should include the following:

- Is VisiGarp easy to (learn to) use?
- Are the visualizations understandable?
- Does VisiGarp support learning about the domain?

As the design is centered around the generation of visualizations and organizing them in a graphical user interface, the focus of the evaluation is more on the first two questions regarding usability, rather than the third question, which concerns the educational value.

VisiGarp has been evaluated by different people in different contexts. First, two pilot studies are described: one in which twenty-five third-year undergraduate students interact with VisiGarp for about thirty-five minutes, and an expert review by a physics teacher, consisting of interacting with VisiGarp and an interview (see section 2.5.2). After these pilot studies, some improvements were made to VisiGarp and the experimental materials. Then, a second evaluation experiment was performed, with thirty first-year undergraduate students who interact with VisiGarp for about an hour and a half. This study is described in section 2.5.3. Finally, an overview is given of how VisiGarp is being used in practice in universities in Amsterdam, The Netherlands, and Brasilia, Brasil.

2.5.1 Test Domain: The Ecology of the Brazilian Cerrado Vegetation

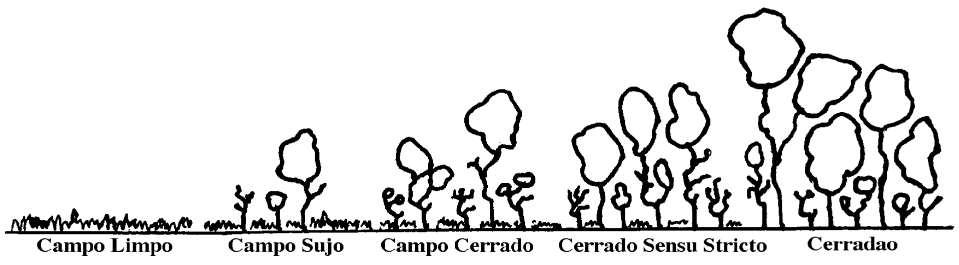


Figure 2.26
Ecological communities occurring in the Cerrado vegetation.

The test domain chosen for evaluating VisiGarp is the ecology of the Brazilian Cerrado vegetation [Salles, 1997]. In this type of vegetation, several ecological communities can occur, consisting of different plant populations of varying size (see figure 2.26). The numerous different types of plants are categorized as either grass, shrubs, or trees. This allows a classification of vegetation types, ranging from Campo Limpo, consisting of mainly grassland, to Cerradao, a vegetational state with mainly trees and shrubs, and (almost) no grass. The Cerrado Succession Hypothesis suggests that a Campo Limpo may develop towards a Cerradao under the right circumstances, in terms of natural conditions and human actions, such as fire management.

The Cerrado domain has been modelled qualitatively in the GARP framework [Salles et al., 1997, Salles and Bredeweg, 2001a], which makes it directly available for use in VisiGarp. One of the simulation scenarios models the Cerrado Succession Hypothesis. The goal of the model is to provide insight in the causal mechanisms involved in the development and management of the Cerrado. This is considered especially useful for ecology students and future decision makers in the field of sustainable development.

2.5.2 Evaluation of VisiGarp 0.9: The Two Pilot Studies

The first, formative, evaluation sessions are called pilot studies because VisiGarp was still undergoing some development at this stage and was expected to be improved. Its version number, 0.9, is meant to indicate this status.

Evaluation by twenty-five students

Twenty-five third-year undergraduate students from the department of Social Science Informatics at the University of Amsterdam took part in the first pilot study. Participation in the study was mandatory, as a part of a course on evaluation of software systems. The study consisted of an introductory presentation about QR and VisiGarp (30 min.), ten exercises to carry out using VisiGarp (ca. 35 min.), and two questionnaires, of which one was aimed at usability in general; the other was intended to test whether the graphical representations could be understood.

Results. The results indicated that students could use VisiGarp to run a simulation, investigate paths of transitions, find out the amount and direction of change in quantities, and determine causal paths leading to such changes, in a domain new to them. These are very positive findings, given the short time they spent using the system. However, there were also points of criticism. The settings for the spring-based layout algorithm were not optimal, causing chunks of graphical elements to be scattered over the screen too widely, sometimes partly out of sight. Consequently, the participants had to spend a considerable amount of time scrolling and dragging to manipulate the layout of the visualizations. This was reflected in the scores on the feedback questionnaire. Diagram layout was rated relatively low: 2.16 on average (SD = 1.25), on the answer scale of 1 (negative) to 5 (positive). Some participants complained they were not given enough time to interpret the domain knowledge presented to them, beyond the level of reading off information from the diagrams. This is not too surprising, given the short time available and their unfamiliarity with the domain. Some participants rated the qualitative modelling primitives as hard to understand (five people scored the minimum score of 1), but there was a lot of variation in the scores for this question (2.88 on av., SD = 1.42). More details of the experiment can be found in Tjaris [2002].

Expert review by a physics teacher

The second pilot study involved a physics secondary school teacher, who did not have any expertise on qualitative reasoning or the domain of Cerrado ecology. He did have experience with using educational software, including some (mathematical) simulation tools.

The teacher was given a short (5 min.) introduction to the goals of this research. No explicit instruction about QR or VisiGarp was given, except for a paper user guide. Then, he was instructed to act as a student, and given a series of exercises to be answered using VisiGarp. These exercises were organized around three different scenarios, and ranged from simple questions about a small simulation (partly to get used to the VisiGarp interface), via intermediate questions about a slightly larger simulation, to more complex questions about the Cerrado simulation. During this session, the experimenter sometimes took over to demonstrate some particular model characteristics or useful capabilities of VisiGarp. After the interaction with VisiGarp, the teacher was interviewed about his experiences with and attitudes towards VisiGarp. Both the interaction and interview were videotaped for analysis.

Results. The teacher did not complete the whole series of exercises using VisiGarp. As the list of exercises was designed to go from simple to complex, this means that the more challenging material was not covered. It is clear that the list of questions was too long, and contained too many simple questions in the beginning. The teacher shared the students' concern about the poor layout of the diagrams, and at first, he had a few misconceptions about the visualizations (e.g., interpreting states as entities, and a transition arrow as an implication). His understanding improved over the course of the session, though, and he also made some positive remarks about the visualizations. He judged the visualization of values inside quantity spaces inside quantities as ”*very logical*”, and called the diagrams ”*quite clear*” in general. When asked whether he used diagrams like these in his teaching already, an important point he made (which will receive more attention in Chapter 3) was the following:

> ”And something along those lines I use myself too, but certainly when you have many factors and influences, then I prefer, whatever can be left out the picture, I'll leave out, so that some more overview remains.”

To help students get used to the interface, the teacher suggested to let them play around with the system for a while, with help provided on screen, rather than working through a set of questions on paper.

Aside from the aspects of the visualization and interface of VisiGarp, also the educational potential of tools like VisiGarp was discussed. In this respect, the teacher worried about a lack of embedding in an educational context. He said he would expect an introductory story, or case-study, to establish the context and relevance of the learning goals.[7] Possibly, this perceived deficiency was partly due to the fact that this teacher was specialized in another subject domain (physics) than the sample domain used (ecology). His sense of case-study seems to match with GARP simulation scenarios, but it seemed he expected human actions to play a more important role in the simulation to arrive at some desired end situation.

When asked whether he knew of any programs similar to VisiGarp, he said:

> ”Well, no, not really. That’s something I like again. Of course, there are very many simulation programs, which I know from my own field, which are no doubt based on something like that, with lots of things being calculated and represented graphically, but that’s mostly something moving–a short film fragment of something, or a java applet, ..., but then it’s immediately quantified.

He also expressed some thoughts on extending the functionality of tools like VisiGarp:

> ”It would be nice to have a program that you could fill with data yourself, and ... could eventually pose questions to the students, e.g., what-if situations ... and something like that does not yet exist, as far as I know.”

These kinds of considerations will be discussed further in Chapter 4 of this thesis, which deals with embedding information in qualitative simulations in an educational dialogue interaction.

Discussion

Overall, the results and comments from the pilot studies indicate that VisiGarp is considered a useful tool to investigate predictions, and (to a lesser extent) explain causal events, especially when users are given a little more time to learn to use and experiment with the system. It is clear that some improvements are desired, however. First, VisiGarp needs more work on the layout and navigation mechanisms, to reduce the time spent on manipulating the visualizations. Second, translation of the English terms (both domain-specific and generic QR concepts) into Dutch (most participants’ native language) could help to improve understanding of the qualitative reasoning primitives and Cerrado domain-specific terms. Third, the set of exercise questions needs to be refined, in order to provide a more focused and meaningful learning experience.

[7] As such material is very domain-specific in nature, this was deliberately left out of the experimental materials as much as possible, leaving this to the teachers who plan to use the tool.

2.5.3 Evaluation of VisiGarp 1.0 by thirty first-year students

After the pilot studies, adjustments were made to VisiGarp, the domain model, and the experimental setup to address some of the shortcomings found during the pilot studies.

2.5.3.1 VisiGarp 1.0

Compared to version 0.9, a translation function (and a three language-vocabulary) was added to VisiGarp to be able to switch between English, Dutch, and Portuguese as the language for all domain terms. Also, the layout algorithms for the state-transition diagram, entities and relations view and the dependencies view were improved. Now, the dependencies diagram is always compressed to fit on the screen on first display. Because the font size of text labels can not be resized without making the text unreadable, the font size is kept constant. This means that there is a minimum size for quantities and entities, beyond which further compression only moves the objects closer together. Therefore, graphical objects can overlap in the case of models containing many elements. This is still suboptimal, but it is considered a better option than to run the risk of leaving things out of view, which could happen in VisiGarp 0.9. To give the user more control over the layout and navigation, zoom buttons (zoom in and out) were added to the interface. These work in such a way that the selected entity remains in focus.

2.5.3.2 Simulations

The domain library used in this study is version 2 of the Cerrado Succession Hypothesis (CSH) model used in the pilot study. It includes an extensive set of scenarios, from single populations involving a few processes, via interactions between two populations, to more complex scenarios about the Cerrado as a whole [Salles and Bredeweg, 2001a]. Two scenarios were selected for this study: a simple one, with one plant population and the influence of four basic processes (birth, death, immigration and emigration), and a complex scenario based on the CSH. Besides the basic processes, this more complex scenario also includes environmental factors such as soil temperature, fertility, light, cover, moisture, nutrients, and fire frequency. The complexity of both simulation scenarios in terms of the number of elements is shown in table 2.2.

	Simulation 1	**Simulation 2**
# states	8	19
# entities & relations	7	39
# quantities	8	32
# dependencies	12	127

Table 2.2
The complexity of simulation 1 and 2.

2.5.3.3 Method

Participants. In this study, thirty first-year psychology students participated (seven male, twenty-three female), who were given study credit points as reward.

Materials. Based on the experiences in the previous two studies, the experimental materials were completely redesigned, to progress more systematically through the material, and to allow controlled investigation of what people knew before and after the treatment. The translation option was set to present the model in Dutch. The experimental setup was as follows:

1. Short introduction by the experimenter.
2. Paper-based pre-test of domain-specific knowledge about the Cerrado.
3. Paper-based pre-test of diagrammatic representations as shown in VisiGarp
4. Treatment: interaction with VisiGarp, about two scenarios:
   - a. Simulation 1: answering questions on-line about one plant population
   - b. Simulation 2: answering questions on-line about the Cerrado vegetation
5. Paper-based post-test of domain-specific knowledge about the Cerrado.
6. Paper-based post-test of diagrammatic representations as shown in VisiGarp.
7. Paper-based questionnaire about the subject's attitudes towards VisiGarp.
8. Short time for additional comments and questions.

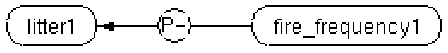
Each of these steps is described in more detail below. When multiple choice questions are involved, the correct answer is displayed in italics.

1. Short introduction by the experimenter. In a few minutes, the goals of VisiGarp are explained and the experimental procedure is described.
2. Paper-based pre-test of domain-specific knowledge about the Cerrado. To test and compensate for any possible difference in difficulty level between the pre- and post-test, two versions of the test were created, CSH-A and CSH-B. They were intended to contain equivalent questions, with some elements changed. Half of the participants received test CSH-A, the other half received test CSH-B as pre-test. An example question from this test is given below.

   CSH Domain test CSH-B, question number 4.
   Does regulation of the fire frequency within the Cerrado vegetation (for example by appointing a manager who can increase or decrease the fire frequency) have an effect on the tree population?
   a. Yes, if the fire frequency is increased, the tree population will increase
   b. Yes, if the fire frequency is increased, the tree population will decrease
   c. No, the fire frequency and the tree population are independent of each other
   d. No, the fire frequency can not be regulated

3. Paper-based pre-test of diagrammatic representations as shown in VisiGarp. Here too, there are two versions of the test, DIA-A and DIA-B. Half of the participants received test DIA-A, the other half received test DIA-B as pre-test. An example question from the diagrams test is given below:

Diagrams test DIA-A, question number 7.



What is the relationship between litter1 and fire-frequency1?

a. If fire-frequency1 decreases, litter1 increases

b. If litter1 increases, fire-frequency1 increases

c. If litter1 decreases, fire-frequency1 increases

4. Treatment: interaction with VisiGarp, about two scenarios. A simulation scenario is loaded into VisiGarp, and the participant is asked to follow the instructions and answer the questions that are presented on the screen, in a window separate from VisiGarp. The participant could switch back and forth between the question window and the VisiGarp windows. When the participant answers a question (by clicking one of the options a, b, or c), VisiGarp records the answer and displays the next question. For each answer, a time-stamp was created, to give an indication of the time spent on reading, interpreting and answering each question. Participants were also asked to rate the difficulty level (on a scale from 1=easy to 5=difficult) of the questions after each scenario. Two help documents were supplied, one about the VisiGarp interface as a whole, and one sheet which contained a list of all types of dependencies and their meaning. The VisiGarp interaction session contained 27 questions in total, divided over the two scenarios. A complete list of these questions can be found in appendix A.

   Simulation 1: The first scenario contains one plant population with the four basic ecological processes affecting the population size: birth, death, immigration and emigration. There were 10 questions (nr. 1-10) about the first scenario. An example question (with instructions for operating VisiGarp) is shown below:

   Session with VisiGarp, simulation 1, question 1.

   In the *State-transition graph*, select state **3**. From the **View** menu, select the option *Entities and relations*.

   Which of the following statements is true?

   a. open-population consists-of population1.

   b. population1 consists-of biological-species1.

   c. open-population is an assumption about biological-species1.

   Simulation 2: The second scenario included the Cerrado vegetation, with three populations (trees, shrubs and grass) and several environmental factors. There were 17 questions (nr. 11-27) about this scenario. Two examples are given below:

   Session with VisiGarp, simulation 2, question 24.

   What can you derive about the growth of the **shrub** population?

   a. The shrub population increases, because immigrations equals emigrations, while births are higher than deaths.

   b. The shrub population increases, because immigrations are higher than emigrations, while births are equal to deaths.

   c. The shrub population increases, because immigrations are higher than emigrations, AND births are higher than deaths.

Session with VisiGarp, simulation 2, question 25.

The factors **soil-temperature, light, moisture**, and **nutrient** have an effect on the **births** and **deaths** of populations in the Cerrado vegetation. Which of the following statements is true regarding the **tree** population?

a. when litter increases, the composition of soil temperature, light, moisture, and nutrient changes in such a way that for the tree population, the number of births (growing of new trees) become equal to the number of deaths (dying of existing trees).

b. when litter increases, the composition of soil temperature, light, moisture, and nutrient changes in such a way that for the tree population, the number of births (growing of new trees) becomes greater than the number of deaths (dying of existing trees).

c. when litter increases, the composition of soil temperature, light, moisture, and nutrient changes in such a way that for the tree population, the number of births (growing of new trees) becomes smaller than the number of deaths (dying of existing trees).

5. Paper-based post-test of domain-specific knowledge about the Cerrado. Those participants who received test CSH-A as pre-test, received test CSH-B as post-test, and vice versa.

6. Paper-based post-test of diagrammatic representations as shown in VisiGarp. Those participants who received test DIA-A as pre-test, received test DIA-B as post-test, and vice versa.

7. Paper-based questionnaire about the subject's attitudes towards VisiGarp. The attitude questionnaire consisted of fourteen questions with an answer range from 1 (negative) to 5 (positive). The complete list of questions is shown in the results section, from page 52 onwards.

8. Short time for additional comments and questions. In this time, participants could ask or comment about VisiGarp, the experiment or the research in general.

Except where noted, all texts in the example questions, answers and VisiGarp figures have been translated from Dutch to English. The original Dutch version is available in Tjaris [2002].

2.5.3.4 Results

All participants completed the experimental procedure, except for some of the treatment questions. In the following subsections, the results are shown and briefly discussed for each type of data. In the statistical tests, we assume the data come from a normal distribution; the data did not give rise to discard this assumption.

Domain knowledge pre-test and post-test

To check whether there was any difference in difficulty level between test CSH-A and CSH-B, the results of both tests were compared. As no difference in results could be found between the results on CSH-A and CSH-B as the pre-test, nor between the results on CSH-A and CSH-B as the post-test, the results of test CSH-A and test CSH-B are lumped together. This way, the pre- and post-test scores can be compared without concern about the fact that half of the participants received test CSH-A before CSH-B and the other half vice versa. Table 2.3 shows the results for the individual participants' mean scores on the CSH domain post-test compared to the pre-test, as the percentage of questions answered correctly.

Participant	Pre-test	Post-test	Difference
1	50	60	10
2	40	80	40
3	30	60	30
4	40	70	30
5	40	70	30
6	40	70	30
7	40	70	30
8	40	80	40
9	40	70	30
10	50	50	0
11	30	50	20
12	60	50	-10
13	40	80	40
14	50	80	30
15	50	90	40
16	40	50	10
17	30	70	40
18	20	80	60
19	40	80	40
20	70	40	-30
21	20	60	40
22	30	60	30
23	20	80	60
24	40	40	0
25	50	70	20
26	50	80	30
27	60	70	10
28	60	80	20
29	40	80	40
30	50	40	-10
M	42	67	25

Table 2.3
Domain knowledge pre-test and post-test scores per participant (% answers correct).

Twenty-six out of the thirty participants improved their scores. The three participants whose scores decreased (number 12, 20 and 30) had rather high scores on the pre-test, which may have been the result of lucky guessing. Averaged over all subjects, the mean scores for the pre- and post-test are depicted in figure 2.27, together with 95% confidence intervals, indicating the error margins. As can be seen, the participant's scores increased significantly from the pre-test (M=42%, SD=12.1%) to the corresponding post-test (M=67%, SD=14.2%, t(29) = -6.75, $p < 0.001$).

In figure 2.28, the mean scores per question are shown for both the domain knowledge pre-test and post-test. Question number 7 (*What is the effect of an increase in sunlight on the amount of*

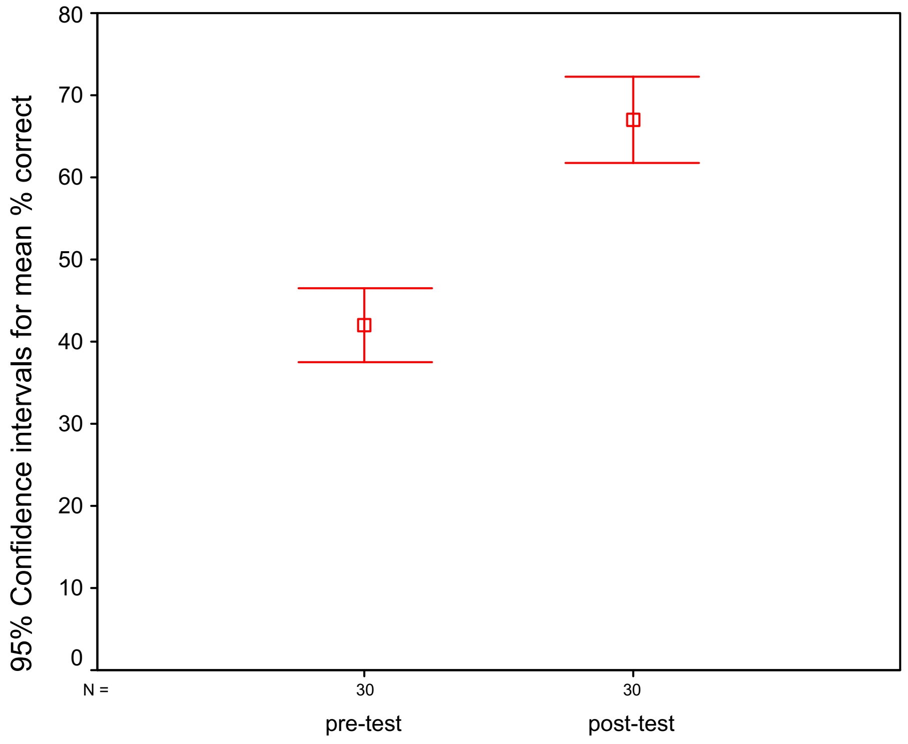


Figure 2.27
Mean scores and confidence intervals for the domain knowledge pre- and post-test.

births for the shrub(A)/tree(B) population? Correct answer for A and B: decrease) shows remarkably low scores: 0% and 7% correct for the pre- and post-test, respectively. A reason for the low pre-test score may be that the participants were not aware that less seeds of trees and shrubs germinate when the amount of sunlight on the ground increases (in contrast to grass seeds), which causes the birth rate of trees and shrubs to drop. It is not surprising that this score did not increase very much, because the effects of sunlight on the different plant populations were not directly addressed in any of the treatment questions. The scores for question 8 are also interesting because they show a small decline. Question number 8 was, in both version A and B the same (*What is the effect of an increase in moisture of the soil on the amount of births for the grass population? Correct answer: increase*), but the answer was a tricky one. The correct answer could be found using VisiGarp to show that there is a P+ dependency from moisture1 to born3 of the grass_population, but because this dependency is submissive to other effects, born3 actually *decreases* in most states of the simulation. This may have caused some confusion, as the status of effects was not visible in VisiGarp at that time (It is in WiziGarp, discussed in Chapter 4). Except for question 8, the mean post-test results are higher than the pre-test for all questions. This indicates that the increase from pre- to post-test scores is a general effect, and not localized to a few questions.

Diagram pre- and post-test

Also for the diagrams tests, the results of version DIA-A and DIA-B were compared, but no significant difference was found. With the results of DIA-A and DIA-B lumped together, the mean scores for the

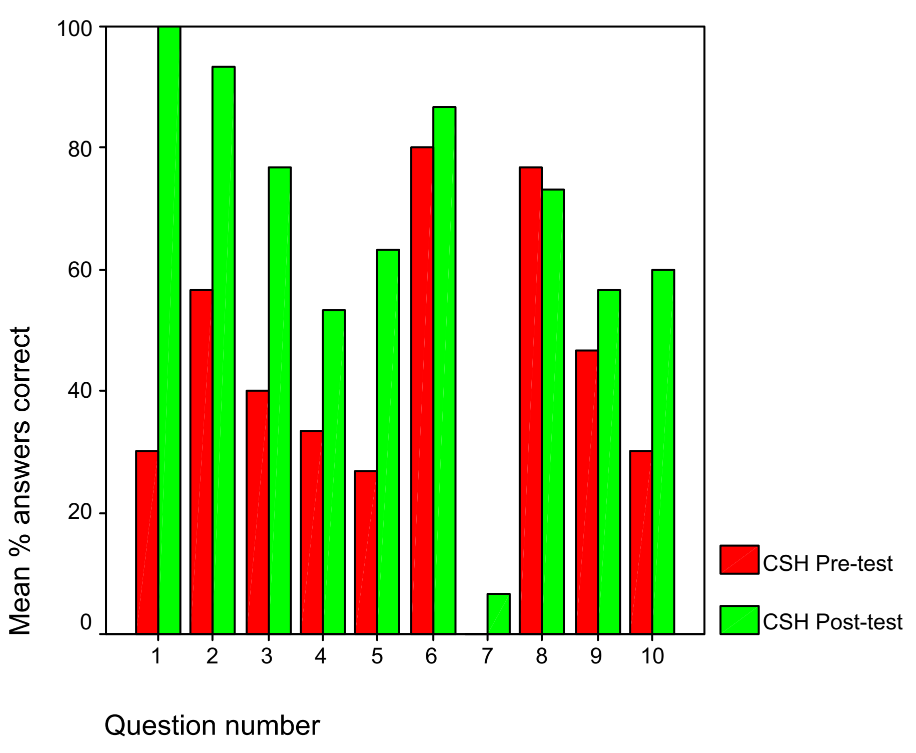
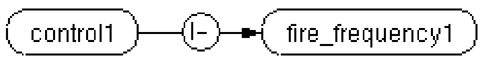


Figure 2.28
Mean scores per question on the domain knowledge pre- and post-test.

diagram pre- and post-test are shown in figure 2.29.
The scores on the pre-test are already reasonably high (M=67%, SD=17.6%). There seems to be a slight increase from pre-test to post-test (M=73%, SD=19.5%), but this difference is not significant. Investigating the scores on individual questions, raises some interesting points, however. Questions about positive relationships visualized from left to right are relatively easy to answer. Questions which involve quantity space correspondences (Q-relationships), negative relationships, or relationships which are depicted from right to left (in contrast to the reading direction) are scored incorrectly more often. All three issues seem to improve from the pre- to the post-test, but some questions remain difficult, even on the post-test. Two of the most difficult questions are shown here.

Diagrams test DIA-B, question number 2.



What is the relationship between control1 and fire-frequency1?

a. if the value of control1 is negative, then fire-frequency1 increases

b. if the value of control1 is negative, then fire-frequency1 decreases

c. if the value of fire-frequency1 is negative, then control1 increases

Mean score on pre-test = 43.5%, post-test = 60% correct. The low results here may be due to the presence (in answer b) of a double negative, which is known to be hard to

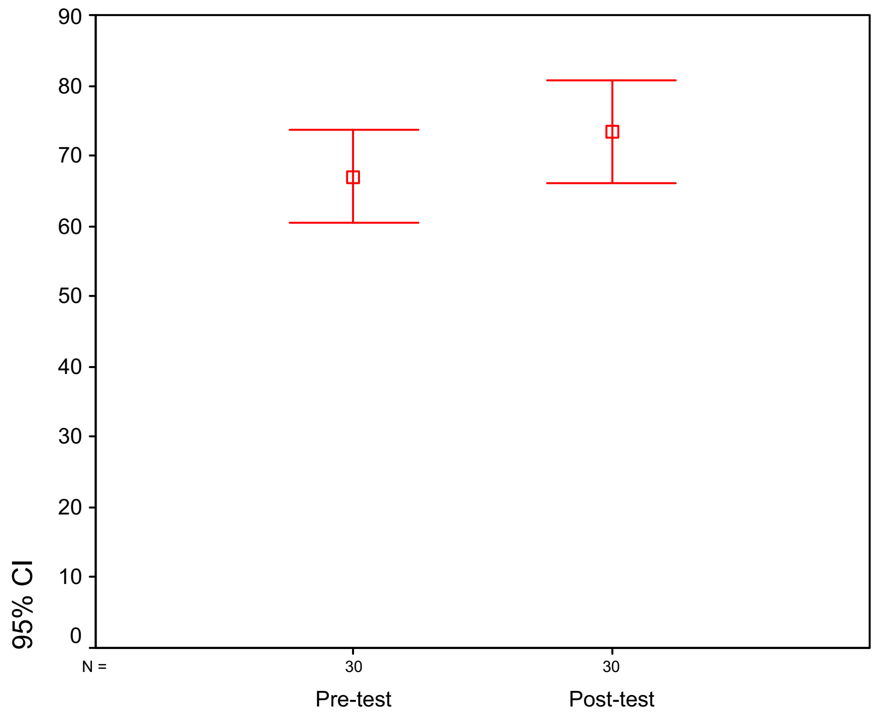
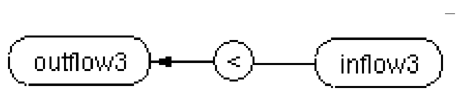


Figure 2.29
Mean scores and confidence intervals for the diagram pre- and post-test.

understand in general. The following question resulted in even lower scores, however: Diagrams test DIA-B, question number 6.



What is the relationship between inflow3 and outflow3?

a. outflow3 is smaller than inflow3

b. inflow3 is smaller than outflow3

c. inflow3 is greater than outflow3

Mean score on pre-test = 13%, post-test = 23% correct.

Here, the visualization clearly introduces a problem in readability. The problem stems from the similarity between the picture and the normal textual representation of the reverse relation, 'outflow < inflow'. The picture is the result of the visualization principle (recall from section 2.2.2) of not rotating text (including relationship symbols such as '<' and '>'), while the quantity nodes are free to be moved around. This does not guarantee that the arrow follows the normal reading direction, and occasionally results in situations as in question number 6. Training may help to follow the rules to read the picture correctly: the symbol '<' means 'smaller than', and the relationship holds between inflow3 and outflow3, in that direction as enforced by the arrow, so inflow3 is smaller than outflow3. To what extent this kind of training will solve the problem requires further experimentation.

Treatment exercises

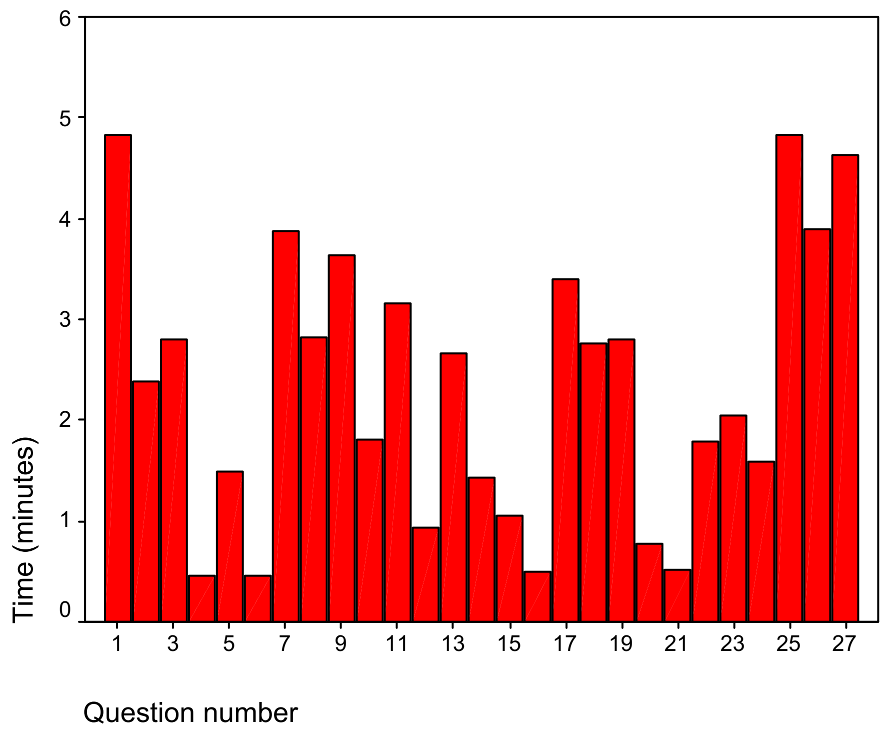


Figure 2.30
Times per question in the treatment with VisiGarp.

As mentioned before, the questions presented during the treatment session with VisiGarp can be found in appendix A. The first 10 questions were about scenario 1 (one population, four processes), the latter 17 dealt with scenario 2 (three populations, a manager and processes for each). Not all participants finished answering all 27 treatment questions, but everybody got at least to question 22. Only the answers given have been considered in the analysis.

The percentage of correct answers was generally high (M=94.3%, SD=8.6%) and did not differ between scenario 1 and 2. That the percentage of correct answers is so high is not surprising given that all answers could be found in the diagrams generated by VisiGarp, if the instructions were followed correctly. Relatively difficult questions were number 3 (the first question about a quantity space, M=67% correct), and question number 26 (a complex question about causal and mathematical dependencies involving six interrelated quantities, M=70%).

The average time spent on each question is shown in figure 2.30. The questions 4, 6, 12, 16, 20, and 21, were all finished within one minute, on average. The questions 1, 25, and 27 took most time - between four and five minutes, on average. It is important to note that this time includes reading the question, instructions and answers, opening the right views, manipulating the layout (if necessary), and choosing an answer. Some diagrams generated by VisiGarp for simulation 2 were quite complex and cluttered, because VisiGarp 1.0 fits everything into view, even if that means that things will overlap. Most participants made use of the zoom buttons, 12 times per person on average over the course of the whole experiment. This made it easier to focus on a certain part of the system, *e.g.*, the tree population, as the topic of the question at hand. Still, a lot of time was spent on adjusting the layout by dragging quantities and entities over the screen. Some students were quite good in improving the layout this way, but others occasionally made it worse, by selecting and moving the wrong element because it partly overlapped with another. Two illustrative examples, from different students working on the same task (exercise 25) are the screenshots displayed in figure 2.31 and 2.32.

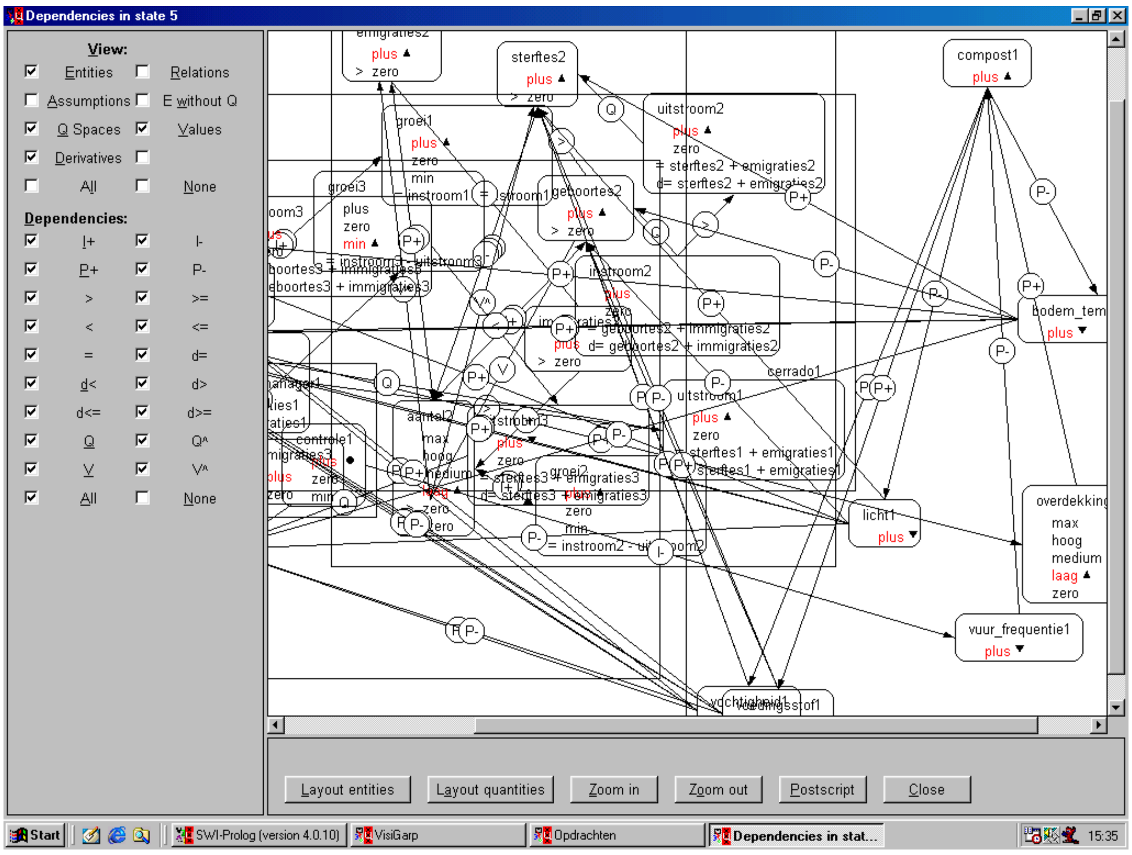
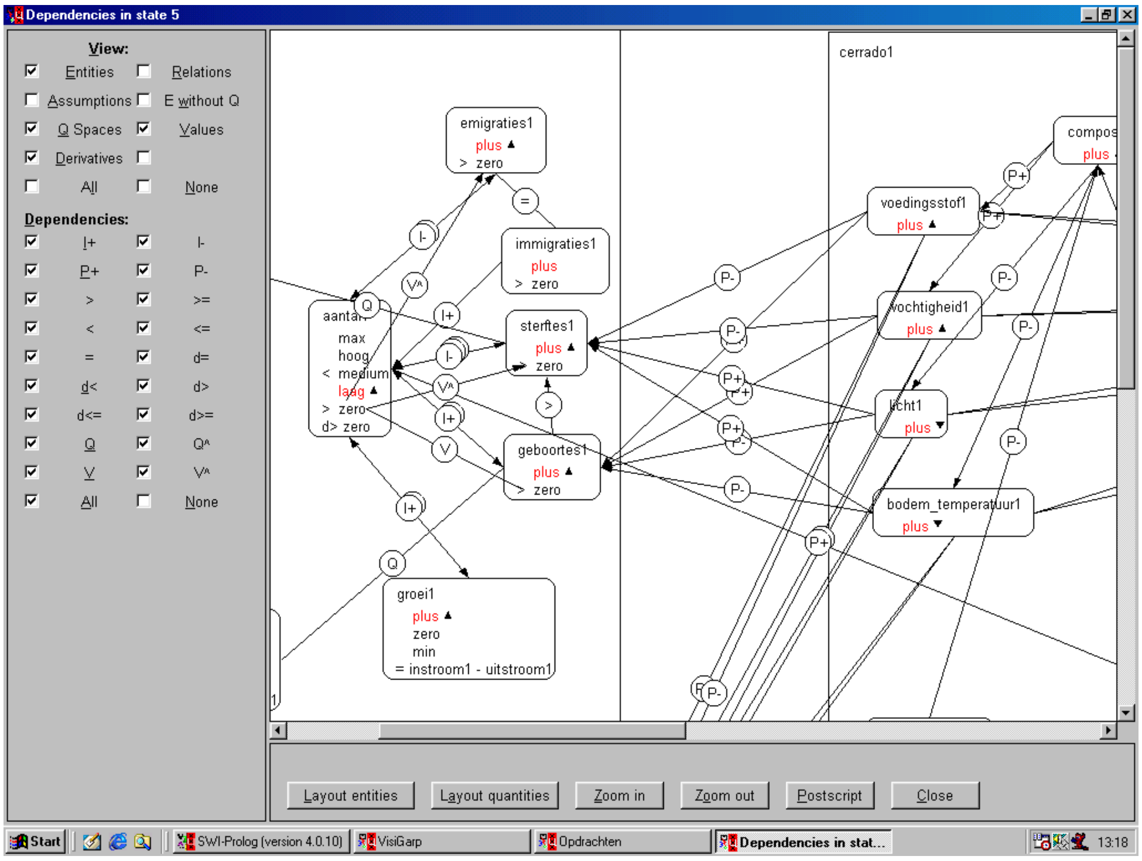


Figure 2.31
A screenshot from a student while working on a task. This is an example of a suboptimal diagram layout of VisiGarp ameliorated further by the student.

In total, the amount of time spent on all questions varied between 46 minutes to 77 minutes between the fastest and slowest participant (M=62 min.).

The participants rated the questions in scenario 2 (M=3.07) as more difficult than scenario 1 (M=2.53) (t(29) = -2.64, p=0.013). This is not suprising, given that the simulation model in scenario 2 contains much more information than in scenario 1. From their comments however, it seems that participants did not find the content so hard to understand, but rather that they found it hard to find the right answer on the screen.


Dependencies in state 5

View:
✓ Entities ☐ Relations
☐ Assumptions ☐ E without Q
✓ Q Spaces ✓ Values
✓ Derivatives ☐
☐ All ☐ None

Dependencies:
✓ I+ ✓ I-
✓ P+ ✓ P-
✓ > ✓ >=
✓ < ✓ <=
✓ = ✓ d=
✓ d< ✓ d>
✓ d<= ✓ d>=
✓ Q ✓ Q^
✓ V ✓ V^
✓ All ☐ None

Figure 2.32
A screenshot from another student working on the same task. Here, the student has adjusted the layout to be much better.

Attitude questionnaire

Table 2.4 shows the participants' mean attitude ratings, averaged over all questions, on a scale from 1 (negative) to 5 (positive). The participants rated VisiGarp quite postively overall (M=3.70, SD=1.14), with only two participants (number 1 & 27) scoring an average of less than 3 (neutral). The hypothesis that these participants experienced more trouble in answering the questions correctly could not be confirmed. Figure 2.33 shows the ratings per question, averaged over all subjects. For ease of reference, all questions of the attitude questionnaire are listed below, translated into English from the original version in Dutch [Tjaris, 2002].

1. VisiGarp is a system which allows me to check predictions about possible changes in the Cerrado vegetation.

2. VisiGarp is a tool which helps me to explain possible changes in the Cerrado vegetation.

3. Concepts in the Cerrado domain, such as *populations, trees, mortality, and emigration*, connect to the ideas I have about an ecosystem like the Cerrado.

4. The primitives used in the VisiGarp tool, such as *states* and the different *causal relations (I+, P+, etc.)* are comprehensible.

Participant	Mean Rating
1	2.79
2	3.00
3	4.50
4	3.79
5	3.57
6	3.29
7	4.00
8	4.00
9	4.29
10	3.43
11	3.79
12	3.86
13	4.29
14	3.36
15	3.79
16	3.57
17	3.71
18	3.93
19	3.93
20	3.86
21	3.36
22	3.07
23	3.71
24	3.71
25	4.29
26	3.71
27	2.50
28	4.00
29	3.14
30	4.64
M	3.70 (SD = 1.14)

Table 2.4
Mean ratings per participant.

5. Because of using VisiGarp, I better understand how an ecosystem, such as the Cerrado, works.
6. The information necessary for carrying out the exercises in the experiment was easy to read from the diagrams.
7. Overall, the exercises in the experiment were easy to carry out.
8. Carrying out the exercises in the experiment have contributed to acquiring knowledge about the Cerrado domain.
9. From the different views available, I was able to find the right view very quickly.
10. The layout of the diagrams as generated by VisiGarp was well-organized.

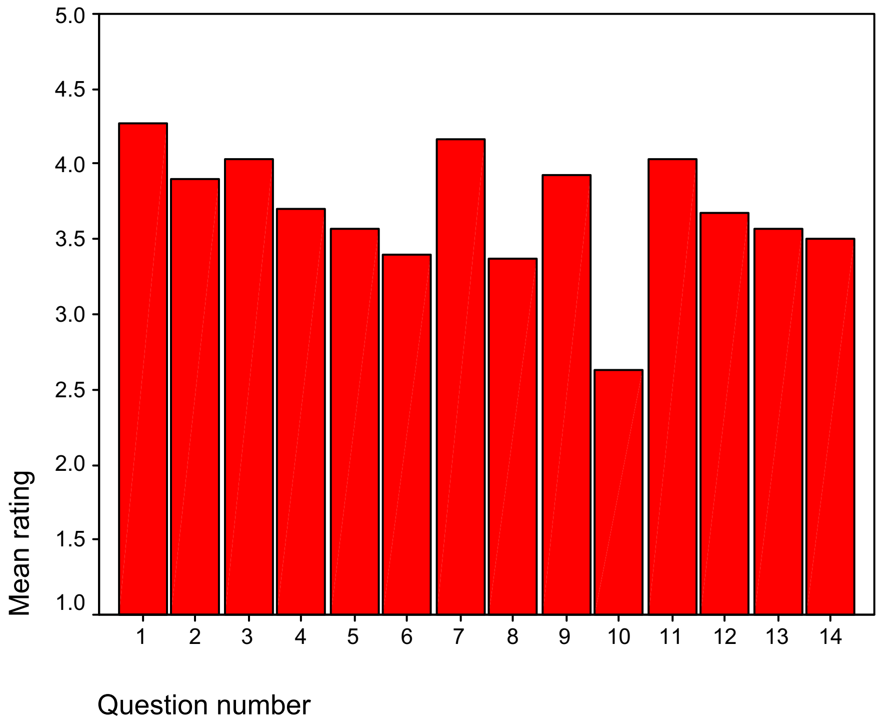


Figure 2.33
Mean ratings per question.

11. The layout of the diagrams was easy to adjust.

12. The instructions on how to use VisiGarp (which preceded the exercises) have contributed to how quickly and easy I was able to find the answers to the exercises.

13. When the information did not fit onto one screen, the scroll-bar was a convenient way to see the total workspace of the different views.

14. The help document has been useful for understanding VisiGarp and finding the information desired.

The subjects considered VisiGarp a good tool to inspect changes in the Cerrado vegetation simulation. They were slightly negative about whether the layout of the diagrams generated by VisiGarp was clear (Question number 10: M = 2.63, SD=1.30), but they felt the layout was easy to adjust (Question number 11: M=4.03, SD=1.00).

2.5.3.5 Discussion

This study confirmed that university students without any experience in qualitative reasoning can learn to make use of VisiGarp to complete relatively complex exercises within an hour and a half. Moreover, the students rated VisiGarp quite positively afterwards. With the improved layout mechanisms and zoom functionality of VisiGarp 1.0, the layout of the diagrams of the small model in the first simulation scenario was fine, but the complex diagrams of the second simulation still posed a problem for many students. The diagrams showed all the information cluttered due to the automatic zooming to make it fit on the screen. The new zoom buttons allowed users to focus on particular parts of the system, and the fact that most participants used them indicates that this functionality fulfils a need. Still, it was often necessary to adjust the layout manually. From comments by the participants and the experimenter's impressions during the interaction, it is felt that the students still spent too much time on manipulating the visualizations, rather than on contemplating the content of the visualizations. The results from the domain pre- and post-tests indicate that a strong learning effect has taken place during the course of the experiment. This is remarkable, given that other experiments with simulation-based learning do not always show a clear learning effect (*e.g.*, see van Berkum and de Jong [1991] and van Joolingen [1999] for a discussion of several projects involving simulation-based learning software). Caution is necessary in interpreting these results however, because no control group was used in this experiment. It could be that the subjects learned from interacting with VisiGarp and looking at the diagrams it generates, but it is also possible that the exercise questions and instructions were the determining factor. A third possibility is that just presenting them with the pre-test sparked more thinking about the material, which led to a higher performance on the post-test. Also, it is unclear whether the learning experience would have been as efficient (or less, or even more) when all material was presented on paper instead of using VisiGarp on a computer screen. Therefore, a more comparative study is necessary to establish how much the use of VisiGarp is responsible for the learning outcome. However, as an educational tool which allows for controlling and inspecting qualitative simulation models of varying complexity, it is hard to think of something that is truly comparable, as indicated by the comments of teachers who participated in evaluating VisiGarp.

2.5.4 VisiGarp in use at Universities

Besides the evaluation experiments described in section 2.5.2–2.5.3, VisiGarp has also been used in other environments, mainly at the following two sites: the department of Social Science Informatics at the University of Amsterdam and the Laboratory of Research in Biology and Science Teaching at the University of Brasilia.

University of Amsterdam. In Amsterdam, VisiGarp is currently in use by several MSc students at the departments of Social Science Informatics and Artificial Intelligence. These students build models in various domains (the hydrological cycle [Koeman, 2003], the process of osmosis [van der Werf, 2003], and the greenhouse effect [Dubbeldam, 2003]), and use VisiGarp to inspect simulation results and to create diagrams for presenting their models. They have given additional comments about possible improvements, but also positive feedback about the added value of visualizations above the text-based interface of GARP.

University of Brasilia. In Brasilia, the ecology expert involved in developing the model for the Brazilian Cerrado vegetation uses VisiGarp in his teaching in several courses about ecological modelling. In undergraduate courses for future teachers of biology, the focus is on educational aspects: causal relations, and how questions can be answered using VisiGarp. In postgraduate courses in ecology, the focus is more on how the knowledge is represented, including details about model fragments, and relationships with mathematical modelling. The models used in the courses are the Cerrado simulation model, and the model describing interactions between two populations [Salles et al., 2003].

In an interview about his experiences, the expert made a number of comments about VisiGarp, which are summarized below:

About the way that VisiGarp is typically used by postgraduate students, the expert said: ‘I give them the software, or they download the software and then they are free to use it’. ‘They are introduced to the software and the modelling approach, and we do some examples–let’s say we work on that for eight hours, and then they can explore on their own’. ‘The students select a simulation scenario, run the full simulation, select all the states, all the quantities, and try to see what is in there’. Then, ‘they notice there is a big bunch of states’ and ‘realize that something has to be done differently’. They start using the path functionality, by selecting select one path, or a group of states, to see how they relate. ‘Even if I [the expert] don’t say anything, this is more or less what they do’.

Reporting on the usefulness VisiGarp and its diagrams, the expert commented the following: ‘The whole thing is very complex right, so, you really need some time to get used to the primitives. OK, fine. I think you’ve done a great job at this point because it’s really important to see graphically how these things are related and also having these options that you can just show some. For instance, if you want to show just I’s and P’s, and remove the rest–these are the most important ones. Or if you want to show which (in)equalities are active, and leaving these causal things out’.

About the representation of values changing, he commented: ‘I think it’s a very nice representation, a very good representation’. ‘Before [there was VisiGarp], it was just a name, the labels, using just the labels for the quantities. Let’s say number-of1 equals normal and then number-of1 equals high. Only that. ... This was my way of representing that. But this is much better’.

About the state-transition graph, he commented: ‘You can see more or less a shape connecting different states, right, which is a good thing to be explored’, and ‘You need the order representation of how things change in order to understand what is going on’.

About the amount of information on the screen: ‘well, you can not avoid it and I think you have done a good job - that’s a positive point, reducing the information by just showing some of them and making others disappear.’ But also, on a more critical note: ‘There is lots of information in each screen. I could feel this preparing an assignment for people for a workshop. It is difficult to say OK, if you are here, for instance, you go to this area, select and then to that area, view, and select one of the things. And then, if you go for value history then again you have three different areas, three more buttons, two windows, you see. It’s lots of information for people to take in. And I know that the thing is complex, and probably, it would be worse to have lots of windows open instead of having just one with more information, but I think it would be good to have maybe some difference here, here, and there or to use the same type of representation, or maybe a help file later on, things that are assistive, when you hold the mouse at a certain point, like in windows you have, then pop the name of the thing you should do. Things like that- some sort of support for the user.’

About the representation of quantities and dependencies, when asked whether VisiGarp diagrams are like what he would draw himself: ‘I do more or less the same. What I don’t find very useful is that there are no differences between quantities that are influenced and the rate ... both are represented the same way ... For instance, consider number-of (a state variable) and birth rate (a process rate). It would be useful to have different representations (a different shape) for the two quantities, because this one is a direct influenced variable and this is a process, a rate, actually. It is a bit misleading that we

> didn't use the word rate in the model itself. Maybe we should have done so'. 'If you would like to be even more precise ... then you have three types of quantities: this one is the rate, this one is the state variable, and this one is the intermediate variable. But I have to say that this is very much system dynamics oriented. You find these things in STELLA, SIMILE, software like that'. 'They are based on Forrester diagrams'.
>
> About the use of model fragments in VisiGarp, the expert said the following: 'They use the options for watching model fragments, but they do this mainly because I tell them to. They don't understand exactly what is in there, but given that we're using this tool in the context of building, or at least designing models, then they start paying attention to the model fragment facility'.
>
> When asked which model fragments are most important, the expert replies: 'First, the processes. I always start with the *processes*, because this is how things change - the basis for the whole change in the system. They have to understand also the *situations*, that is, the description views'. At the time of building the model, we were exploring the idea that every concept should be represented as a model fragment. 'Then people, when reading the list of model fragments would have an idea of the main concepts that are there. But there is a problem of having such a big library'. 'We had some model fragments that would not be necessary, for instance the types of model fragments that just describe the size of a population'.
>
> About the model fragment views (graphics vs. text), he commented: 'We prefer the text version–it is more compact'.

Interestingly, this teacher is far more positive about VisiGarp than the physics teacher involved in the evaluation discussed in section 2.5.2. Perhaps this is partly because as a model builder and expert in the simulation domain, the ecology teacher is better able to establish the educational context for the tool himself than the physics teacher. Another likely explanation is that it takes more time to get used to VisiGarp and to realize its possibilities than the two hours scheduled for the evaluation experiments.

The willingness of the students and teachers at the two universities in Amsterdam and Brasilia to use (and keep using) VisiGarp indicates that it has reached an acceptable level of usability, and that it fills a gap in the spectrum of tools which make qualitative simulation techniques available to students at different levels of expertise in different domains. This is further corroborated by the fact that recently, people from other places have also begun to use VisiGarp for research and educational purposes [Alvarez-Bravo et al., 2004, Neumann and Bredeweg, 2004, Nuttle et al., 2004, Salles et al., 2004, Tullos et al., 2004].

2.6 Related Work on Visualization of Qualitative Models

Around the same time that VisiGarp was being developed, related projects have also resulted in graphical tools which display qualitative models for educational purposes. Most of these are tools for building models, rather than visualization tools for existing simulation models, however [Soloway et al., 1997, Reichherzer et al., 1998, Jellema, 2000, Leelawong et al., 2001, Forbus et al., 2001, Bessa Machado and Bredeweg, 2002]. Model-It [Soloway et al., 1997] incorporates Forrester diagrams, often used in system-dynamics. Giant [Reichherzer et al., 1998] and Betty's Brain [Leelawong et al., 2001] are based around concept maps, with a very basic ontology. The main idea here is that students teach an artificial agent (or *teachable agent*), which can reason about the concept map and ask questions to stimulate the student to expand it. Evaluation studies with Betty's Brain suggest that constructing concept maps during studying is better than reading and writing summaries [Leelawong et al., 2001]. Betty's Brain and VModel [Forbus et al., 2001] include limited capabilities for simulation, including influence resolution to determine values and derivatives.

HOMER [Jellema, 2000] and MOBUM [Bessa Machado and Bredeweg, 2002, 2003, Bessa Machado, 2004] are model-building tools based on the GARP framework (like the work in this thesis) which allow detailed qualitative models to be built which can directly be used to run multi-state simulations. The visualization in HOMER is quite different from that in VisiGarp, with different icons for different types of model primitives (entity, quantity, quantity space, derivative), and lines in between, to indicate which belongs to which (instead of the container metaphor, which is mainly used to represent the belongs-to relationship in VisiGarp). These differences are at least in part due to the fact that the modelling task poses different requirements on the visualization. For instance, when a model-builder wants to create a quantity, then having an icon for that in the interface is a logical affordance for doing that. Additionally, MOBUM includes a sketch-pad, in which students can draw a schematic figure of the system (similar to figure 2.6) they want to model, with the possibility to make reference annotations [Groen, 2003]. These annotations can be used as the start for modelling the system in the formal GARP representation.

Next to MOBUM, VModel [Forbus et al., 2001] probably comes closest to VisiGarp, in terms of the visualization, although this tool is aimed at middle-school students, a younger target audience than the high-school and university students that VisiGarp aims to address. This is reflected in some of the design decisions, such as a heavier use of colours, a smaller set of model primitives, and most importantly, the decision to limit the complexity of simulation to one state only. An interesting difference in the visualization is that processes are represented in the same way as entities (black labeled boxes), with the addition of a rate parameter, which is displayed within the process box. Other than that, the container metaphor is not used as extensively in VModel as in VisiGarp.

The EXPOUND system [Mallory, 1998] is based on the QSIM framework [Kuipers, 1994], and is, like VisiGarp, designed to explain *multi-state* simulations. EXPOUND generates *abstracted* state-transition graphs and influence diagrams (showing quantities and causal dependencies), which are positively rated by users (QR experts). However, EXPOUND generates these diagrams as non-interactive output; they are not integrated in the interface as in VisiGarp. A useful feature of EXPOUND which is currently not included in VisiGarp is the possibility to focus on particular system parts to create simpler diagrams. This issue will reappear in Chapters 3 and 4.

2.7 Conclusion

Qualitative simulation is a powerful tool for predicting and explaining system behaviour in various domains. The explicit representation of all kinds of knowledge involved makes it a fruitful resource for education. Communicating the information contained in qualitative simulations is not a trivial task, however. In this chapter, *visualization* has been investigated as the means of communication for this purpose. A tool, VisiGarp, has been developed, which generates diagrams of any simulation model expressed in the format of GARP, a framework for qualitative reasoning. The interface of VisiGarp is organized around a number of views, each of which focuses on particular kinds of information. There are views which show generic information in a domain library, such as the entity is-a hierarchy, the transition rules, and several views of the model fragments, chunks of knowledge about situations or processes. Furthermore, there are views which show the results of a specific simulation, such as the state-transition graph, the quantity value history view, and the dependencies view. The state-transition graph shows which qualitative states the system may manifest, in which possible order. The quantity value history view shows how the value and derivative of quantities in the simulation change over time. The dependencies view shows the details of the system in a particular state, including the entities, attributes and structural relations involved, but also the causal and mathematical dependencies which hold between quantities.

VisiGarp contributes to the state of the art in the form of the visualization principles behind the design of the visualizations, including the mapping from model primitives to a set of visual primitives applicable to qualitative simulation. Furthermore, it is one of the few systems, next to the EXPOUND system [Mallory, 1998], capable of generating explanatory diagrams for qualitative multi-state simulations. Compared to EXPOUND, VisiGarp includes a larger set of views, showing a richer set of model primitives. Furthermore, VisiGarp's diagrams are an integrated part of the interface, which supports the navigation process.

VisiGarp was designed to support interactive investigation of qualitative simulation models and results. To check whether it fulfills this goal, VisiGarp has been used and evaluated by both students and teachers in several classroom and experimental settings. The results of these studies indicate that students without knowledge of qualitative reasoning quickly learn to use VisiGarp, and rate it positively. Furthermore, while using VisiGarp, the students are able to run qualitative simulations and answer questions about system behaviour in a domain unfamiliar to them, and they actually learn something about the domain in the process. After using VisiGarp for a longer period, students who are more knowledgeable about qualitative modelling in a certain domain appreciate the graphical interface above the text-based interface of GARP, the underlying simulation engine.

There is room for improvement, however, as both teachers and students suggest. Visualization of simulation models and results may not suffice to explain them. One teacher argued that more attention should be paid to the embedding in an educational context, including illustrative examples and case-studies which indicate the added value of simulation and explanation tools. This would require a less generic, and more domain-dependent approach, however.

When considering the interface of VisiGarp, comments from users indicate that improving the quality of the diagram layout seems to be most worthwile. Three possible solutions come to mind: improving the layout algorithm, improving the zooming mechanism, or improve the instructions on how to use the interface. Although all these three options contain room for improvement, the improvement may be only small, and not very interesting from a research point of view. In the following chapter, a fourth option, with a far higher potential, is investigated in more detail: aggregation of the information, thereby reducing the number of graphical elements to show.

3
Aggregation of Qualitative Simulations

“Facts are stupid things.”

“The poverty rate has begun to decline, but it is still going up.”

Ronald Reagan (1911-2004), Former US President

3.1 Introduction

This chapter discusses ways to simplify and reorganize the information in qualitative simulations to facilitate communication of this information to users. Qualitative simulations can become very large and complex when the model covers a large part of a domain or includes a great amount of detail. In the context of the GARP framework [Bredeweg, 1992], the size of the model can be measured in terms of the number of information elements in the library or the scenario (*i.e.*, entities, relationships, quantities, values, dependencies, and model fragments). Existing GARP models vary in size, but the largest contain more than twenty entities, more than thirty quantities, more than fifty model fragments, and more than a hundred dependencies. The size of the resulting simulation can be measured as the number of states and transitions, while branching further increases the complexity, in terms of the number of behaviours. The largest GARP simulations contain more than a hundred states, and over three thousand behaviours. How to deal with such an amount of information?

The VisiGarp tool, described in chapter 2, offers visualizations of the information (both state-based and path-based), and ways of navigating it. This is necessary, but it does not suffice to make simulations easy to understand. The *state-based* visualizations (*i.e.*, the dependencies view, entities and relations view, and the model fragments views) provide a detailed view of what happens in a particular state, but when the number of states in a simulation grows large, investigating all states in turn becomes impractical. Sometimes, a state-transition only involves a single quantity changing, whereas other state-transitions induce more complex changes, involving changes of multiple quantities, as well as the introduction or removal of dependencies. Therefore, it would be useful if the system could support the user in determining when and where the interesting differences occur. The *path-based* visualizations (*i.e.*, the value history and transition history view) provide an overview of how the quantities can behave over time, but when there are multiple paths, all their corresponding visualizations have to be investigated and compared by hand to see what the differences are. Since a simulation often contains a number of paths in which the behaviour is rather similar, it would be useful if the system could make generalizations where possible. Furthermore, the path-based visualizations include mainly information about quantity values and (in)equalities, and not about other types of knowledge. How the set of the causal dependencies changes over time is not easily discernible. It would be useful if such developments were also made explicit.

In more general terms, the problem is that the amount of information makes it hard for a user to grasp the essence of what happens in a simulation. This leads to the research question addressed in this chapter: how can the system reduce and restructure the information from a simulation to support the user in interpreting it on both a detailed and more abstract level? The approach taken to answer this question consists of a conceptualization of aggregation levels and aggregation methods, and the implementation of a set of techniques that restructure and/or reduce the amount of information on each level of aggregation.

The structure of this chapter is as follows. Section 3.2 reviews the relevant literature on simplifying qualitative models and simulations. Section 3.3 and 3.4 describe the approach and the necessary theoretical concepts in terms of the aggregation levels and aggregation methods, respectively. In section 3.5, the aggregation techniques that have been implemented are specified in detail. In section 3.6, numerical results are presented. Section 3.7 points out the added value as well as the limitations of the approach in the context of related work, and section 3.8 concludes this chapter.

3.2 Review of the Literature on Aggregation

Adequately handling the complexity of simulations has been recognized as a problem by many in the field of qualitative reasoning [Frantz, 1996]. Some approaches to the problem try to solve the problem at the start, *i.e.*, how to limit the size of the model and simulation in the first place. Examples of this kind of work are compositional modeling [Falkenhainer and Forbus, 1991], automated modeling [Rickel and Porter, 1997], and automatic qualitative abstraction [Sachenbacher and Struss, 2001], which are all essentially methods to build the smallest possible model which can answer a given (set of) question(s). In this thesis, however, the approach is different. The simulation is treated as a given, and aggregation can only take place afterwards. Some work exists on model simplification, which also takes place after a simulation has been run, *e.g.*, [Weld, 1992, Bredeweg, 1992, Kamps, 1993]. The goal of model simplification is to examine which parts of the model can be left out without negative consequences for the accuracy of the simulation results. When states only differ with respect to their quantity values, and not in terms of the model fragments which apply, these states may be considered irrelevant distinctions ([Bredeweg, 1992], ch. 7). Bredeweg suggests that such states may be replaced by just one state, in which the specific quantity value statements are replaced by two inequality statements, which specify the range of values attained. When multiple quantities exist which are fully corresponding in all states of a simulation, in terms of both their value and derivative, all but one are considered redundant and can in principle be deleted [Bredeweg, 1992]. Kamps uses this idea for automatically determining irrelevant knowledge in GARP simulations, also including other types of knowledge, such as irrelevant values in a quantity space, irrelevant system elements (entities) and irrelevant model fragments [Kamps, 1993]. However, as he notes, there may be reasons for a knowledge engineer to keep some distinctions, such as cognitive plausibility, or a better relation to other existing models.

Still, model simplification is not the same as the work described in this thesis. The goal of aggregation in this work is not to create a better model, but to communicate the results of a simulation based on the model as it is. It is unknown in advance which questions the simulation should or can answer - this is in fact one of the things that needs to be determined by the aggregation process. The following work comes closer to this approach.

Aggregation for diagnosis. De Koning et al. have developed a method for aggregating qualitative models for model-based diagnosis [Davis, 1984] of learner behaviour, which is employed in an intelligent learning environment prototype, called STAR^{light} [de Koning, 1997, de Koning et al., 2000]. Their approach is as follows. Given a GARP simulation, first a base model is created which models the whole prediction process in terms of reasoning steps. These reasoning steps, or inferences, combine expressions (*e.g.*, statements about values, derivatives, or dependencies) about the domain to deduce new expressions. Each inference is represented as a component, with a name, inputs, support knowledge, and a specific output. Because the typical GARP simulation contains hundreds of expressions, this results in a similar large number of components. To allow model-based diagnosis, the number of components is reduced using the following three aggregation principles: (1) hiding non-essential details, which discards everything that does not belong to the main reasoning trace which leads to the actual behaviour of the system modelled; (2) chunking, which replaces subsequent components by a more abstract component, *i.e.*, in the case of a causal chain of an influence whose effect propagates to other quantities; (3) grouping, which groups components in a state specification, or state transition step. Each of these steps further reduces the number of components on the next level of aggregation, while retaining the links to the underlying, more detailed information. The resulting hierarchical aggregation of the model is used in the prototype intelligent learning environment STAR^{light}. The key idea is to ask questions about what remains at the highest level of aggregation first. As STAR^{light} handles the simulation by state, the first question is usually to predict which quantity will change during the next state transition, and in which direction it will change. If the learner answers this question correctly, STAR^{light} assumes that the learner performed the correct reasoning steps, and proceeds to the next question. If the answer is wrong, the General Diagnostic Engine (GDE, inspired by de Kleer and Williams [1987]) is activated, which tries to identify which of the underlying reasoning steps have not been performed correctly. If the diagnostic engine cannot yet determine this accurately, STAR^{light} will ask the learner to answer another (probe) question to refine its search. The GDE process ensures that a question is asked about the reasoning step in the base model which is most informative to arrive efficiently at a correct diagnosis.

This method is quite powerful in that it is capable of reducing the complexity of a simulation model to a level which can be used in a practical way in a diagnostic session, localizing particular reasoning steps missing in the learner' reasoning trace. However, it also suffers from a number of limitations. (1) The scope of reasoning steps in the base model only includes values, derivatives, and causal and mathematical dependencies, whereas more types of reasoning steps are actually required in the prediction process. For example, also the inferences about structure, and the application of model fragments could be taken into account. This becomes especially important when the goal becomes explanation rather than diagnosis, because it may provide a more coherent basis for explanation than the loose dependencies, which the diagnosis currently comprises. (2) The focus on the direct successor state is too narrow for explanation purposes. This makes it impossible to view developments that take place over a course of time, *i.e.*, a sequence of multiple states in a simulation. Looking further than only the next state provides more opportunities to apply the ideas of hiding, chunking and grouping, and makes other focusing strategies possible, such as histories. (3) It is impossible to specify a particular focus for the aggregation process. When making use of explanatory communication goals, there may be more opportunities to select particular information (e.g., a specific part of the system) and hide the rest.

Abstraction based on user interests. Mallory adopts the idea of using user interests for abstracting information from a qualitative simulation in his explanation generation system EXPOUND [Mallory, 1998], based on the QSIM framework (e.g., see Kuipers [1994]). Given *"a selection of variables of primary interest, and aspects of their behaviour to focus on (typically either the qmag* (i.e., value) *or the qdir* (i.e., derivative)*, the behaviour graph* (i.e., state-transition graph) *is simplified by grouping into one abstract state adjacent base (original) states that do not differ in these aspects. The user will typically build several different abstractions that highlight the behaviour of individual state variables or pairs of variables in the model"* ([Mallory and Porter, 2000], p. 1). These abstractions are faithful to the original simulation in the sense that no behaviours are lost (*i.e.*, the method is *complete*), and all remaining behaviours can be mapped to a path in the original graph (*i.e.*, the method is *sound*). Another important point of EXPOUND is that instead of states, *events*, such as maxima, minima, and landmark crossings, are considered as the foundation for explanation. *"Events occur at time points, but their descriptions and explanations encompass the surrounding time intervals"* [Mallory and Porter, 2000, p. 2]. Events provide more depth in explanations than individual states can provide, because they describe the interesting changes in the behaviour of variables. More in line with the work on model simplification, but with the goal of explanation in mind, Mallory also suggests some ways to reduce the number of variables in the model. As he puts it, one should *"retain variables which are important to explanation and seem likely to be the ones of interest to the user"* [Mallory, 1998, pp. 14–15]:

- State variables (whose time derivative is explicitly computed)
- Exogenous variables (with no influences from other variables in the model)
- Tail variables (which influence no other variables)
- Flow variables (which determine the derivatives of state variables, *i.e.*, creation or destruction of substance, or its transport)

Eliminated are two classes of variables:

- Derivatives of state variables in cases where the derivatives are not themselves the flow variables
- Intermediate variables that connect the flow and tail variables to state and/or exogenous variables via functional constraints

By reducing the complexity of the model in this way, explanations can be generated that focus on the remaining variables of interest. However, Mallory's work also has its limitations. (1) Only events related to values and causal relationships are considered, while the role of other elements, such as entities, attributes, relations, and model fragments, is neglected. This stems from the fact that the work is based on the QSIM qualitative simulation framework [Kuipers, 1994], which is rooted in a strong mathematical foundation, but does not include as many types of knowledge as the GARP framework [Bredeweg, 1992]. (2) The abstraction mechanism requires that the user specifies his or her interests, which may be hard, especially for inexperienced learners. The automatic selection of variables which are likely to be of interest (described above) is not applicable to the GARP framework, because in GARP, all quantities would be classified as state variables, since each quantity's derivative is computed whenever possible. Also, (3) no attempt is made to explain branches in the simulation.

Abstraction based on didactic considerations. Besides user-specified interests, and technical aspects, there may also be considerations of a more didactic nature. For example, it may be useful to hide specific information because it has not been introduced before, and is scheduled at a later moment in the curriculum. The information may be considered confusing, or distracting at the current moment. To give an example, in the work by Suthers and Weiner [1995] on the Belvedere learning environment, the authors found that students spent too much time discussing the type of (argumentation) relationships that were present in early versions of the Belvedere interface. For this reason, the developers decided to simplify Belvedere's interface to include less types of relationships, essentially retaining only positive and negative associations. In the context of GARP, an analogous simplification would consist of grouping I+ and P+ dependencies together, as well as I- and P- dependencies.

Although this could be useful for beginners, or model builders in a preliminary informal stage of modelling, the goal is ultimately for learners to understand the differences between the different types of dependencies. Therefore, a trade-off exists between (a) making the distinction between the different dependency types from the start, and (b) starting simple with the need to introduce the distinction later.

Conclusions from the literature review. Reflecting on the advantages and limitations of the existing work defines an outline of the approach taken in this chapter. Given a particular model and simulation, the goal is to reorganize the information in a simulation to make it easier to communicate. A combination of two approaches to aggregation appears promising: hierarchical aggregation of simulation models à la De Koning, based on events à la Mallory, although the set of event types should be extended to include all kinds of information contained in GARP simulations. The local events view of Mallory already extends the state-based view of De Koning, but should be extended further to include a more global view, which also includes branches in a simulation. Specification of user interests to focus the aggregation should be possible but it should not be obligatory in order to start the aggregation process. Didactic considerations should be taken into account where applicable, to ensure that the appropriate information is selected for inclusion in the interaction.

3.3 Levels of Aggregation

The approach to aggregation presented in this thesis is to organize the information in a qualitative simulation in ways which support the simplification, interpretation and selection of interesting information. In order to do this, *levels of aggregation* are introduced, which correspond to different views on system behaviour. Conceptually, one can view the behaviour of a system represented by a qualitative simulation in three ways: (a) the state view, (b) the behaviour view, which follows a particular trajectory through the simulation, and (c) the global view, at which all alternative behaviours are compared. It makes sense, however, to subdivide the behaviour view in three intermediate levels, dealing with local events, path segments (necessary sequences) or paths (possible sequences). This further division leads to five levels of aggregation on top of the basic GARP level, which is included as level 0. They are shown in table 3.1, together with their mapping to the corresponding notions in the state-transition graph of a simulation. Each level is described in more detail in the remainder of this section.

Level	Conceptual view	State-transition graph
0. GARP	Output from the simulator	Individual states
1. System state	Current state of the system	Individual or aggregated states
2. Local event	Events at time point/interval	Two or three successive states
3. Path segment	Necessary behaviour	State sequence without branching points
4. Path	Possible behaviour	Path from start to end state
5. Global	Alternative behaviours	Multiple paths

Table 3.1
Levels of aggregation.

0. GARP: The information in the original states, generated by the GARP simulation engine. Each state specifies the structural elements and relations which hold at a particular moment or time interval, the quantities along with their values and derivatives, the mathematical and causal

relationships between quantities, and the active model fragments (representing situations, or processes). The information at this level is the source for the information at the higher levels of aggregation.

1. System State: A particular state of the system. This can be mapped to a single GARP state, but besides the information already present at level 0, some information is added: each causal dependency is assigned a *status* indicating to which extent it has an effect, as described in section 3.5.1.1.

2. Local Event: Things that (can) happen at some time point/interval. Two (or, in some cases, three) successive states are compared, and meaningful differences are represented as *local events*. This level largely corresponds to Mallory's notion of *trajectory* [1998], although he focuses mostly on value and derivative events, while this work includes also other types of events, such as (in)equality events, structural events, and model fragment events. It is similar, but not the same as GARP transitions. Like GARP transitions, this level deals with the difference between adjacent states, but it does this in a more integrated way. After a transition to a new state has been generated, GARP potentially adds more knowledge to the new state, based on whether the conditions of certain model fragments apply. In other words, a transition specifies the basic changes from one state to the next, in terms of quantity values and (in)equality relationships, but it does not fully specify the successor state. Hence, not all the differences between adjacent states (like the situation description changing, or processes becoming active) are included in a transition. On the local event level, however, such differences are explicitly represented, whenever they can be classified as meaningful events.

3. Path Segment, or Uniquely Determined Behaviour: Necessary sequence of events. A path segment is defined as a sequence of states without (outgoing) branching points, *e.g.*, the state sequence from s_m (possibly via s_{m+1}, s_{m+2}, etc.) to s_n in figure 3.1. Because it does not contain outgoing branches, it is possible to say that the situation at s_m has led to the situation at s_n. The path segment level contains the same types of events as the path level, but it is worth distinguishing the two because of the conceptual difference between necessity and possibility.

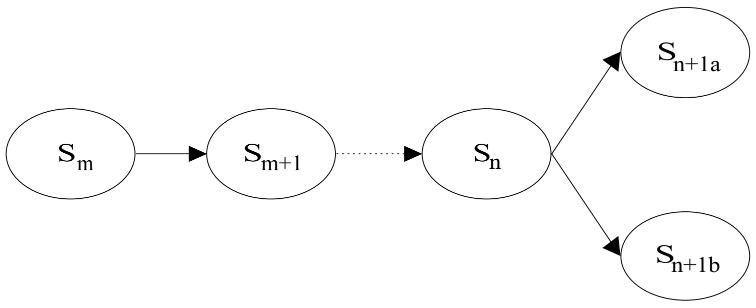


Figure 3.1
Path segments vs. paths. The path from S_m to S_n is a path segment, but the path from S_m to S_{n+1a} (or S_{n+1b}) is not, because it includes an outgoing branching point.

4. Path, or Possible Behaviour: Possible sequence of events. A path is defined as a sequence of states, *e.g.*, the state sequence from s_m to s_{n+1a} in figure 3.1. A path may include branching points, so this definition can be seen as a generalization of a path segment. Note that it is not valid to say that s_m has led to s_{n+1a}, because it could have led alternatively to s_{n+1b}. About a path, one can only say that the start situation *may* lead to the end situation, if there are alternative branches. Therefore, a path represents a possible behaviour of the system simulated. The presence of a cyclic path indicates that the system manifests repetitive behaviour.

5. Global View of Possible Behaviours: Alternative possibilities. This level includes all alternative paths, if there are any. If the simulation results in only one path, there is only one possible behaviour of the system modeled, given the input to the simulator. In many cases, however, the input or the model does not fully specify the behaviour in advance, and multiple possibilities will arise, creating branches in the state-transition graph. At the global simulation level, we can look at the differences between such alternative paths. In some cases, alternative paths only differ with respect to the order in which events occur, but in other cases, alternative paths may contain very different events altogether.

3.4 Aggregation methods

In order to present information clearly and efficiently, it is often necessary to limit the number of information elements to be communicated. This can be done by leaving out, or combining elements in various ways. In the approach presented here, four aggregation methods are distinguished for this purpose: *selection*, *chunking*, *generalization* and *grouping*. They are characterized in general terms in the text below, and depicted in the figures 3.2–3.5. These generic methods will be used later to define specific aggregation techniques.

In the figures, the boxes below the line denote arbitrary information elements (*e.g.*, value statements, dependencies, or events in a qualitative simulation), while the boxes above the line represent the resulting information elements after the application of one of the aggregation methods. The labels of the boxes are only meant to illustrate these general principles; no specific meaning should be attached to them.

1. Selection. Selection is a method which takes out certain elements of a larger set, based on a particular categorization (the selection criteria). For example, in figure 3.2, the elements C and F are selected from the set displayed below. The selection may be shown on its own, or shown in a way that sets it apart from the rest, e.g., in focus, while the rest of the set is still visible in the background.

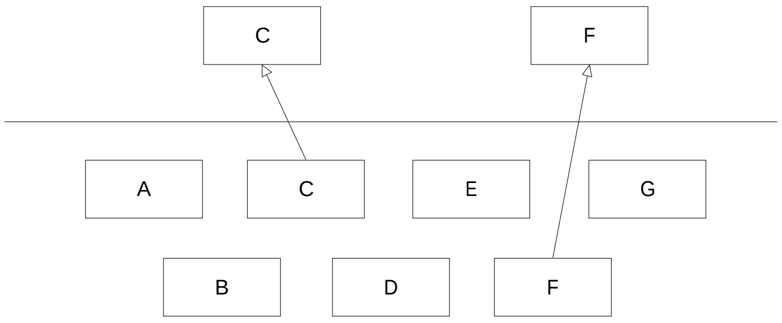


Figure 3.2
An example of selection.

2. Chunking. Chunking combines a sequence of information elements into a new element. For example, in figure 3.3, the elements A to B are chunked into AB. This is useful when the combination creates a more meaningful element, which is hard to recognize when the original elements are considered in isolation. Sometimes, a significant start and end can be recognized in a longer sequence of elements. In that case, the start and end can be taken as the basis for the chunk, leaving out intermediate elements which are essentially implied by the resulting chunk (*e.g.*, in figure 3.3, the elements C, D, and E are chunked into CE).

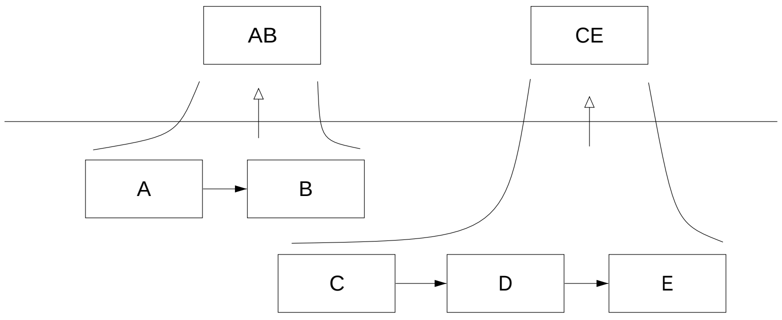


Figure 3.3
An example of chunking.

3. Generalization. By abstracting from minor differences between information elements, a generalization can be created. The resulting element stands for the abstracted information in the original elements. For an example, see figure 3.4, where G1 stands for the generalization of A or B (*e.g.*, because they are of the same type), and G2 stands for C and (D or E), generalizing over the fact that both original elements contain C.

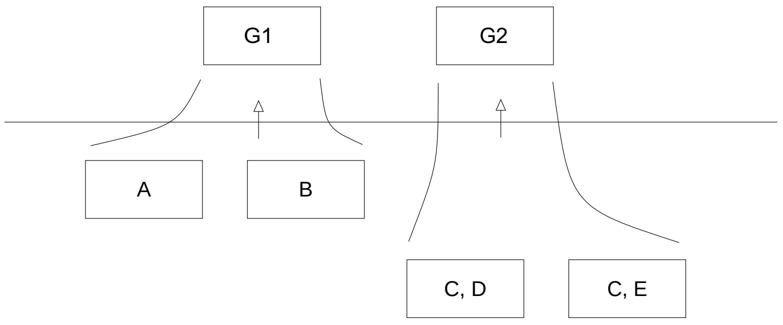


Figure 3.4
An example of generalization.

4. Grouping. This method groups similar elements according to some criteria, and adds a label to this group. Instead of showing the individual elements, the label can be used to refer to the group as a whole (*e.g.*, see figure 3.5, where the label X refers to the group of elements consisting of A, B and C, and Y refers to D, E, F, and G). Grouping, as defined here, is a weaker method than the other methods, because it does not actually reduce the number of elements; a label does not carry the information of the group's elements, it only refers to them.

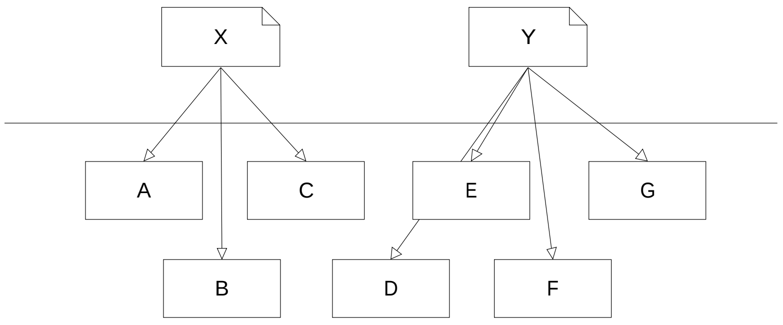


Figure 3.5
An example of grouping.

Note that chunking and generalization are both methods for combining multiple elements to form a new one, although they differ in the requirements on the original elements (sequential relationships for chunking, similarity for generalization). Generalization and grouping are both methods which deal with groups of similar elements, but only generalization transforms them into a new element, while grouping merely provides a label referring to the original elements.

3.5 Aggregation techniques: applying the methods to the five levels

This section describes the aggregation techniques which have been implemented as extra functionality on top of VisiGarp. They specify how the four aggregation methods of section 3.4 can be applied to reduce and reorganize the information from a GARP simulation into a set of data structures on all five levels of aggregation, described in section 3.3. Table 3.2 provides an overview of the aggregation techniques.

Level	Aggregation techniques
1. System state	Classification of causal effects (for selection)
2. Local event	Recognition and grouping of events (selection, chunking & grouping)
3. Path segment	Recognition of segments (chunking) & segment events (selection & chunking)
4. Path	Recognition of cycles & path events (selection & chunking)
5. Global	Transitive reduction (selection), Aggregation of alternative orderings (generalization) & Aggregation of sequence (chunking)

Table 3.2
Aggregation techniques for the five levels of aggregation.

On the system state level, the status of dependencies is analyzed to arrive at a *classification of causal effects*. This allows grouping and selection of those dependencies which have an actual effect, discarding dependencies whose effect does not contribute to the outcome of the simulation. On the local event level, *recognition of events* means that information from adjacent states is selected and combined to form larger chunks of information, specifying meaningful events of various types (allowing grouping by type of event, among other things). On the next level, path segments are recognized, which specify behaviour that necessarily takes place. Additional events are recognized by selection and chunking of lower level events. The same happens on the path level, where, contrary to the path segment level, paths may extend beyond a branching point. When a path contains branching, the events after the branching point do not necessarily occur. Paths also have to be analyzed for the existence of cycles to recognize repetitive behaviour. Finally, on the global level, *transitive reduction* decreases the number of transitions (selection), and *aggregation of alternative orderings* decreases both the number of transitions and states in the state-transition graph by generalizing over reuniting branches. Both are based on the idea that it is often useful to abstract from temporal information if the general story is the same, and it does not matter for later developments in the simulation.

These techniques are presented in detail in the following subsections for each of the five levels of aggregation. To illustrate the working and use of these techniques, a simulation is used throughout the remainder of this chapter as a running example. It is based on a re-implemented model of the Cerrado Succession Hypothesis (CSH) model as originally presented in [Salles and Bredeweg, 1997]. This is the most complex simulation that was used in the VisiGarp experiment, described in section 2.5.3.

3.5.1 System State Level

From the system state point of view, the system is seen at a particular moment or interval in time, during which the behaviour is qualitatively constant. This does not necessarily mean that quantities do not change; a quantity can increase or decrease, which typically causes a transition to a next state. Because this dynamic behaviour is often more interesting than quantities which do not change (at least, this is the best default assumption), this distinction offers a way to select information from a state: only select that part of the model which is related to quantities changing, and hide the rest.

But this crude selection does not take into account the reasons behind the changes of quantities, which can be found in the influences and proportionalities, the causal dependencies between the quantities. The causal network consisting of these dependencies may be complex (involving tens to hundreds of relationships); therefore it is useful to consider meaningful portions of it:

1. A quantity Qx (indirectly) influencing quantity Qy. Special cases of this are feedback loops;
2. A quantity Qx directly influencing all quantities Qy_1 to Qy_n;
3. All quantities Qx_1 to Qx_n directly influencing one quantity Qy.

These three cases enable highlighting of linear propagation of an influence, an influence spreading in multiple directions, and multiple influences combining, respectively. In combination, they can be used to explain why any quantity Qx is increasing, steady, or decreasing. To do this, it is useful to consider the status of the causal dependencies involved.

3.5.1.1 The status of causal dependencies

In this work, all causal dependencies are assigned a status, to distinguish between potential and actual effects. This is related to the difference between causality and causation (e.g., see Hempel [1993], Lehmann [2003]). According to Lehmann, ”causality may be thought of as the *possibility* of a causal relation”, while causation ”may be considered as the *particular* or *instantiated* causal relation” [2003, p.58–59]. In a GARP simulation, the general and particular viewpoints correspond to a causal dependency as specified in a generic model fragment, *i.e.*, $Q_x \xrightarrow{I+} Q_y$, and the same causal dependency instantiated in a particular state of a simulation, with specific values and derivatives instantiated for the quantities Q_x and Q_y, respectively. For example, when $Q_x = positive$, this will have a positive effect on Q_y, indicating that, if no other influences are present, Q_y will increase. When $Q_x = negative$, however, the same (positive) influence will have a negative(!) effect, causing Q_y to decrease. When $Q_x = zero$, there will be no effect on Q_y at all.

In a particular situation, there can be multiple causes whose consequences may interact, which further complicates the story. Consider a situation with multiple quantities Q_1 ... Q_n influencing Q_y in opposing directions. If the positive effects are dominant, Q_y will increase; if the negative effects are dominant, Q_y will decrease; if the positive and negative effects are in balance, there will be no effect on Q_y.

Representing the status of causal dependencies. In GARP, the status of causal dependencies is not explicitly represented. The GARP simulation engine determines the effects of each dependency, but only to a degree necessary to generate all possible successor states. Information about the status of dependencies must therefore be interpreted from the actual values and derivatives of the quantities in a particular state of interest. For example, when Q_y is known to increase, the positive influences on Q_y must be greater than the negative.[1]

In Mallory’s work [1998] on Expound, an explanation generation system for qualitative simulations in the QSIM framework, there is an explicit representation of status, in terms of six causal classifications: *causing, opposing, steady, chattering, push-inc,* and *push-dec* (the last two creating a balance). De Koning only uses two kinds in his approach: *dominant*, for dependencies which have

[1]Note that in quantitative simulations, interpretation is not necessary; there it is simply a matter of calculating the net effect from all individual effect sizes.

the expected effect, and *submissive*, for all dependencies which do not have the expected effect, especially those who are dominated by other effects [de Koning, 1997]. An issue with Mallory's labels is that they are attached to variables, which can pose a problem because the classifications are actually more about the effect of dependencies. In principle, a variable could have both a positive and negative effect on another variable, *e.g.*, via different intermediating variables. With Mallory's method, it is unclear which label to attach in such a case. To avoid this problem, and to keep the ontological distinctions clear, the classification labels are attached to dependencies (as in De Koning's method), not the variables themselves.

Based on these insights, a new set of labels has been developed to specify the status of a causal dependency $Q_x \xrightarrow{I/P_s} Q_y$. In the following definitions of these labels, I stands for influence, P for proportionality, Q for quantity, and s for the sign of the dependency, i.e., positive (+) or negative (-).

Inactive: when no effect is expected, *i.e.*, for

I$_s$: when the value of Q_x is zero

P$_s$: when the derivative of Q_x is zero

Submissive: when the effect is dominated by other effect(s), *i.e.*, for

I$_s$: when the derivative of Q_y is the reverse of what is expected, based on s and the value of Q_x

P$_s$: when the derivative of Q_y is the reverse of what is expected, based on s and the derivative of Q_x

Balanced: when the effect is balanced by other effect(s), *i.e.*, for

I$_s$: when the derivative of Q_y is zero, instead of what is expected, based on s and the value of Q_x

P$_s$: when the derivative of Q_y is zero, instead of what is expected, based on s and the derivative of Q_x

Effective: when the effect has the expected effect, *i.e.*, for

I$_s$: when the derivative of Q_y is what is expected, based on s and the value of Q_x

P$_s$: when the derivative of Q_y is what is expected, based on s and the derivative of Q_x

Dominant: effective, while there is at least one complementary submissive effect on Q_y from Q_z

The first four categories are mutually exclusive and cover the complete set of causal dependencies. The fifth, *dominant*, is a special case of *effective*. A causal effect can be effective either because there are no conflicting effects (there may be other effects in the same direction), or because it is dominant (possibly together with other effects in the same direction) over conflicting submissive effects.
An alternative definition of balanced, which takes into account more detailed information about the relationship between individual effect sizes, is the following:

Balanced*: when there are complementary effects on Q_y from Q_z, and $Q_x = Q_z$ (*i.e.*, the effects are equal in size). This means that if the derivative of Q_y is not zero, there should be other effects explaining it.
This second definition would create problems, however, when there are multiple possibilities for the status of one particular dependency. For example, a *birth* process may be dominant with respect to a conflicting *death* process, but it may be balanced with respect to another process, *e.g.*, *emigration*. This would mean that the status categories are not mutually exclusive anymore. Because this could lead to confusion, the first definition is used in the remainder of this thesis.

The labels described above can be related to Mallory's work, as follows. *Causing* can be mapped to *dominant*, *opposing* to *submissive*, and *steady* to *inactive* without problems. *Push-inc* and *push-dec* can both be mapped to *balanced*–the distinction between increasing and decreasing is considered a separate dimension, and not as different categories. Mallory's category *chattering* is not included. Chattering variables (when it is unknown whether the variable is increasing, steady, or decreasing) are important in the QSIM approach to simulation, but in GARP, modellers can incorporate more kinds of constraints to minimize the amount of unknown information. Furthermore, the label only indicates that the effect is unclear, because the behaviour of the influencing variable is unclear. In the work described in this thesis, the approach is to leave unknown information unknown.

In figure 3.6, an example is shown in the domain of population dynamics, which involves two situations where the same set of causal dependencies has different effects. In the situation on the left, the negative influences of dead1 and emigrated1 (displayed in grey) are submissive to the positive influences of born1 and immigrated1. Number_of1 increases. In the situation on the right (which might follow on from the situation on the left after some time), the status of dependencies is different. Here, all four influences are in balance, and number_of1 is steady. Consequently, the proportional effects of number_of1 on born1, dead1 and emigrated1 are inactive (displayed in grey).[2]

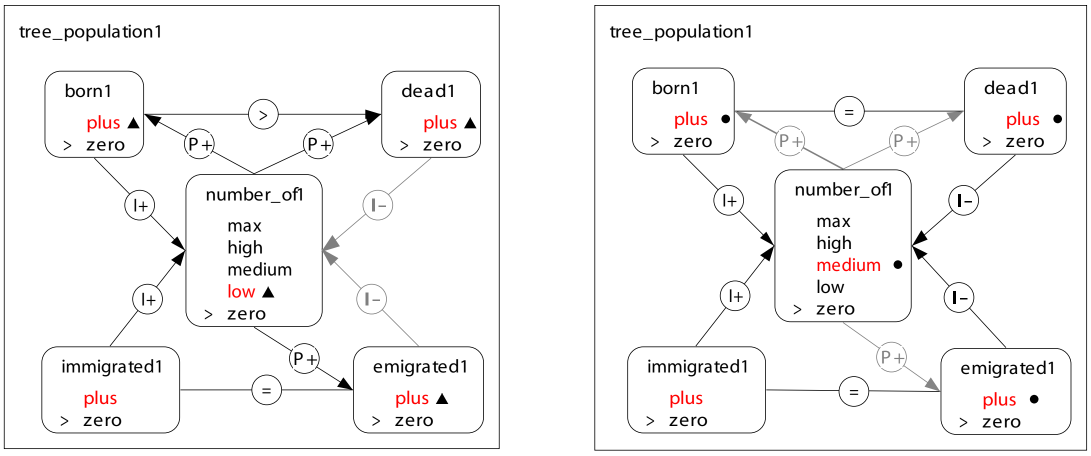


Figure 3.6
The status of dependencies in a tree population in two different situations.

3.5.1.2 Aggregation of causal models

When besides the dependencies themselves also their status is considered, this creates potential for abstraction, as demonstrated by the existing aggregation technique of de Koning [1997], mentioned before on page 62. This technique is discussed here in detail in order to explain how it could be applied to the work in this thesis. The technique works as follows. First, the status of each dependency is marked either dominant or submissive (a less extensive set than defined above). This is used to abstract from all submissive dependencies, and focus only on the effects that lead to actual value events. Second, causal chains are constructed in which non-branching sequences are chunked, and fully corresponding quantities (*i.e.*, which behave in exactly the same way) are grouped together as one. Third, the causal chains which do not directly lead to a state transition from the current state, are discarded. The goal of De Koning's abstraction method was to facilitate hierarchical diagnosis of learners' reasoning, but the approach also seems useful for explanation purposes. However, in order

[2]Note that the (in)equality relationships between born1 and dead1, and between immigrated1 and emigrated1 are not necessary for the system to derive the status of the dependencies. They are merely added to make the example more intuitive to understand.

to better fit the work described in this thesis, the following three changes are proposed to De Koning's abstraction method. (1) Instead of always leaving out all dependencies that do not contribute to an effect, it is now also possible to show why they do not contribute, by showing their status explicitly as submissive, inactive, or balanced. When there are so many dependencies that it is hard to identify the effective ones, showing only the effective ones may still be a good idea, however. (2) Chunking sequences of dependencies and grouping of fully corresponding quantities are both useful, too, but when the aggregated quantities belong to different entities, this may be a reason for keeping them separate. (3) A more important point concerns discarding the causal chains which do not directly lead to a state transition. This is not desirable for explanation purposes, because it may disconnect an effect from its ultimate cause, as indicated by the following example. When an influence is introduced in state S_n, this causes some amount Q (whose value currently lies in some interval, e.g., low) to increase. This increase does not directly lead to a state transition, however (e.g., because another quantities reaches a new value first), but it does so later, e.g., in state S_{n+3}, only then reaching the border of the interval low, and changing to medium. De Koning's mechanism would only include the causal chain from influence to the changing quantity in state S_{n+3}, although the trend was already started in state S_n. Instead, a causal chain should preferably be introduced as soon as the cause occurs, and be discarded only when it does not lead to any transition event later on in the simulation. As De Koning notes, people often make inferences and claims about events happening at some later point in time, not necessarily the first next state [1997, pp. 65–70]. The proposed amendment addresses this concern.

3.5.2 Local Event Level

On the local event level, the system is considered at a moment in time at which some significant change takes place. These significant changes are called *events* in the remainder of this thesis. Like in the work by Mallory [1998], they are constructed by comparing two, or in some cases, three, successive states, and forming *chunks* of information elements which can be interpreted together as an event. The most important kind of events are value and derivative events, but there are also other types of events, involving (in)equality relationships, causal effects, and structural properties of the system. An overview of these kinds of events with some illustrative examples is given in table 3.3.

Table 3.3
The types of event and some typical examples of each.

Event type	**Typical examples**
Value events	Quantity Q crosses a point value, or becomes unknown
Derivative events	Quantity Q starts increasing, or reaches a maximum
(In)equality events	Change from $Q_x < Q_y$ to $Q_x > Q_y$
Correspondence events	Change in value of Q_x corresponds to change in value of Q_y
Causal effects events	Positive effect of $Q_x \xrightarrow{I+} Q_y$ introduced
Structural events	Entity, Attribute, Relation or Quantity introduced or removed
Model fragment events	MF_x introduced, removed, or replaced by MF_y
S-T graph characteristics	Start state, end state, outgoing branching point, incoming branching point

The remainder of this section describes all possible events in detail, defined in terms of lower-level information elements. They are categorized by the kind of information, and accompanied by examples in the context of the CSH simulation, described in section 2.5.1. Because this simulation contains too many events to show at once, the examples in the following subsections involve only the events occurring along a particular path through the simulation (see figure 3.7), focusing on the tree and grass populations in the Cerrado.

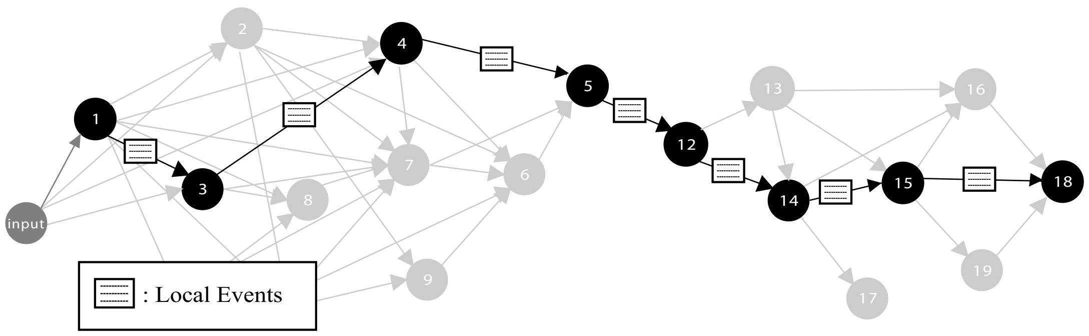


Figure 3.7
A selected path in the CSH simulation.

3.5.2.1 Value and derivative events

Value and derivative events specify how quantities behave. A *value* event takes place whenever a quantity changes from one qualitative value to another. A *derivative* event happens when the derivative of a quantity (min, $zero$ or $plus$, in the GARP framework) changes qualitatively. However, in many cases, these kinds of events occur together. Therefore, they are treated together here. An enumerated list of the events which are recognized is given below. The numbers refer to figure 3.8, which shows a graphical representation of the events in the value history format, as defined in section 2.3.9. The examples in the figure utilize mostly positive derivatives for improved clarity, but of course, mirrored versions of these events with all derivatives reversed also exist.

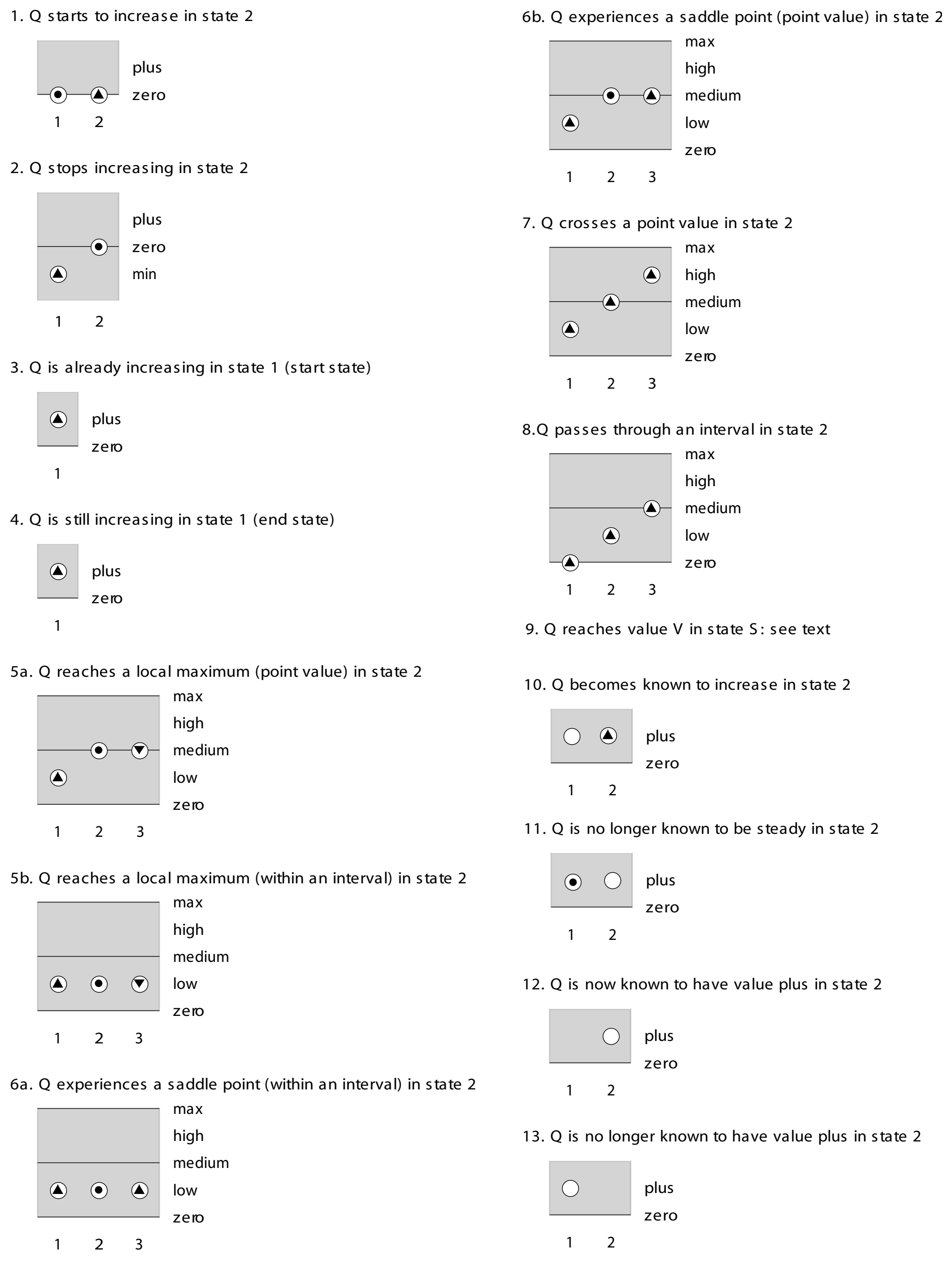

1. Q starts to increase in state 2
plus
zero
1 2
2. Q stops increasing in state 2
plus
zero
min
1 2
3. Q is already increasing in state 1 (start state)
plus
zero
1
4. Q is still increasing in state 1 (end state)
plus
zero
1
5a. Q reaches a local maximum (point value) in state 2
max
high
medium
low
zero
1 2 3
5b. Q reaches a local maximum (within an interval) in state 2
max
high
medium
low
zero
1 2 3
6a. Q experiences a saddle point (within an interval) in state 2
max
high
medium
low
zero
1 2 3
6b. Q experiences a saddle point (point value) in state 2
max
high
medium
low
zero
1 2 3
7. Q crosses a point value in state 2
max
high
medium
low
zero
1 2 3
8.Q passes through an interval in state 2
max
high
medium
low
zero
1 2 3
9. Q reaches value V in state S: see text
10. Q becomes known to increase in state 2
plus
zero
1 2
11. Q is no longer known to be steady in state 2
plus
zero
1 2
12. Q is now known to have value plus in state 2
plus
zero
1 2
13. Q is no longer known to have value plus in state 2
plus
zero
1 2

Figure 3.8
A graphical representation of the value and derivative events in the value history format.

1 Quantity Q starts to increase/decrease (from value V) in state S_2.
State transition $S_1 \rightarrow S_2$
State S_1 : $\delta Q = zero, Q = V$
State S_2 : $\delta Q = plus/min$

Example 1 in figure 3.8 shows a simulation with a quantity Q whose derivative is zero in state 1, and plus in state 2, which comes next. Then, a typical way of interpreting these two facts is that Q starts increasing in state 2. Note that this interpretation can only be made when these two pieces of information, from two successive states, is combined to form a larger chunk of information.

2 Quantity Q stops increasing/decreasing (at value V) in state S_2.
State transition $S_1 \rightarrow S_2$
State S_1 : $\delta Q = plus/min$
State S_2 : $\delta Q = zero, Q = V$

Example 2 in figure 3.8 shows a simulation with a quantity Q whose derivative changes from plus in state 1 to zero in state 2. In other words, Q stops increasing. In the example, the value also changes (from min to zero), but this is not required for the definition of this derivative event.

The next two events involve information from single states. Strictly speaking, they are not events in the sense that a change of value or derivative has taken place–only one state is considered. However, they are considered important because they mark the start and end of the behaviour of a quantity.

3 Quantity Q is already increasing/decreasing in state S
State S is a start state[3]
State S : $\delta Q = plus/min$

A quantity Q may be increasing or decreasing from the start, because its derivative is specified as such in the scenario, or because it can be derived from other information in the scenario. If this is the case, for any start state S, this is recorded as a special kind of event, indicating that quantity Q is already increasing or decreasing. It is interesting to note because the increase or decrease may cause changes later on.

4 Quantity Q is still increasing/decreasing in state S State S is a final state
State S : $\delta Q = plus/min$

A quantity Q may still be increasing or decreasing at a final state. Because it is often interesting to know how a quantity behaves at the end of a simulation, it is necessary to record this as an event, even if no change in derivative has occurred.

The next two events involve chunking information from *three* successive states.

5 Quantity Q reaches a local extreme V (maximum/minimum) in state S_2
Two consecutive state transitions $S_1 \rightarrow S_2 \rightarrow S_3$
State S_1 : $\delta Q = plus/min$
State S_2 : $\delta Q = zero, Q = V$
State S_3 : $\delta Q = min/plus$

[3]The notion of start and end states is discussed and defined formally in section 3.5.2.7.

A local extreme happens when the value of a quantity reaches a (local) maximum or minimum. A maximum can be inferred when a simulation includes three successive states, S_1, S_2 and S_3, in which quantity Q is increasing, stops increasing, and starts decreasing again, respectively (a minimum requires all changes in the reverse direction). The event is called *local* extreme, to distinguish it from *path*, or *global* extremes, which denote the highest/lowest value in a whole path, or simulation, respectively. Figure 3.8 shows two examples of a maximum: example 5a shows a maximum taking place at a point value (medium), while example 5b shows a maximum within an interval value (low). The last example shows that a maximum can occur without any qualitative change in value.

6 Quantity Q experiences a saddle point V in state S_2 (while increasing/decreasing)
Two consecutive state transitions $S_1 \rightarrow S_2 \rightarrow S_3$
State S_1 : $\delta Q = plus/min$
State S_2 : $\delta Q = zero, Q = V$
State S_3 : $\delta Q = plus/min$

A *saddle point* event takes place when, after becoming steady momentarily, the quantity resumes its change in the same direction as before. The mathematical definition which inspired this applies to surfaces as well as functions, but here, an interpretation is given for qualitative functions. [4] In figure 3.8, again two examples are shown: example 6a shows a saddle point taking place at an interval value (low), while example 6b shows that this event can also occur at a point value.

Note that a local maximum can be regarded as a combination of a *stop increasing* and a (directly following) *start decreasing* event chunked together. Likewise, a saddle point event is a chunk consisting of a *stop increasing* and *start increasing* event. These smaller events are discarded however, favouring the larger events.

The events discussed so far involve one or more changes in the derivative. In some cases the value may also change, but this is not required by the definition of these events. If V is an interval value, Q may have this value in all states S_1, S_2 (and S_3) in order for any of these events to occur. But there are also events in which the derivative does not change, but the value does. Here, the difference between point and interval values becomes extra interesting. Because points and intervals alternate, a quantity Q which keeps increasing or decreasing through two consecutive values in its quantity space will either cross a point value, or an interval value. This makes it possible to refer to two consecutive value changes as one event instead of two, which is another application of chunking.

7 Quantity Q crosses the point value V_2 in state S_2 (while increasing/decreasing)
State transitions: $S_1 \rightarrow S_2 \rightarrow S_3$
$S_1 : Q = V_1, \delta Q = plus/min$
$S_2 : Q = V_2, \delta Q = plus/min, point\_value(V_2)$
$S_3 : Q = V_3$
$V_1 \neq V_2 \neq V_3$

When a quantity Q keeps increasing or decreasing over two consecutive states, and the second value is a point value, this value is crossed. Although three different values are present in the definition of this event, only the middle, point value V_2, is explicitly represented in the event, because it provides the focus. Not much information is lost by this process of selection,

[4]Eric W. Weisstein. "Saddle Point." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com

however, because the previous value V_1 must have been involved in a prior event (*i.e.*, Q has started to change from that value V_1), and the next value V_2 will be involved in another event later on, whether Q becomes steady, unknown, or crosses another value. Defining the event in this way puts a focus on what is happening, which a list of all values and derivatives over the consecutive states does not give.

8 Quantity Q passes through interval V_2 in state(s) $S_{2(i)}$ (while increasing/decreasing)
Path consists of S_1, S_2, (possibly more states S_{2i}), and S_3.
$S_1 : Q = V_1, \delta Q = plus/min$
$S_2 : Q = V_2, \delta Q = plus/min, interval\_value(V_2)$
$S_{2i} : Q = V_2, \delta Q = plus/min$
$S_3 : Q = V_3$
$V_1 \neq V_2 \neq V_3$

Although very similar to the ’cross point value’ event, this is not a strictly local event, because it may involve more than one state in which quantity Q keeps increasing or decreasing within interval V_2. This is possible because a quantity may be increasing within an interval value for several states S_{2i} before it reaches the next value in state S_3. If this is the case, Q must have the same value and derivative in state(s) S_{2i} as in S_2. Note that this is mathematically impossible in the case of the ’cross point value’ event, because a quantity can not be increasing at a point value for more than one state.

9 Quantity Q reaches value V_2 in state S_2.
State transition $S_1 \rightarrow S_2$
State S_1: Q = V_1, $\delta Q = plus/min$
State S_2: Q = V_2

When a quantity Q changes value because its derivative is plus or min in state S_1, but none of the events described above apply, a ’reaches value’ event is recorded, involving two consecutive states.

In a way, the event ’crossing value’ can be regarded as two ’reach value’ events chunked together. The two lower level events are discarded in favour of the ’crossing value’ event, if present. So, in determining which events take place, the algorithm first searches for the ’crossing value’ events, and only if it can’t find anymore, it will look for remaining ’reach value’ events.

The following events recognize the possibility that the value or derivative of a quantity is unknown and becomes known, or vice versa. As such, these events are epistemological in nature; they do not necessarily correspond to a real change in the world.

10 Quantity Q is now known to increase/decrease/be steady
State transition $S_1 \rightarrow S_2$
State $S_1 : \delta Q = unknown$
State $S_2 : \delta Q = plus/min/zero$

In this case, the derivative of quantity Q becomes known, as is shown by the appearance of the derivative symbol in the example in the figure.

11 Quantity Q is not longer known to increase/decrease/be steady
State transition $S_1 \rightarrow S_2$
State $S_1 : \delta Q = plus/min/zero$
State $S_2 : \delta Q = unknown$

Here, the derivative of quantity Q becomes unknown.

12 Quantity Q is now known to have value V
State transition $S_1 \rightarrow S_2$
State $S_1 : Q = unknown$
State $S_2 : Q = V$

This event occurs when the value of quantity Q was not known first (hence, no circle indicating the value in the figure example), but becomes known in the next state.

13 Quantity Q is not longer known to have value V
State transition $S_1 \rightarrow S_2$
State $S_1 : Q = V$
State $S_2 : Q = unknown$

In this case, the value of quantity Q becomes unknown, hence the disappearance of the value circle in the example in the figure.

The following two events (not depicted in figure 3.8) are *generalizations* of more specific events described above. They are introduced to enable the recognition of developments in the value history on higher levels of aggregation.

14 Start of increase/decrease for Quantity Q in state S

This event is a generalization of the three kinds of events indicating the start of an increase or decrease: start to increase/decrease (event no. 1), already increasing/decreasing (3), and becomes known to increase/decrease (10).

15 End of increase/decrease for Quantity Q in state S

This event is a generalization of a number of events, which have in common that the change in a certain direction does not continue any further: stop increasing/decreasing (event no. 2); still increasing/decreasing (4); local extreme (maximum/minimum) (5); saddle point (6); and is no longer known to increase/decrease (11).

Running Example. In the CSH simulation, the running example, there are 553 value and derivative events occurring across the various states and transitions in the state-transition graph shown in figure 3.7. In the path selected in the figure, 109 of these events occur. Focusing on only the quantity number_of3 (which represents the amount of grass in the Cerrado), the following 6 events remain:

```
number_of3 is max and already decreasing in state 1
number_of3 passes through the interval high while decreasing [4, 5, 12]
number_of3 crosses the point medium while decreasing in state 12
number_of3 passes through the interval low while decreasing [12, 14, 15]
number_of3 has a local minimum of zero in state 15
number_of3 starts to increase at zero in state 18
```

3.5.2.2 (In)equality events

Besides values, also mathematical relationships between values can play an important role in the simulation - for brevity, they are called (in)equalities in the remainder of this chapter. In GARP, the following kinds of (in)equalities are represented: equal ($X = Y$), greater ($X > Y$), smaller ($X < Y$), greater_or_equal ($X >= Y$), and smaller_or_equal ($X <= Y$). In these (in)equalities, X and Y denote a quantity, a particular value, or a formula (sum or difference). The lefthand side is usually a quantity, however. Note that when the lefthand side of an equality $X = Y$ is a quantity, and the righthand side (Y) is a value from X's quantity space, the equality statement mathematically resembles a value definition.

There are also (in)equalities which hold between the derivatives of particular quantities: d_equal ($\delta X = \delta Y$), d_greater ($\delta X > \delta Y$), d_smaller ($\delta X < \delta Y$), d_greater_or_equal ($\delta X >= \delta Y$), and d_smaller_or_equal ($\delta X <= \delta Y$).

Mathematical dependencies between quantity values or derivatives can change, just like values and derivatives can change. This can be the cause of other events in the simulation, such as the introduction or removal of an effect or a model fragment. To determine (in)equality events, two consecutive states are compared. The following three (in)equality events are specified:

1. Event = (in)equality_changed(($S_1 \rightarrow S_2$), Ineq1, Ineq2).
2. Event = (in)equality_introduced(($S_1 \rightarrow S_2$), Ineq2).
3. Event = (in)equality_removed(($S_1 \rightarrow S_2$), Ineq1).

If both states contain a different (in)equality pertaining to the same quantities, and this constitutes a valid transition, the change is recorded as an *(in)equality changed* event (event no. 1). The valid (in)equality transitions are listed in table 3.4. When state S_2 contains a dependency Ineq2, which was not yet present in state S_1, and there is no *(in)equality_changed* event involving Ineq2, then it was newly introduced (event no. 2). When state S_1 contains a dependency Ineq1 while state S_2 does not, and there is no *(in)equality_changed* event involving Ineq1, then it has been removed (event no. 3). For the start state(s), there is no predecessor for comparison, so here, only the introduction of (in)equalities is recorded. For end states, only the removal of (in)equalities is recorded.
Note that a change such as from $X < Y$ to $X > Y$ (or vice versa) does not constitute a valid local event, because this would break the continuity rules in GARP, which command that the values become equal first.

Event description	S_1	S_2
from equal to greater	$X = Y$	$X > Y$
from equal to greater_or_equal	$X = Y$	$X >= Y$
from equal to smaller	$X = Y$	$X < Y$
from equal to smaller_or_equal	$X = Y$	$X <= Y$
from greater to equal	$X > Y$	$X = Y$
from greater to greater_or_equal	$X > Y$	$X >= Y$
from greater_or_equal to equal	$X >= Y$	$X = Y$
from greater_or_equal to greater	$X >= Y$	$X > Y$
from smaller to equal	$X < Y$	$X = Y$
from smaller to smaller_or_equal	$X < Y$	$X <= Y$
from smaller_or_equal to equal	$X <= Y$	$X = Y$
from smaller_or_equal to smaller	$X <= Y$	$X < Y$
from d_equal to d_greater	$\delta X = \delta Y$	$\delta X > \delta Y$
from d_equal to d_greater_or_equal	$\delta X = \delta Y$	$\delta X >= \delta Y$
from d_equal to d_smaller	$\delta X = \delta Y$	$\delta X < \delta Y$
from d_equal to d_smaller_or_equal	$\delta X = \delta Y$	$\delta X <= \delta Y$
from d_greater to d_equal	$\delta X > \delta Y$	$\delta X = \delta Y$
from d_greater to d_greater_or_equal	$\delta X > \delta Y$	$\delta X >= \delta Y$
from d_greater_or_equal to d_equal	$\delta X >= \delta Y$	$\delta X = \delta Y$
from d_greater_or_equal to d_greater	$\delta X >= \delta Y$	$\delta X > \delta Y$
from d_smaller to d_equal	$\delta X < \delta Y$	$\delta X = \delta Y$
from d_smaller to d_smaller_or_equal	$\delta X < \delta Y$	$\delta X <= \delta Y$
from d_smaller_or_equal to d_equal	$\delta X <= \delta Y$	$\delta X = \delta Y$
from d_smaller_or_equal to d_smaller	$\delta X <= \delta Y$	$\delta X < \delta Y$

Table 3.4
The events concerning (in)equality changes between quantity values (top) and quantity derivatives (bottom).

Running Example. In the example simulation, there are 380 (in)equality events. In the selected path, 90 of these events occur. Focusing on the (in)equalities involving the quantity number_of3 (the amount of grass), the following 10 events remain:

```
0 -> 1:
   introduced: d(number_of3) < zero
   introduced: number_of3 > zero
   introduced: number_of3 > medium
1 -> 3:
   no (in)equality events
3 -> 4:
   no (in)equality events
4 -> 5:
   no (in)equality events
5 -> 12:
   changed:    number_of3 > medium   ->  number_of3 = medium
   introduced: number_of3 < max
12 -> 14:
   changed:     number_of3 = medium    ->  number_of3 < medium
   removed:     number_of3 < max
14 -> 15:
   changed:     d(number_of3) < zero  ->  d(number_of3) = zero
   changed:     number_of3 > zero     ->  number_of3 = zero
15 -> 18:
   changed:     d(number_of3) = zero  ->  d(number_of3) > zero
```

The first three events show which (in)equalities are present (introduced) in the start state 1. Then, for several transitions, nothing happens with respect to the (in)equalities involving number_of3. From transition 5→12 onwards, several (in)equality events occur, involving the introduction of new (in)equalities, the removal of old ones, and changes from one (in)equality to another. Some involve the value of number_of3, others involve its derivative. Some (in)equality statements may seem superfluous because they can be mathematically derived from another, *e.g.*, when number_of3 $>$ medium, it is logically also true that number_of3 $>$ zero. The reason for their occurrence is that these inequalities appear as the conditions of different model fragments.

3.5.2.3 Correspondence Events

As described in section 2.1, there are four types of correspondence dependencies in GARP: undirected and directed value correspondences (denoted by V and Vˆ, respectively), and undirected and directed quantity correspondences (Q and Qˆ, respectively). There are two categories of correspondence events, for all four kinds of correspondence dependency. The first category just registers whether a correspondence dependency is introduced or removed, and contains the following two events:

1. Event = CorrespondenceDependency introduced

2. Event = CorrespondenceDependency removed

The second category registers whether the correspondence is actually relevant, *i.e.*, the quantities currently have or reach the values speficied in the correspondence dependency. This involves the recognition of value events, as specified in section 3.5.2.1. There are two events of the second category:

1. Event = ValueEventX corresponds to ValueEventY (KindOfCorr)

2. Event = ValueEventX corresponds to CorrIntroEvent

The first event implies that a quantity Q_x experiences a value event that corresponds with a value event experienced by another quantity Q_y, via a correspondence of type KindOfCorr (V/Vˆ/Vrev/Q/Qˆ/Qrev). For an undirected correspondence, an extra event is registered with ValueEventX and ValueEventY reversed, with KindOfCorr = Vrev (for a reversed value correspondence), or Qrev (for a reversed quantity correspondence). This way, either the normal or the reversed version can be presented, based on the order which best fits the line of reasoning.

The second event implies that a quantity Q_x experiences a value event at the same time that a correspondence dependency is introduced involving that quantity value.

Running Example. In the example simulation, there are 126 correspondence events of the first category, and 241 of the second category. In the selected path, 29 and 58 of these events occur, respectively. Focusing on the correspondences involving the quantities number_of1 and number_of3, the first 22 events denote correspondences which are introduced in state 1. The rest is shown below.

```
1 -> 3:
   no correspondence events
3 -> 4:
   no correspondence events
4 -> 5:
   cover1: zero->low corresponds to number_of1: zero->low (Q^)
5 -> 12:
   cover1: low->medium corresponds to number_of1: low->medium (Q^)
12 -> 14:
   cover1: medium->high corresponds to number_of1: medium->high (Q^)
14 -> 15:
   number_of3: low->zero corresponds to growth3: min->zero (Vrev)
   number_of3: low->zero corresponds to introduced:
     zero(number_of3) V zero(growth3)  (V)
   born3: plus->zero corresponds to number_of3: low->zero (V^)
   dead3: plus->zero corresponds to number_of3: low->zero (V^)
   emigrated3: plus->zero corresponds to number_of3: low->zero (V^)
   growth3: min->zero corresponds to number_of3: low->zero (V)
   born3: plus->zero corresponds to introduced:
     zero(number_of3) V^ zero(born3)  (V^)
   growth3: min->zero corresponds to introduced:
     zero(number_of3) V zero(growth3)  (V)
15 -> 18:
   cover1: high->max corresponds to number_of1: high->max (Q^)
```

3.5.2.4 Causal Effects Events

As defined in section 3.5.1.1, the status of a causal dependency describes whether it actually has an effect or not. Therefore, changes in the status of causal dependencies are important to consider when explaining what happens to quantity values and derivatives. In the following definitions of causal events, Dir stands for *positive* or *negative*, unless the status is inactive. EffectStatus, Eff1, and Eff2 stand for *inactive, submissive, balanced*, or *effective*.

1 changed($S_1 \rightarrow S_2$, effect_status(causal_relation(Q1,R,Q2), from(Dir1, Eff1), to(Dir2, Eff2)))

The above event is recorded when the status of an effect (via the same causal dependency) changes from one state to the next. Because GARP's continuity rules govern the changes of quantity values and derivatives, the only changes that can actually occur are given in table 3.5.

2 introduced($S_1 \rightarrow S_2$, effect_status(causal_relation(Q1,R,Q2), Dir, EffectStatus)).

Whenever a new dependency is introduced (by a model fragment which has become applicable), its status is determined, and the above event is recorded.

3 removed($S_1 \rightarrow S_2$, effect_status(causal_relation(Q1,R,Q2), Dir, EffectStatus))

This event is recorded when a dependency is removed from one state to the next.

Status in S_1	**Status in S_2**
inactive	pos/neg effect
	pos/neg balanced
	pos/neg submissive
pos/neg submissive	pos/neg balanced
	inactive
pos/neg balanced	pos/neg submissive
	pos/neg effect
	inactive
pos/neg effect	inactive
	pos/neg balanced

Table 3.5
The possible changes in status of a dependency between two states $S_1 \rightarrow S_2$.

Running Example. In the example simulation, there are 455 events involving causal effects. In the selected path, 92 of these events occur. Now, the focus is on the events for the quantity number_of1, which represents the number of trees in the Cerrado. This results in the following 15 events:

```
0 -> 1:
   introduced: number_of1 P+ born1: inactive
   introduced: number_of1 P+ cover1: inactive
   introduced: number_of1 P+ dead1: inactive
   introduced: number_of1 P+ emigrated1: inactive
   introduced: number_of1 P+ growth1: inactive
1 -> 3:
   introduced: immigrated1 I+ number_of1: positive effect
   changed: number_of1 P+ born1: inactive becomes positive effect
   changed: number_of1 P+ cover1: inactive becomes positive effect
   changed: number_of1 P+ dead1: inactive becomes positive effect
   changed: number_of1 P+ emigrated1: inactive becomes positive effect
   changed: number_of1 P+ growth1: inactive becomes positive effect
3 -> 4:
   no causal_events
4 -> 5:
   introduced: born1 I+ number_of1: positive effect
   introduced: dead1 I- number_of1: negative submissive effect
   introduced: emigrated1 I- number_of1: negative submissive effect
   introduced: growth1 I+ number_of1: positive effect
5 -> 12:
   no causal_events
12 -> 14:
   no causal_events
14 -> 15:
   no causal_events
15 -> 18:
   no causal_events
```

The five events at the start show that there are already proportional relationships from number_of1 to several other quantities, but these are currently inactive (because number_of1 is steady). In transition

1→3, an influence is introduced from immigrated1, which has a positive effect on number_of1. The resulting change in the derivative of number_of1 is propagated by the positive proportionalities, so all quantities involved start to increase. In transition 4→5, four additional influences are introduced (because number_of1 has reached the value low, which is higher than zero). Immigration, birth, death, and emigration are the four elementary processes in population ecology; growth is an aggregate process which is defined in terms of the elementary processes. From that point onwards, nothing changes to the causal model for the tree population.

3.5.2.5 Structural Events

Whenever something happens to the structure of the system, in terms of an entity, attribute, attribute relation, or quantity appearing or disappearing, this is recorded as an event, as shown in table 3.6. Note that when a quantity appears, this does not necessarily imply that its value or derivative becomes known–hence, these are specified as different kinds of events (*cf.* section 3.5.2.1).

Type of primitive	**Event**
Entity	Entity appears
	Entity disappears
Attribute	Attribute appears
	Attribute disappears
Attribute relation	Attribute relation appears
	Attribute relation disappears
Quantity	Quantity appears
	Quantity disappears

Table 3.6
The structural events between two states $S_1 \rightarrow S_2$.

Example. An example of an attribute relation disappearing and another appearing is the following, from a simulation modelling a physics experiment in which a piston in a container moves outwards because the gas inside the container expands.[5]

removed:	R:	piston1 in container1
introduced:	R:	piston1 outside container1

In line with the recognition of meaningful units of information, it would be better to recognize this as one event, *i.e.*, the relation ’in’ is replaced by ’outside’ the container. However, this has not been implemented, because there is currently no way for the system of knowing that the relations ’in’ and ’outside’ are indeed related. The reason for this is that the relation labels are freely chosen by the domain modeller. More elaborate capabilities for reasoning about structure would require extensions to the GARP framework, for example by adding a set of predefined structural relations, together with knowledge about how they relate to each other. This falls outside the scope of this research, however.

[5]In the CSH simulation, the system structure is stable, so after the introduction of all quantities, entities, attributes and relations at the start state, no structural events take place.

3.5.2.6 Model fragment events

Model fragments play a crucial role in the simulation process, as they are responsible for introducing new information related to particular situations or processes. Therefore, model fragment events are useful for explanation purposes. To determine events concerning model fragments, two consecutive states S_1 and S_2 are compared to see which model fragments have been added, and which ones have been removed in state S_2. To determine whether some removals and additions are related to each other, model fragments which apply to the same entities (the variables they refer to in their predicate definition) are grouped together in a list (MFList). When a removal event and an addition event refer to the same entities, these events are discarded, and a *replacement* event is recorded instead. This leads to a number of events of the following form:

1. MFEvent = added($S_1 \rightarrow S_2$, Entities2, MFList2)
2. MFEvent = removed($S_1 \rightarrow S_2$, Entities1, MFList1)
3. MFEvent = replaced($S_1 \rightarrow S_2$, Entities, MFList1, MFList2)

Thus, for every entity, or combination of entities, there is at most one separate model fragment event, which lists all model fragments that are added, removed, or replaced for that entity, or combination of entities.

Running Example. In the example simulation, there are 108 model fragment events, involving 631 model fragments. In the selected path, 24 of these events occur (involving 146 model fragments). Below, the (6) model fragment events for the tree population are shown.

```
0 -> 1:
   introduced:
            assume_open_population(tree_population1)
            determine_born_while_non_existing(tree_population1)
            determine_dead_while_non_existing(tree_population1)
            determine_emigrated_while_non_existing(tree_population1)
            determine_growth_while_non_existing(tree_population1)
            non_existing(tree_population1)
            non_existing_and_steady(tree_population1)
            population(tree_population1)
            tree_population(tree_population1)
1 -> 3:
   replaced:
            non_existing_and_steady(tree_population1)
   by:
            colonization(tree_population1)
3 -> 4:
   no MF events for tree_population1
4 -> 5:
   replaced:
            colonization(tree_population1)
            determine_born_while_non_existing(tree_population1)
            determine_dead_while_non_existing(tree_population1)
            determine_emigrated_while_non_existing(tree_population1)
            determine_growth_while_non_existing(tree_population1)
            non_existing(tree_population1)
   by:
            assume_BornInflow_DeadOutflow_correspondence(tree_population1)
            assume_Immigrated_equal_to_Emigrated(tree_population1)
            born_greater_than_dead(tree_population1)
            decreasing_fire_effect_on_tree(tree_population1)
            emigration(tree_population1)
            existing(tree_population1)
            immigrated_equal_to_emigrated(tree_population1)
            immigration(tree_population1)
            increasing_population(tree_population1)
            low_size_A(tree_population1)
            mortality(tree_population1)
            mortality_for_tree_in_Cerrado(tree_population1)
            natality(tree_population1)
            natality_for_tree_in_Cerrado(tree_population1)
            population_growth(tree_population1)
            positive_population_growth(tree_population1)
5 -> 12:
   replaced:
            low_size_A(tree_population1)
   by:
            medium_size_A(tree_population1)
12 -> 14:
   replaced:
            medium_size_A(tree_population1)
   by:
            high_size_A(tree_population1)
14 -> 15:
   no MF events for tree_population1
15 -> 18:
   replaced:
            high_size_A(tree_population1)
   by:
            maximum_size_A(tree_population1)
```

Initially, there “is” a “tree population” which is non-existent, and steady, as specified in the four last model fragments in the first event. The assumption model fragment *assume_open_population* is a condition for colonization and immigration to become possible later. The rest of the model fragments specify constraints about the value or derivative of other quantities when number_of1 is zero. In transition 1→3, the process model fragment colonization replaces the non_existing_and_steady model fragment. Then, in transition 4→5, all model fragments which related to number_of1 being zero are replaced by new ones, which are based on the condition that number_of1 $>$ zero. This includes, besides several model fragments implementing assumptions and constraints, the generic process model fragments for natality, mortality, immigration, and emigration, as well as specialized versions for the tree population. From that moment, the rest of the model fragment events involve only descriptive model fragments related to the value of number_of1.

3.5.2.7 State-transition graph characteristics

On a more abstract level, the state-transition graph can also be considered to contain events, such as the occurrence of a branching point, or the start and end of a path in the simulation. These events do not directly relate to the system itself, but rather to the qualitative simulation thereof. As such, they play a role in determining the time frames which are used on the higher level views: the path segment, path and global view. Because they are all defined in terms of individual states, and because they have a different ontological status than the other event types, they are called characteristics, rather than events. The following state-transition graph characteristics are considered:

- Start state: a state which is a direct interpretation of the input model
- End state: a state without any successor state
- Outgoing branching point: a state with two or more successor states
- Incoming branching point: a state with two or more predecessor states

Figure 3.9 shows the state-transition graph for the example simulation, with labels attached to states where the above characteristics occur: s for start state, e for end state, o for outgoing branching point and i for incoming branching point.

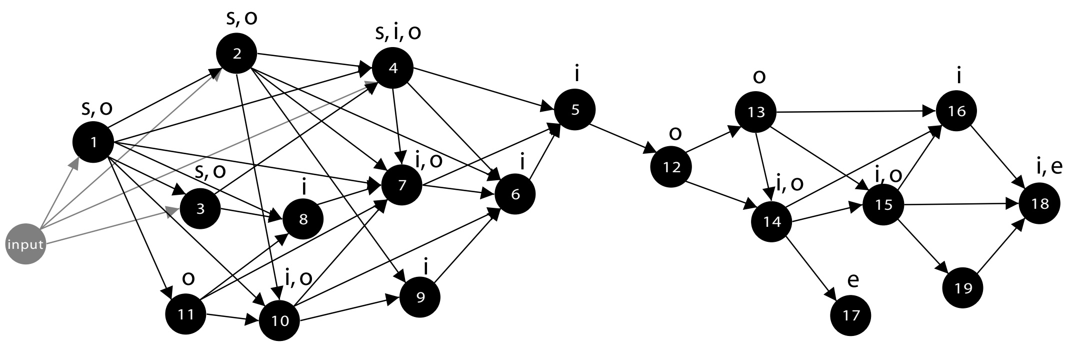


Figure 3.9
The state-transition graph for the CSH simulation, with labels for start state (s), end state (e), outgoing branching point (o), and incoming branching point (i).

It should be noted that using the definition above, there can be multiple *start states* in a simulation if the input state (the scenario) leaves room for multiple interpretations, *i.e.*, when a quantity value or derivative is underspecified. Sometimes, different start states lead to genuinely different behaviours, but in some cases they are just different entry points in a sequence, and it is preferable to focus on just one of these start states. For example, the CSH simulation contains four start states: 1, 2, 3 and 4 (see figure 3.9). In the scenario, the values for the quantities $number\_of1(trees), number\_of2(shrubs)$ and $number\_of3(grass)$ have been set to $zero, zero$ and max, respectively, but the derivatives of these quantities have not been specified. The derivative of $number\_of3$ can be derived to be negative from information in the scenario, but the derivatives of $number\_of1$ and $number\_of2$ can be either $plus$ or $zero$ (not min, because both quantities have their lowest value possible so they can not decrease). Two quantities, multiplied by two possibile derivatives makes four possibile start states. In the transition $1 \rightarrow 2$, the shrub population starts increasing, while in the transition $1 \rightarrow 3$, the tree population starts increasing. In state 4, both populations have started to increase. When considering the transitions between the four start states, which make up the paths $1 \rightarrow 2 \rightarrow 4$, $1 \rightarrow 3 \rightarrow 4$, and $1 \rightarrow 4$, it is clear that state 1, in which the two derivatives are still zero, precedes the other three states. Therefore, it would make sense to consider state 1 as the only start state in figure 3.9.[6] An alternative, stricter, definition for *start state* which would take this into account is: a state without any predecessor state, except the input state. However, this definition leads to problems when there is a cyclic behaviour leading back to this (start) state, because this state would then no longer comply with the definition. Therefore, the original definition is preferred. Nevertheless, the number of start states can often be reduced using *transitive reduction* or *aggregation of alternative orderings*, which are discussed in section 3.5.5.1.

3.5.2.8 Grouping of events

In the examples in the previous subsections, events were presented in hierachically structured groups. First, they were grouped by type of event *i.e.*, *value and derivative events*; second, by state-transition(s), *i.e.*, transition $1 \rightarrow 3$; third, by system part, *i.e.*, the quantity *number_of1*.

Alternative structures for combining these groupings are also possible. For example, grouping events first by transition, and only second by event type, gives a clear chronological history of what happens. Grouping by event type as the first ordering principle makes it easier to search for a particular type of event, in case it is expected, but unknown when or where it occurs.

The labels for the transitions, event types, and system parts serve as flags to indicate what kind of information is presented there. In an interactive software system it is also possible to show only the labels to refer to the underlying groups of events. By selecting a particular label, that group can then be investigated in detail.

3.5.3 Path Segment Level

A *path segment* is a sequence of states, connected by transitions, which does not extend beyond a branching point. It represents the *necessary* behaviour of the system modelled during a particular stretch of time. The behaviour is *necessary*, according to the model, in the sense that whenever the system is in a state at the beginning of a path segment, it will behave accordingly (because there are no other possibilities), and end up in the state at the end of the segment.

Because a path segment is a special kind of path, the event definitions for the path level (see section 3.5.4) also apply to this level. The distinction between path segments and paths becomes important again in section 3.5.5 about the global level.

3.5.4 Path Level

A *path* is a sequence of states, connected by transitions. It represents a *possible* behaviour of the system modelled during a particular stretch of time. The behaviour is *possible*, because a path may contain branching points, indicating that other behaviours might also be possible. A *full path* is a path from a start state to an end state. It represents a possible behaviour of the system modelled from begin to end. On the path level, additional value and derivative events are recognized. Other than that, a path (segment) contains a sequence of event sets occurring at the local level. The additional value and derivative events on the path level are listed below:

[6]Note that the numbering of states is a result of the order in which GARP generates them, which does not necessarily correspond to a logical time sequence. Therefore, the numbers should only be regarded as identifiers.

1. The highest/lowest value V of quantity Q in path P is reached in state(s) $S_{(i)}$
2. Quantity Q increases/decreases from V_1 in S_1 to V_2 in S_2 in path P
3. Quantity Q fluctuates between V_1 and V_2 in path P

Because a path contains multiple consecutive states, which represent a possible behaviour during a particular time frame, it makes sense to compare the values of a quantity Q between these states, and look for the highest/lowest value in the sequence (see event nr. 1, above). When the value of Q is unknown in any of the states in the path, it is impossible to determine the highest/lowest value. In that case, no event of this kind is recorded. This event should not be confused with the local maximum/minimum on the local level, although they may correspond; the highest/lowest value does not require the derivative to turn around, as in the case of a local extreme.

Event nr. 2 records the start and end of each increase and decrease in path P. It is created by chunking together a *start of increase/decrease* and *end of increase/decrease* event, which were defined at the local level. Hereby, the intermediate value changes, if present, are abstracted away. Note that S_1 and S_2 are not necessarily the start and end state of P, but S_1 always comes before S_2 in P.

Event nr. 3 is recorded when both a local maximum and a local minimum occur for the same quantity Q in path P, and these are not within the same interval value; in other words, when quantity Q is *fluctuating* between two values. Again, the method of chunking is applied. In this case, even more information (about what happens in between the local extremes) is abstracted from. No details about states and the direction of change are registered with this event; these are already present in the events described above.

Example. To give an example of these three events, consider the value history in figure 3.10. For the (cyclic) path displayed in the figure ($1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 1$), the highest value is $plus$, reached in state $2, 3$, and 4; the lowest value is min, reached in state $6, 7$, and 8. Quantity Q increases from min in state 8 to $plus$ in state 3 (via state 1 and 2), and decreases from $plus$ in state 4 to min in state 7. Quantity Q fluctuates between $plus$ and min.

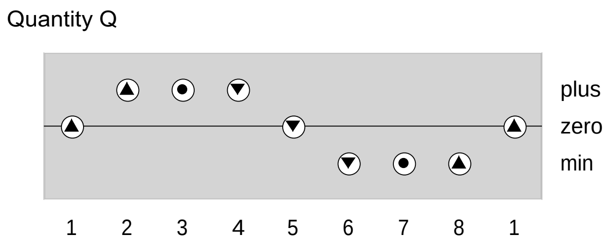


Figure 3.10
Value history for a quantity fluctuating between the values plus and min.

Note that in this case, the *highest* and *lowest value* events correspond with the local maximum and minimum on the local level (occurring in state 3 and 7, respectively), but as discussed above, this is not necessarily true. For example, when another, shorter, path is chosen, $3 \to 4 \to 5$, the highest value is $plus$, in state 3 and 4, and the lowest value is $zero$ in state 5, while no local extremes are present in the path segment.

3.5.4.1 Ordering path (segment) level events.

When considering events of a particular type, happening to a particular subsystem along a path, there is usually a complete ordering possible which fully specifies their sequence in time. However, when multiple event types, or multiple subsystems are considered, this need not be the case. Because events can last for multiple states, it is possible for events to overlap. To allow merging of multiple histories (e.g., of different quantities, or different types of events) into one, a complete chronological ordering of events would be desirable. Therefore, the following rules were devised to order overlapping events. A schematic example is shown in figure 3.11, which shows four events, E1, E2, E3 and E4, ordered according to these rules.

1. IF event E1 starts before event E2, THEN order E1 before E2.
2. IF events E1 and E2 start synchronously, THEN order them based on which one ends first.

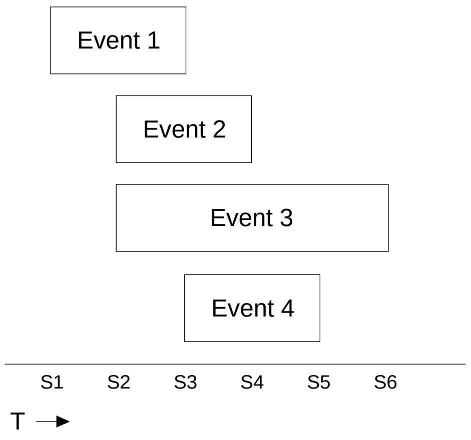


Figure 3.11
Four events on a time line, ordered according to the ordering rules described in the text.

These ordering rules focus on the beginning of the event. An alternative ordering could focus more on the end of events, and favour the one which ends first (in all cases, not just in the case of rule 2, as above). This would not make a difference for the relative ordering of event 1, 2 and 3, but it would place event 4 before event 3 (*cf.* figure 3.11). Instead of this alternative, the first option (focus on the beginning of the event) was chosen, because it seems natural in such cases to consider a large event (lasting more states) before a smaller event.

Cyclic paths. When a the state-transition graph contains a cycle, this means that the system modelled displays repetitive behaviour. Note that not all repetitive behaviour leads to a cycle in the state-transition graph, however. This only happens when the system as a whole goes back to an earlier state. When only a part of the system displays repetitive behaviour, while other parts do not, this may lead either to multiple smaller cycles, or no cycles at all, depending on the interaction between the subsystems.

Cyclic paths can pose problems with ordering events, in case the same transition occurs more than once in the same path. Therefore, cyclic paths are only permitted as long as they do not contain the same transition more than once. For example, in a simulation containing a cycle, the paths [1,2,3,4,1] and [1,2,3,4,1,3] are allowed, but not [1,2,3,4,1,2], because in the latter, the same transition $1 \rightarrow 2$

occurs twice. This restriction ensures that every cycle is passed at most once in a path, to avoid uninteresting repetition.

3.5.5 Global Level

As soon as there is more than one path in a simulation, a sequential view is insufficient, and a more global view is necessary to recognize the different alternatives. To convey the complexity of the behaviour in general, a graphical representation of all alternatives (such as the state-transition graph of a simulation provided by VisiGarp) can be useful. But when a user wants to know what happens in the simulation, with more focus on commonalities and differences between alternative behaviours, then the abstract graphical format does not suffice anymore. In a typical simulation, the number of states and branching complexity becomes too large to distinguish irrelevant details from the main developments. Therefore, a more natural way of communicating this information is necessary. Because of the linear nature of human language, people seem to be highly attuned to interpreting events in a linear fashion (*e.g.*, see Levelt [1982]). Therefore, the goal is to linearize the complex behaviour in a simulation as much as possible. A trivial solution is to present every path segment (between pairs of start, end, or branching points) in turn on its own, but when the number of path segments grows, this quickly becomes impractical. In many cases, however, there is potential for aggregation. This can reduce the number of states, transitions, and paths in a state-transition graph.

Three main methods of graph reduction can be distinguished: (1) abstracting from particular subsystems; (2) abstracting from temporal information; (3) abstracting from particular kinds of events. The first method, abstracting from particular subsystems, has been treated by Mallory and Porter [2000], as described in section 3.2. Although powerful, a limitation of this method is that it requires (user) specification of interests. Therefore, the remainder of this chapter focuses on the second and third kind of abstraction.

3.5.5.1 Abstracting from temporal information

Whenever multiple quantities are changing independently, this generates branching which, from a conceptual point of view, is often not very interesting. The simulator must consider every possible order in which the various quantities may change, leading to more complexity than necessary. As an example, consider the original state-transition graph for the CSH simulation, displayed in figure 3.12. If the different branches reunite later on, this indicates that the order of the changes involved does not matter for what happens later on. In these cases, we can perform *transitive reduction*, which reduces the number of transitions by selection, or *aggregation of alternative orderings*, which reduces the number of transitions and states by generalization.

There are two main principles behind these techniques: (1) to prefer linear descriptions of what happens, and abstract from alternative lines of events whenever possible; (2) to focus on begin and end of event sequences, if the intermediate stages are mostly continuous. A premise for applying these techniques is that it is acceptable to abstract from information about temporal ordering of events in case of multiple possibilities. In the following paragraphs, the specific techniques are described in more detail.

3.5.5.2 Transitive Reduction

Transitive reduction (as expressed by algorithm 1, without the second condition) is a well-known technique in graph theory [Battista et al., 1999]; it reduces the number of edges, while preserving

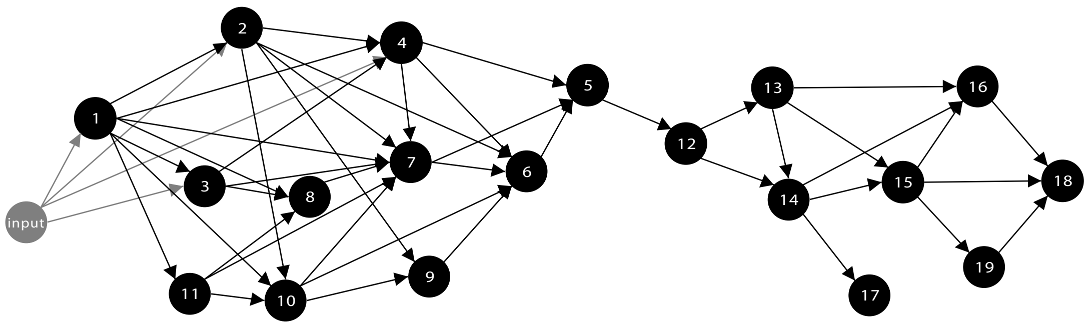


Figure 3.12
The original state-transition graph as output from the simulator, consisting of 19 states, 43 transitions, and 896 behaviour paths.

all information, provided that the edge-relationship is transitive. Intuitively, it works by removing all shortcut edges from a directed graph. Note that this does not affect which node can be reached from any given node.

In the case of qualitative state-transition graphs, the edges represent transitions which can also be considered transitive in some sense. For example, in figure 3.12, the information that state 7 can be reached directly from 3 can be abstracted, because 7 can also be reached via some other path (*e.g.*, $3 \rightarrow 4 \rightarrow 7$). There is an exception, however, when the events in the direct transition do not match the events in the longer path. In that case, the shortcut involves less, more, or different events than the longer path, and they should be considered alternative behaviours. Therefore, the second condition is added to the algorithm to ensure that we only abstract away transitions in which the same events occur as in the longer path.

Algorithm 1. Transitive reduction of the state-transition graph:

```
Abstract from (i.e., remove) all transitions T (= $S_x \rightarrow S_y$) for which holds:
1. There is a path P from $S_x$ to $S_y$ which does not contain transition T AND
2. P contains the same events as T.
```

Since this technique involves looking at transitions, the events that should be considered in the comparison of P and T are at the local level. All local events can be considered, or a subset of event types.[7] Some events (like local maxima) may exist only in the longer path because they involve two transitions; in such cases, the second condition in the algorithm does not hold. Note that using transitive reduction, only transitions are abstracted, not states. The only information that is lost after transitive reduction is the information about which events can occur simultaneously.

The result of performing transitive reduction on the original state-transition graph is visible in figure 3.13. It leads to a reduction from 43 transitions (which can be combined into 896 different paths) in the original graph to 25 transitions (resulting in only 24 paths).

The algorithm has been phrased in terms of removal of transitions, but when considering the remaining transitions instead, it is clear that this technique performs abstraction by *selection*. Although transitive reduction already greatly simplifies many state-transition graphs, it may not be sufficient for graphs with a large number of states, such as the one used in the example. Therefore, a new

[7] In the example shown in figure 3.13, the value and derivative events are considered. In section 3.5.5.5, abstraction from different kinds of events is treated in more detail.

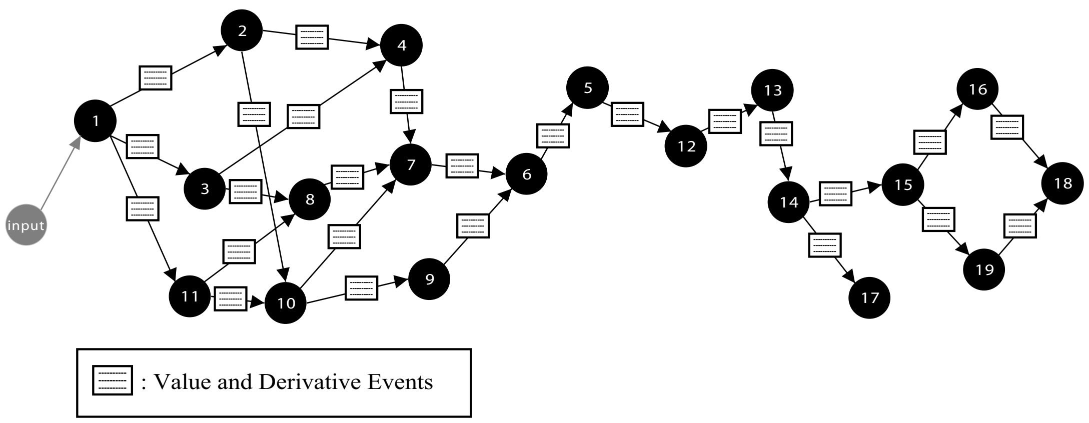


Figure 3.13
The result after identifying events and *transitive reduction*, consisting of 19 states, 25 transitions, and only 24 distinct behaviours.

technique has been devised, called *abstracting from alternative orderings*, which uses the method of *generalization*.

3.5.5.3 Aggregation of alternative orderings

The idea of aggregation of alternative orderings is to compare the sets of events in different branches which reunite again later, e.g., $15 \rightarrow 16 \rightarrow 18$ and $15 \rightarrow 19 \rightarrow 18$ in the original state-transition graph. If these different paths contain the same events (in a different order), or when the events can be aggregated to the same events, we can *generalize* over these alternative orderings, and see them as one *aggregated alternatives* transition, until the user is interested in more detail and the order becomes important again. The algorithm is presented here as algorithm 2.[8] The result is shown in figure 3.14.

Algorithm 2. Aggregation of alternative orderings in the state-transition graph:

Find a group of paths P_1 to P_n with the same begin-point (S_x) and end-point (S_y), for which holds: P_1 to P_n contain the same events, or events which can be abstracted into the same higher level events, and do the following:

1. Add a shortcut edge from S_x to S_y, to represent an aggregated transition, containing all (aggregated) events (of all, or selected event types) occurring in paths P_1 to P_n.
2. Delete every edge from the original paths P_1 to P_n, unless:
   (a) the edge appears *after* an *incoming* branching point, OR
   (b) the edge appears *before* an *outgoing* branching point.
3. Delete states which have no incoming and outgoing edges anymore.

[8]The idea behind this algorithm is similar to one of the behaviour aggregation techniques illustrated by Frantz [1996], but no algorithm is presented in his report.

Repeat this process (including step 1, 2 and 3) until no more alternative paths can be found.

It is assumed that the procedure responsible for finding groups of equivalent paths starts with the shortest paths, so that the abstraction is done bottom-up. The *unless*-conditions in step 2 of the algorithm are necessary to prevent deletion of an edge when this would also cut off other paths than the ones abstracted. Using this technique, both states and transitions are abstracted, thereby reducing the number of paths, or apparent ambiguities. It is important to note that no events are lost when applying this technique–only the information about the possible orders in which they may take place is discarded. For example, in figure 3.14, the aggregated transition box between state 1 and 5 contains the same events as the collection of events occurring along any path between state 1 and 5 in figure 3.13 or figure 3.12. Also note that a state may be preserved in one path, but abstracted in another. For example, state 14 in figure 3.14 is preserved for its inclusion in path $12 \rightarrow 14 \rightarrow 17$, but the information that you can reach state 18 from 14 in the original simulation is abstracted into the aggregated transition $12 \rightarrow 18$. This is a consequence of focusing on events rather than individual states.

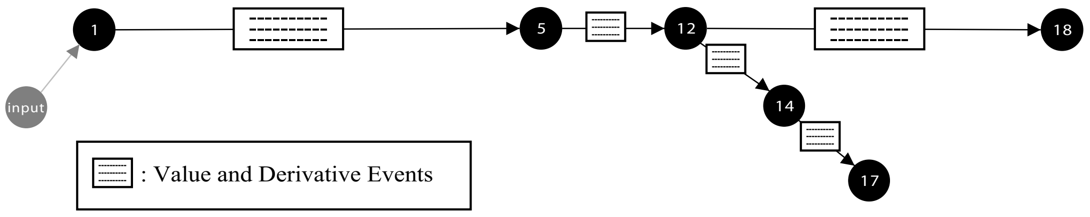


Figure 3.14
The result after aggregation of alternative orderings, based on value and derivative events. The graph now consists of 6 states, 5 transitions (of which 3 are aggregated), and only 2 distinct behaviour paths.

3.5.5.4 Aggregation of sequence

In this step of the aggregation process, path segments (sequences of states without branching points) are investigated for the presence of value and derivative events on the path (segment) level, as defined in section 3.5.4. If present, they replace the underlying local level events. When this is done for all path segments, each path segment is checked for the presence of any remaining local level events. If these are absent, the whole path segment can be chunked into one aggregated sequence transition. This technique further reduces the number of states and transitions, but not the number of paths.

In the running example, the current situation, as displayed in figure 3.14, is as follows. The transition $1 \rightarrow 5$ contains many value events, but the rest of the transitions ($5 \rightarrow 12$, $12 \rightarrow 14$, and $14 \rightarrow 17$) contain only a few. Below are shown only the events for selected quantities:

```
States  Events
-----------------------------------------------------------------
1 -> 5: cover1      increases: zero             -> low
      : number_of1  increases: zero             -> low
      : number_of2  increases: zero             -> low
      : number_of3  decreases: max              -> high
      : ...

5 ->12: cover1      increases: low              -> medium
```

```
      : number_of1  increases: low             -> medium
      : number_of2  increases: low             -> medium
      : number_of3  decreases: high            -> medium

12->14: cover1      increases: medium          -> high
      : number_of1  increases: medium          -> high
      : number_of2  increases: medium          -> high
      : number_of3  decreases: medium          -> low

14->17: number_of2  increases: high            -> max
```

Because the path segments $1 \rightarrow 5 \rightarrow 12$ and $12 \rightarrow 14 \rightarrow 17$ both contain value events which can be chunked together into path segment events, the result is as follows:

```
States  Events
---------------------------------------------------------------
1 ->12: cover1      increases: zero            -> medium
      : number_of1  increases: zero            -> medium
      : number_of2  increases: zero            -> medium
      : number_of3  decreases: max             -> medium
      : ...

12->17: cover1      increases: medium          -> high
      : number_of1  increases: medium          -> high
      : number_of2  increases: medium          -> max
      : number_of3  decreases: medium          -> low
```

When in the remaining graph a path is considered on its own (e.g., $1 \rightarrow 12 \rightarrow 17$, even further chunking into a single transition is possible, but this is not desirable when the goal is to overview the possible alternatives. Hence, no chunking is applied over the boundary between path segments: state 12, which is a branching point.

3.5.5.5 Aggregation based on particular kinds of events

As discussed above, the recognition of events plays a key role, as these are the basis for determining which paths could be considered equivalent, when abstracting from the temporal aspects. In the example, only the value and derivative events were considered. However, each of the various types of events (Values/Derivatives, Derivatives only, (In)equalities, Dependencies, E-R structure and Quantities, Model fragments, and Causal effects) can be selected or left out of the aggregation process. This way, the role of the various modelling primitives in the complexity of the simulation can be investigated.

Comparing figure 3.16 with figure 3.15 shows for the example simulation how selection of different combinations of event types results in different amounts of abstraction. For example, the maximum amount of abstraction is reached when selecting only value and derivative events, or causal events, or the E-R structure and Q events (see figure 3.15). For the last case, the reason is trivial: no structural events take place during this simulation. For the causal effects, the result is the same, but not because nothing happens. Many causal effects events occur, but between state 1 and 5, all paths contain the same events, and the same holds for all paths between 12 and 18, and all paths between 12 and 14.

When aggregating over model fragment events only, some more transitions remain (see figure 3.16). For example, state 15 and 16 are preserved, because they differ with respect to the presence of the model fragment *open cerrado*, which is present in state 15, but not in 16. Similar differences occur in the states that are located between state 1 and 5, because the model fragment *campo sujo with no tree* is only present in state 6, and *campo limpo with less grass* only in state 7, while neither of the two model fragments is present in state 4.

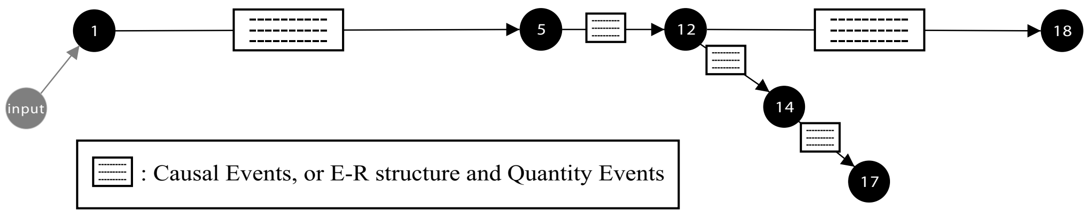


Figure 3.15
The result after aggregation of alternative orderings, based on causal events, or based on E-R structure and quantity events. The result is the same as for the value and derivative events, shown in figure 3.14.

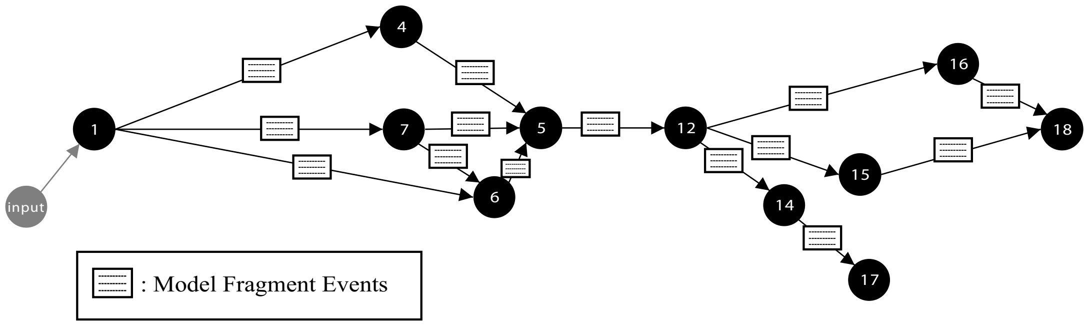


Figure 3.16
The result after aggregation of alternative orderings, based on model fragment events.

3.5.5.6 Abstraction from details to allow comparisons

To allow a meaningful comparison between two behaviours in terms of events, it is necessary to abstract from certain details in the event representations as defined in sections 3.5.1 to 3.5.4. The representation of an event includes details about the circumstances, such as the path, or transition in which it occurs. Therefore, the time-frame information in the event representation is abstracted when a match is made between two events occurring in different paths. Furthermore, the representation of events includes the derivatives and values which also precludes direct matching. In many cases, a more conceptual matching is desired, based on the type of event, but not all details involved. Therefore, the following rules are defined for abstracting from values in two value events of the same kind.

- For events which involve a derivative change, the values involved are abstracted from, even if they are not (semi-)identical. This is reasonable, because the focus is on the derivative change, and not the value.
- For events which involve only a value change, the values are important, so the events are only matched when the values are (semi-)identical.

- Values are considered semi-identical when they have the same basic name. For example, *max(number_of1)* and *max(number_of2)* are considered identical for the purpose of event abstraction, although it is possible that they are mathematically unequal or do not correspond.

3.6 Numerical Results

In this section, numerical data are presented which show the impact of the aggregation techniques on the complexity of a simulation. In table 3.7, the data are shown for the aggregation processes applied to the CSH simulation.

					Facts					
Level	St.	Tr.	Sg.	P.	V/D	CM	MM	ERQ	MF	Total
0	19	47	41	869	608	846	1028	1368	950	4800
1	19	47	41	869	608	582	1028	1368	950	4536
2-5					Events					
2	19	47	41	869	553	455	621	288	108(631)	2025
3-5 (TR)	19	26	16	24	157	123	-	72	-	-
3-5 (AO)	6–11	6–14	3–7	2–9	100	87	180	72	41(284)	480

Table 3.7
Numerical results for the aggregation techniques applied to the CSH simulation, showing the number of states (St.), the number of transitions (Tr.), the number of path segments (Sg.), and the number of paths (P.) on the different levels of aggregation, as well as the number of facts and events in the categories Value and derivatives (V/D), Causal Model dependencies (CM), Mathematical Model dependencies (MM), Entities, Attributes, Structural Relations and Quantities (ERQ), and Model Fragments (MF).

The first four columns stand for the number of states (St.), the number of transitions[9] (Tr.), the number of path segments (Sg.), and the number of paths (P.) in the simulation. For the levels 1 and 2, the number of facts is shown for every category: Value and derivatives (V/D), Causal Model dependencies (CM), Mathematical Model dependencies (MM), Entities, Attributes, Structural Relations and Quantities (ERQ), and Model Fragments (MF). For the levels 3-5, the number of events is shown for the same categories. These results are determined as follows.

At level 0, the number of Garp facts is counted in all states. For example, there are 72 ERQ facts in each state, times 19 is 1368. In total, there are 4800 facts represented in the simulation. This certainly does not mean there are 4800 *different* facts; how much distinguishing information is encoded in these 4800 fact specifications is unknown at this stage. In order to distill the most important information from this data, the aggregation techniques are applied in the following manner.

At level 1, the number of causal dependencies can be reduced by leaving out the submissive dependencies (210) and inactive dependencies (54). This reduces the total number of facts to 4536.

At level 2, the local events are recognized for the original state-transition graph, leading to 2025 events in total. There are 553 value and derivative events, 455 causal model events (which include the submissive and inactive dependencies again), 621 mathematical model events, 288 structural events, and 108 model fragment events (as discussed in section 3.5.2.6, the model fragment events address

[9]The transitions from the scenario input state 0 to the start states are included in this number, as these may include different events as well.

groups of model fragments, which involve 631 applications of model fragments). This means a reduction by more than a factor of two with respect to the number of facts. This is clearly seen in the number of structural events. Because there are four start states, in each of which all 72 structural facts are introduced, there are 288 ERQ events recognized. After that, no structural events take place.

At level 3–5, first, transitive reduction (TR) is applied, taking into account only the value and derivative events. This reduces the number of transitions, path segments and paths significantly. Because the number of events is determined by summing them up per path segment, this leads to a further reduction by a factor of four, approximately. At this stage, the system recognizes that there are essentially 157 value and derivative events, and only 72 ERQ events.

Finally, aggregation of alternative orderings (AO) is performed, which leads to an even greater reduction in the number of states as well as in the number of transitions, path segments and paths. These results are displayed as ranges, because they vary depending on which event types are considered in the aggregation process (see section 3.5.5.3). For the categories V/D, CM, and ERQ, the reduction is maximum, leading to 6 states and 2 paths, while for MM there are 7 states and 5 paths, and for MF, there are 11 states and 9 paths remaining. The number of events for these categories takes this into account, leading to a total number of 480 events. This means that this simulation can be totally explained by mentioning 480 events, which contain even more information (about the status of causal dependencies and the appearing, disappearing and changing of other kinds of facts) than included in the 4800 facts. 480 events is still a lot of information to present, but it is nevertheless a reduction by a factor of ten. Furthermore, as described in section 2.5.1, the CSH simulation is not a toy example but a complex simulation in a realistic domain.

Applying the aggregation techniques to simulations in other domains leads to maximum reduction factors (in terms of number of facts to events) of four (for the container-piston simulation depicted in figure 2.8) to twenty-one (for a simulation involving the inelastic collision of four balls). This indicates that the amount of reduction achieved in the CSH simulation is not unique, and that for some simulations, even higher reduction factors are possible. As the techniques were tested on these simulations while being implemented, evaluation of the techniques will continue on simulations models unknown to the author, to ensure that the techniques are not geared too much to specific simulations.

3.7 Discussion

This section compares the approach taken in this thesis with some of the existing literature on aggregation and points out directions for further research.

Abstracting before or after a simulation. The aggregation methods presented in this chapter work on a simulation that is treated as a given. That makes it different from the abstraction methods, such as compositional modeling [Falkenhainer and Forbus, 1991], automated modeling [Rickel and Porter, 1997], and automatic qualitative abstraction [Sachenbacher and Struss, 2001], which try to simplify a model beforehand, given a set of questions to be answered by the model. However, simplifying the model beforehand is often not possible. When the model builder is not the same person as the user studying it, it may be unclear to the user which kinds of questions the model is able to answer. In addition, it may be hard for a user to formulate queries in advance, especially for learners. In both cases, simplifying the model beforehand is impossible, and other ways are needed to determine what is interesting and what is not. The methods discussed in this chapter are designed to support making this distinction.

Specifying user interests. To what extent should it be possible or necessary for a user to specify a focus for the aggregation process? The work by Mallory and Porter [2000] requires that the user specifies quantities of interest, so that the state-transition graph can be abstracted based on the value and derivative events for those quantities. In the approach used in this chapter, the user does not have to specify quantities of interest before the aggregation process starts. The techniques proposed here also extend the work by Mallory and Porter, in the sense that more kinds of events are distinguished than just value and derivative events, and aggregation is done on higher levels, also including branching and global views. The user can now investigate the system's behaviour on each of these levels of aggregation. These extensions make the current approach more generic and flexible than previous work. Still, specifying a focus is a powerful method of selection, which might even be necessary considering that the end result in the example simulation contains hundreds of events to present. However, such a focus should not only concern quantities of interest, but also include a particular time frame, and/or particular types of events. This issue will reappear in chapter 4.

Is an aggregated state-transition graph still a state-transition graph? When reducing the state-transition graph using the techniques described in this chapter, the question may arise whether the result is still a state-transition graph in the original sense. The answer to this question is yes, if this is interpreted to mean that states still refer to states of the system, and transitions to changes which occur between states. The aggregation techniques did not change information within the states, after all, and the aggregated transitions contain precisely the specification of the differences between the states. However, it is not a state-transition graph anymore in the sense that it includes *all* qualitative states and transitions, since some of them may be abstracted away into the aggregated transitions. But the claim is that important states are not abstracted away. An abstracted state can not have been important *on its own*, because (1) there must have been a transition which bypasses it, in which the same happens as in paths including the state, or, (2) there must have been similar states that have also been abstracted, which differ only in the order in which things have changed. A *collection* of abstracted states can not have been important either, because this would mean that the transitions between them are considered important as well (otherwise, each of the states could be considered on its own). Because all information in the transitions is kept in the abstraction except the temporal ordering, this would imply that the premise that the temporal ordering of events can be neglected is violated.

In short, this technique is valuable when it is accepted that it is not the individual states that matter, but rather the events that take place between them. Since the events are preserved, the reduction of the number of states and transitions is considered an advantage. There is an issue, however, when other kinds of events are considered than value, derivative, and (in)equality events (the things specified in a traditional state-transition graph). Using other kinds of events, such as causal events, structural events, and model fragment events changes the nature of the transitions slightly, compared to the traditional notion of state-transition graph. Nevertheless, these events also denote differences between states and therefore, this issue should not be regarded as a problem, but rather as a feature which increases the flexibility of inspecting a simulation.

3.7.1 Further research

The aggregation techniques described in this chapter use only the information contained in the simulation itself. As the goal was to develop generic techniques, this is an important advantage. However, in some cases, there is potential for further aggregation when a little information is added.

Aggregation for decision support. With respect to aggregation on the global level, the work Eisenack and Petschel-Held [2002] is relevant, because they also perform graph-theoretical analysis on state-transition graphs to simplify communication of simulation results. Their approach puts emphasis on branching points of which at least one edge leads to a *locked set*: a set of states which cannot be left again once entered. Because these branching points are on the verge of irreversible changes, they play an important role in management simulations. By adding a categorization of parts of the state-transition graph as either desirable or non-desirable, this method allows automated decision support for managers in domains such as sustainable fishery and agriculture [Eisenack and Petschel-Held, 2002].

Finer recognition of model fragment events. Regarding the recognition of model fragment events, there is also potential for further aggregation. Such an event includes a list of model fragments that is currently based on which entities the model fragments apply to. Although this grouping is better than no grouping at all, in some cases the lists of model fragments involved can get long, which is not desirable. A more refined aggregation technique could make use of the is-a or applies-to hierarchy of model fragments to determine which model fragments are in fact closely related, and therefore candidates for recognizing a replacement of model fragments. For example, this way a definition of the replacement event could require that a model fragment can only be replaced by a brother concept in the model fragment is-a hierarchy. How strict such a definition should be exactly to be of practical use remains to be tested (*i.e.*, should they both be either ’processes’ or ’situations’, or should both model fragments be at the exact same level in the hierarchy), as these options have not been implemented.

Aggregation for explanation vs. diagnosis. The work presented here on the recognition of causal events builds on research by Mallory and Porter [2000], and de Koning et al. [2000], and alleviates some of its limitations, by specifying a well-defined set of dependency status categories for the GARP simulation framework. Although the work described in this thesis is geared towards explanation generation, it is also interesting to consider its value for the task of diagnosis of a learner’s reasoning process. In principle, the ideas presented here could well be integrated in the General Diagnostic Engine employed by de Koning et al. [2000]. This would extend the scope of predictions to be diagnosed from single transitions to longer paths in the simulation. Because model fragment events are now also included in the aggregation mechanisms, it would also be possible to trace a missing reasoning step to (unknown conditions for/applying the results of) a model fragment.

The main purpose of the aggregation mechanisms described in this chapter is explanation generation. But in order to generate explanations, other issues come into play besides aggregation, such as didactic goals, styles, and plans. These include information about what should be included in an explanation, who has the initiative in the interaction, and so on. These issues are dealt with in chapter 4, on interactive explanation.

3.8 Conclusion

To fulfil the explanatory potential of qualitative simulations, an explanation generation system needs to be equipped with methods for simplifying the contents of qualitative simulations. This is necessary because presenting all information and counting on the user's exploration capabilities to make sense of it all (as described in chapter 2) does not suffice. This chapter has provided an overview of existing techniques for simplifying qualitative models and simulations, and has introduced several new techniques, specifically aimed at aggregating qualitative simulations. The techniques added are based on combinations of the generic methods selection, chunking, generalization, and grouping. Using these techniques, data from the original simulation results are interpreted as facts and events of different categories on five levels of aggregation. On the system state level, which describes a system at a particular time point or interval, causal effects are classified. On the local event level, meaningful differences between adjacent states are recognized as events. The path segment level includes sequences of events which denote behaviour that *necessarily* takes place. On the path level, branching points and cycles may occur; hence this level specifies behaviour which is *possible*. The global level includes alternative behaviours to provide an overview of what may happen to the system. The techniques of transitive reduction and aggregation of alternative orderings are presented, which reduce the amount of states and transitions in a state-transition graph by abstracting from temporal information, while keeping the essential information about what happens. As a result, the complexity of a simulation is significantly reduced, both in terms of the number of states, and the amount of information within states that needs to be considered, which makes it easier to communicate effectively.

Further research might address using aggregation for integrated decision support and diagnosis of learner behaviour. The following chapter investigates how the results of the simulation after aggregation can be used in an explanatory interaction.

4

Interactive Explanation

“Information consists of *differences that make a difference*”

Edward R. Tufte, US expert on information design, in his book entitled “Envisioning Information” (1990)

4.1 Introduction

This chapter discusses the generation of interactive explanations based on qualitative simulations, utilizing the results of the visualization and aggregation methods discussed in chapter 2 and 3.

In explanation generation, a distinction is often made between what to say, and how to say it [Winkels, 1992]. The former of these issues corresponds to didactic goals [Bloom, 1956]; the latter relates to didactic means and style. *Didactic goals* specify the material that should be mastered, including specific facts as well as generic rules. For instance, a learner may be required to predict what happens to a particular system given an initial situation description, or to explain how a particular state of the system came about. *Didactic means* are the means of communication used to convey information related to the didactic goals. The graphical representations discussed in chapter 2 are an important example of didactic means, but this chapter also investigates textual means such as explanations and exercise questions. *Didactic style* is concerned with how to utilize didactic means in interaction with a learner. Didactic styles can differ in several aspects. Who has the initiative: the system or the learner? When multiple things are to be communicated, how to order them?

Combining a set of didactic goals with a didactic style leads to a didactic plan, which directs the system to fulfil the didactic goals using selected didactic means. When the user is given some of the initiative, the system should offer the user facilities to choose topics and ask questions about them. When the system has the initiative, it should not only present the information, but also make an effort to check whether the information it presents is adequately understood. The users of intelligent learning environments are students, who, at first, do not have much knowledge about either the domain, or qualitative simulations. Because of this, students may not be aware of the questions that the learning environment can answer, or even of the questions they want to ask. Therefore, it is important to make the system flexible so that it can show information in an easily accessible way for novice students, as well as answer more specific questions from more experienced students, or other users. The research questions addressed in this chapter are the following.

- What are the didactic goals that an interactive learning environment based on qualitative simulations should support?
- Which didactic means should be used to communicate the information relevant for a didactic goal, and how can they be generated automatically?
- How can different didactic styles be supported in a qualitative simulation-based learning environment?

The chapter is structured as follows. In section 4.2, the existing literature on interactive explanation generation (for qualitative simulation) is reviewed. From there on, the chapter focuses on a particular approach which leads to the implementation of an interactive explanation generation system, called WiziGarp. Section 4.3 outlines the conceptual design, including a discussion of didactic goals, means, and styles. In section 4.4, the technical design of WiziGarp is presented, with details about internal mechanisms and the user interface. The work presented is discussed in relation to previous work in section 4.5, and section 4.6 concludes this chapter.

4.2 Review of the Literature on Explanation Generation

This section describes existing research on explanation generation, with a particular focus on the use of (qualitative) computer simulations to generate explanations.

The work by Acker et al. [1991] uses a large knowledge base to generate explanations of various kinds. Their system is not capable of simulation, but the knowledge base includes information about static as well as dynamic aspects of the domain (of botany). They developed a hierachy of question types, and distinguish view types (unifying viewpoints, as defined earlier by McKeown [1985], among others), specifying necessary relations and permissible relations to be included in an explanation of that view type: class-dependent (explaining a concept in terms of a class hierarchy); structural (explaining the physical or temporal structure of an object or process); functional (explaining the role of an object in a process); modulatory (explaining how one object or process affects another); attributional (specifying the properties of a concept); comparative (comparing or contrasting two concepts).

The work by Gautier and Gruber [1993] and Forbus et al. [1999] includes explanation generation based on simulations. But instead of using qualitative simulations, they use a numerical simulator. Nevertheless, both approaches incorporate QR techniques, such as compositional modelling. Model fragments are used to specify individual components and processes, which are employed to generating qualitative explanations. To give an example from Gautier and Gruber's work, when a component changes from one mode to another, because some quantity reaches a threshold value, explanations can be given to answer questions such as 'what caused this change in behavior?' (see Gautier and Gruber [1993], p. 91). A graphical interface is available which allows clicking on an icon, word, phrase, or sentence. The system produces a menu of possible queries, from which the user can select one to be answered by the system. When an explanation is presented, the user can ask follow-up questions about any part of the explanation. Because no explicit representation of causal relationships is present in their simulation framework, this information is derived based on causal ordering, a technique which determines which quantities are affecting each other by analyzing how they occur in the mathematical formulae [Iwasaki and Simon, 1986]. To arrive at explanations which are interesting, but not too lengthy, Gautier and Gruber make a selection of causal influences from the subgraph of influences, using two salience heuristics: collapsing equality chains, and collapsing paths of the same dimension. The first heuristic uses the transitive nature of the equality relationship to transform a chain of equality statements $X1 = X2 = ... = Xn$ to just one: $X1 = Xn$. The second heuristic specifies that a sequence of variables of the same physical dimension (*i.e.*, *pressure*) is presumed to be driven by the same influence, and can therefore be collapsed into a single-step influence (*e.g.*, heating a container may result in the rise of the pressure in a series of connected components). The influences which are collapsed in the remaining graph can still be shown on request. From the perspective of generating explanations for qualitative simulations, a limitation of this work is that no alternative behaviours can be explained, as there are no branches in a quantitative simulation.

The work by Forbus et al. [1999] is similar in the sense that it also includes explanations to user queries, which are presented once the user clicks on a system-derived statement. This work has resulted in a fielded system, called CyclePad, which supports students learning about thermodynamic cycles. In the interface, a domain-specific visualization is presented of the simulated system. CyclePad performs routine calculations, presents graphs to facilitate sensitivity analysis, keeps track of modelling assumptions, and coaches the student by showing contradictions leading to impossible designs, and general advice on designing and analyzing cycles. CyclePad also provides explanations in response to user queries, such as ’Why is *stuff* made of *substance*?’. CyclePad has been adopted by instructors in various countries for both introductory and advanced courses in thermodynamics since 1995. Using CyclePad, students are able to gain understanding by experimenting with designs in ways which go beyond traditional textbook-type problems. The limitations of this work include the domain-specificity of the interface, and the fact that CyclePad only allows steady-state analysis.

The work by Mallory [1998] on the EXPOUND system explicitly addresses explanation generation based on qualitative simulations in the QSIM framework [Kuipers, 1994]. It is centered around the idea that explanation needs a particular focus, defined as a set of quantities of interest. Based on this focus, the state-transition graph is simplified and presented as a diagram, and causal explanations are presented in textual form which describe the behaviour of the quantities of interest in terms of *events*. Preliminary evaluation with EXPOUND indicates that the subjects (QR experts) found the abstracted behaviour graphs especially useful, and the textual explanations to a lesser extent. As already discussed in section 3.2, limitations of this work include the fact that only a limited set of event types is considered, the requirement that an expert user has to specify his or her interests, and the fact that branches in the simulation are not dealt with. An additional limitation is that the generated diagrams are not integrated in the interface of the system; in the evaluation of EXPOUND, they were presented to the subjects on paper.

The work by Salles et al. [1997] is based on qualitative simulations in the GARP framework, and focuses on the domain of Brazilian Cerrado ecology. Relevant domain concepts are represented as model fragments, which are an important ingredient for deriving explanations. Four types of explanation primitives are distinguished: (1) basic concepts (*i.e.*, entities, their attributes and relations between them), (2) quantities (with their value, quantity space and derivative), (3) causal dependencies between quantities, and (4) mathematical constraints between quantities and/or values. A set of question types is defined, most of which focus on a particular state of the simulation [Salles, 1997, p. 221]:

1. What are the objects, quantities, quantity values, and quantity relations involved in a particular model fragment or state of the simulation?
2. What are the conditions for a particular view or process to become active?
3. What are the initial causes of change in the current state?
4. How does change propagate to other quantities in the present state?
5. How does a particular quantity change over states?
6. How can a particular view (or process) be compared with another view (or process)?
7. How can a particular state of the simulation be compared with another state?

Each question type maps to the explanation primitives, or a combination thereof. The relevant information is determined based on the selected topic and question type. Templates are presented which include natural language expressions for the GARP facts to be expressed. However, they mention that the state of their prototype does not allow for the full use of the explanatory possibilities. Further limitations of this work are the primary focus on single states, and the inflexibility of the answer generation, due to the direct mapping from simulation primitives to explanation content.

The work by Rühl [1997] also addresses explanation generation based on the GARP framework, in the context of the STAR project [de Koning, 1997, de Koning et al., 2000]. The resulting prototype, called STAR^{light}, teaches about a domain in physics: two leaking containers on a balance. The interaction starts with a prediction question, such as 'What will be the difference between volume left and volume right in the next state?'. Based on the user's answer (multiple choice), the system either continues to the next prediction question, or goes into diagnosis mode by generating probing questions to find out what the user does not know. Explanations are generated as hints or as remedial feedback after the user's reasoning error is diagnosed. An explanation focuses on a particular reasoning step which is necessary to predict what happens in the transition to the next qualitative state, or an intermediate result within the current state. An example of such a reasoning step is determining the derivative of a quantity Q_y based on the value of another quantity Q_x, given an influence dependency $Q_x \xrightarrow{I+} Q_y$; other reasoning steps deal with other types of dependencies, value and derivative determinations, and value/(in)equality terminations. Evaluation of the explanation generation capabilities of STAR^{light} indicated that especially advanced subjects (who were familiar with structured behaviour analysis) were positive about the explanatory interaction. Novice subjects were less satisfied, and reported problems with the specific, one-component based explanations. Although the authors suggest that the domain used may not be suitable for novices, they note that a more advanced question and explanation generator is required to communicate with novices [de Koning et al., 2000]. Therefore, the main limitation of this work is the lack of support for novice users. Furthermore, some preprocessing and adjustment to the interface is necessary to adapt STAR^{light} to a new domain.

Conclusions from the literature review. Qualitative reasoning and simulation are a good basis for generating explanations, as they contain explicit representations of the kinds of information required to answer explanatory questions. Several types of questions and topics are distinguished in the literature, which relate to entities, quantities, states, model fragments, and causal and mathematical relationships. Much of the existing work focuses on the behaviour in a single state, although the characterization of *events* and comparing states are important to explain dynamic behaviour. Most work does not handle explanation of alternative behaviours. Several projects have led to successful applications, but many of these are aimed at users with QR expertise, without much support for novices. Some applications are domain-specific, or lack an integrated graphical user interface.

4.3 Conceptual Design of WiziGarp

Using the outcomes of the literature review, the functionality of VisiGarp is extended to include more interactive explanation capabilities, which make it deserve a name change: VisiGarp is transformed into WiziGarp.[1] WiziGarp is intended to support students who may be novices in their domain, or have no experience with QR, although it should still be useful for expert users too. Section 4.3.1 outlines the scope of didactic goals which is addressed by the WiziGarp system. In section 4.3.2, the didactic means are discussed which may be used to fulfil the didactic goals. Section 4.3.3 presents the kinds of didactic styles which may be adopted in interaction with a user.

[1]The name change reflects the move from pure visualization (VisiGarp) to wisdom, wishes, and wizardry (WiziGarp), while still echoing the original idea. Wisdom refers to the ability to ask and answer the right questions (the queries, tutoring questions, and textual explanations), wishes to the desire to plan the future (the didactic plans), and wizardry to the art of making things smaller or disappear (the aggregation mechanisms).

4.3.1 Didactic Goals

This section outlines the didactic goals which an interactive learning environment based on qualitative simulation should support. Didactic goals represent the things that students should learn. They are divided in three main categories: terminology, generic domain knowledge, and simulated behaviour, which are discussed below.

Terminology. First, the most basic knowledge concerns the meaning of the terminology used. This includes the terminology of GARP, and the concepts used in the aggregation, visualization and interface of WiziGarp. Knowing how to interpret the terminology is indispensable for making sense of the other didactic goals, as it provides the 'vocabulary' and 'grammar' to talk about models and behaviour. The assumption is that students will at first have only a limited and implicit understanding of the terminology, but will nevertheless be able to use it, similar to the way children use verbs and nouns before they learn about verbs and nouns as grammatical concepts. The students' understanding will gradually become more explicit, and at some point, they will realize that the distinction between concepts in the terminology is useful for learning about new domains, just like it is useful to know whether an unknown word is a verb or a noun. There is evidence that (a subset of) qualitative reasoning terminology is useful in educational settings, even for middle-school students [Forbus et al., 2004].

Generic domain knowledge. Second, the generic domain knowledge involves the knowledge about the domain which can be specified without referring to a specific situation. The student must learn the domain vocabulary, which may include unfamiliar words, or new meanings of familiar words in a particular domain (e.g., *what does immigration mean?*, in the domain of ecology). This is important as it grounds the knowledge into real-world concepts. Although it is not easy to accomplish this grounding in practice, supporting understanding of the domain vocabulary is not the main focus of this thesis, because it is highly domain-specific, and does not deal with the dynamics of system behaviour per se.

Building on an understanding of the domain vocabulary, and the terminology described above, generic domain knowledge includes knowing the is-a hierarchy of entity types, and knowing all about model fragments: their contents, and the is-a and applies-to hierarchies. This provides an understanding of which kinds of processes and situations may occur in a given domain, under which circumstances (conditions) and which additional facts they would introduce (results). When the domain model library is large, didactic goals may be specified addressing only part of the model fragments. The generic domain knowledge might be a goal on its own, but it is also a necessary requirement for reasoning about the simulated behaviour.

Simulated behaviour. The third category of didactic goals deals with understanding the behaviour of a system as it is being simulated. This falls apart in two: (1) recognizing the behaviour, and (2) reasoning about the behaviour.

Recognizing behaviour includes knowing which facts are true, and which events occur in a simulation.The latter requires comparing states, in particular across transitions. It can also be useful to compare facts across states which may not necessarily be in sequence, i.e., any two states. When alternative behaviours are considered, there is a need to distinguish between facts and/or events across different paths. Besides differences across time-frames, it is also useful to learn how to compare subsystems, in terms of facts within a state, and events within a transition or path (segment). Because a GARP qualitative simulation includes explicit representations of causal and mathematical dependencies, organized in model fragments, recognizing the behaviour of a system should support students in developing an explicit mental model of the simulated system.

Reasoning about behaviour becomes important when the simulation results are not taken for granted, but more insight is required about where the simulation results come from. It includes determining results of dependencies within a state, as well as predicting and explaining behaviour. Determining and predicting amounts to forward reasoning, while explanation requires reasoning backwards.

Within a state, several didactic goals can be considered, requiring values, derivatives, effects, or (in)equalities to be determined. First, the effects of causal dependencies and correspondences need to be determined. For causal dependencies, this involves determining which are active, and if so, what their effect is, given the sign of the influencing quantity value/derivative and the sign of the dependency itself. For correspondences, it is necessary to check whether they are active, and in which direction values may be propagated. Second, dependencies may involve mathematical formulae involving addition and substraction. For each of these cases, the result needs to be determined. Third, given that all effects and formula results are known, one can determine a quantity's value and/or derivative based on the analysis of the dependencies involved. Finally, (in)equalities may be determined between quantity values or derivatives.

Prediction is an important skill, which requires having a 'runnable' model of behaviour. It involves knowing facts in a begin state, and reasoning from there to future states, by identifying which information is relevant, determining results of dependencies (as discussed above), updating a state, determine occurrence of events and the effects of these events in terms of other events.

Explanation is another important skill, which is almost the mirror-image of prediction. Explaining a fact in a particular state requires relating it to other facts or events which causally *explain* or *trigger* the fact. Of course, these facts must hold (or events must happen) in the same state, or before, in a path from a begin state to the state in question. A *full explanation* states all causes, while a *trigger explanation* only states the cause which has become true most recently, and hence triggers the fact to become true. Explaining an event in a transition works basically the same; events are explained by other events, or by facts which were already true in a begin state, or have been assumed somewhere along the path.

4.3.1.1 The Need for Focus: What's in a Topic?

When the list of didactic goals is instantiated with information from a specific simulation it can grow very large, depending on the number of elements and relationships involved in the model. Addressing all of the possible didactic goals is impractical, if not impossible in a realistic educational setting, with limited resources such as time and attention span of students. The aggregation techniques discussed in chapter 3 offer ways to reduce the number of behaviours to consider, but still, the need for focus is paramount. For this reason, it must be possible to divide the information in a simulation into topics, which can be selected and treated in order of importance, or preference. A topic defines a portion of the information in a simulation, which can be used to instantiate a particular didactic goal. A topic addresses a particular part (or parts) of the system, at a particular moment or interval, or different moments, or intervals of time, for certain ontological kinds of information. In short, a topic is defined here as a three-tuple: (1) system scope, (2) time frame, and (3) information type. Each of these three elements is discussed in detail below.

System Scope. The system scope defines which part(s) of the system one is interested in. If there are many entities or quantities in a simulation, narrowing down the system scope is an efficient way to reduce complexity. The system scope may be defined as a single subsystem or multiple subsystems, depending on whether the communicative goal addresses one subsystem, or the comparison between multiple subsystems. Note that a single subsystem may include multiple entities and/or quantities.

Time Frame. The second way to focus is to look at information in a particular time frame. The most elementary time frame is the qualitative state, but also multiple states and even multiple paths can be taken as the time frame of interest, to compare behavioural states, or behavioural paths.

Information Type. The third way of focussing is to include only certain types of information, while excluding others. For example, it is quite common to look at the values and derivatives of quantities before going into causal or mathematical dependencies. Some ontological types of information, such as model fragments, are complex to understand for novices, but they may be especially interesting for experts and model builders. Thus, the selection of types of information allows topics to be specified for different levels of expertise.

4.3.2 Didactic Means of Communication

The WiziGarp interactive learning environment realizes the didactic goals by using interactive visualizations and textual dialogue as means of communication.

4.3.2.1 Interactive Visualizations

The main method of communicating information in WiziGarp is through diagrammatic visualization, as it was in VisiGarp. Summarizing from chapter 2, there are four reasons for this. (1) The diagrams show different types of information elements as different types of graphical elements, which makes it easy to recognize them as such. (2) The diagrams map multiple references to a specific information element onto a unique graphical element, which makes it easy to find associated information elements. (3) The diagrams make use of graphical space to communicate meaning, using inclusion, indentation, and horizontal and vertical ordering as graphical means. This makes it easier to communicate information which is hard to communicate in a textual format. (4) The diagrams provide a means for focusing the attention and structuring the interaction, by providing ways to highlight or hide parts of the visualization, and by providing hyper-links to associated information. This is what makes the diagrams *interactive visualizations*.

Like VisiGarp, WiziGarp contains library views which show the generic domain knowledge and simulation views which show the information from an actual simulation, as well as controls for the simulation, and options for navigation and manipulating the display. The main screen has been reorganized to show more types of information at the same time, and to allow easy switching between (consecutive) states. In addition, two new views are added: the global view, and the event history view. The new views, and other improvements are discussed in more detail later in this chapter.

4.3.2.2 Textual Dialogue

The second method of communication employed in WiziGarp is textual dialogue. For the aims of this work, textual dialogue is decomposed into the following elements: textual descriptions, queries, causal explanations, contrastive explanations, exercise questions, and canned texts.

Textual Descriptions. Textual descriptions are used to communicate facts and events in the simulation. State-based facts are also visualized using graphical means, but textual descriptions are added for two reasons: Textual descriptions of facts alongside a diagram with the same information displayed graphically are considered useful (1) to assist users in understanding what the diagram means (especially in the beginner stadium), and (2) to allow a natural way to engage in (follow-up) explanatory dialogue. It is considered more natural to show both queries and answers in a textual format, rather than change media from text to diagrams. In the case of events, text seems the most appropriate format, since events are higher-level interpretations of combinations of state-based facts. Another reason for textual descriptions of facts and events is that a textual format lends itself to an ordered list-based presentation of a history along a path.

Queries. When a user clicks on some textual information element, such as a fact or event description, the system automatically generates relevant queries, from which the user can choose one to be answered by the system: an explanation query. Depending on the type of query, an answer is retrieved, or generated automatically and again presented in a textual format. Queries about the meaning of a basic information element from the modelling, or domain vocabulary are answered by retrieving a canned text fragment which is written by a modelling expert, or a domain expert. Queries about why a fact is true, or why an event takes place, are answered by generating a causal explanation.

Causal Explanations. Causal explanations include descriptions of facts and/or events, as mentioned above, but add an important element: causal links between facts and/or events. A causal explanation is meant to explain why a fact is true, or why an event takes place, by mentioning earlier or concurrent facts or events which have led to the fact or event in question.

Contrastive Explanations. Contrastive explanations show the differences between things which are compared. For comparison of facts between different states, the global view visualization is often adequate, but it is also useful to present the differences in a textual format. Following McKeown [1985], contrastive explanations compare two things (X and Y) only, and not more. Essentially, a contrastive explanation consists of three lists of descriptions: a list of elements which hold for both X and Y, a list of elements which hold for X but not for Y, and a list with elements which hold for Y but not for X.

Exercise Questions. Didactic goals can also be addressed by presenting them to the student in the form of questions to answer. This requires students to find the relevant information and reason about it for themselves. The reason to include exercise questions is that they can motivate the student and focus the attention by setting a concrete goal [Sinclair and Coulthard, 1975, Pilkington, 1992].

Canned Texts. Hand-written, *canned* texts are used for explanations of the terminology. They are presented when a user wants to know the meaning of a term used. While the textual descriptions mentioned above describe facts and events known to WiziGarp, canned texts are explanations at the bottom-level, because this information can not be further processed by GARP or WiziGarp. Explanations of terms from the qualitative simulation terminology are built in WiziGarp, but explanations of terms in the domain vocabulary should be written by domain experts, preferably those who actually built the domain models.

4.3.3 Didactic Styles

The WiziGarp interactive learning environment can employ the means of communication in several ways, according to a preferred didactic style. The didactic style can vary along a spectrum from student-initiated exploration, where the student is completely in control and the system does not interfere, to system-initiated tutoring, where the system is in control of choosing topics to explain and questions to answer. Although more distinctions can be made along the spectrum, the two end-points are discussed below, as they exemplify the range of possibilities.

System-initiated tutoring. System-initiated tutoring is the didactic style in which WiziGarp is in control of the interaction. Based on a specific didactic plan (created by a teacher, or an automated curriculum planner), WiziGarp presents some material about a certain topic and poses some questions to the student, before moving on to another topic. How these topics are sequenced and treated using textual descriptions and/or questions is specified in the didactic plan.

Student-initiated exploration. Student-initiated exploration is the didactic style which allows the student to choose topics, navigate through the visualizations, and ask for explanations by selecting queries about certain facts of events. To offer a useful starting point for the student's exploration, a didactic plan may specify what WiziGarp should do before the student is free to explore.

4.4 System Design of WiziGarp

This section describes the technical design choices related to the implementation of the interactive explanation system WiziGarp. The main data flow related to WiziGarp is depicted in figure 4.1.

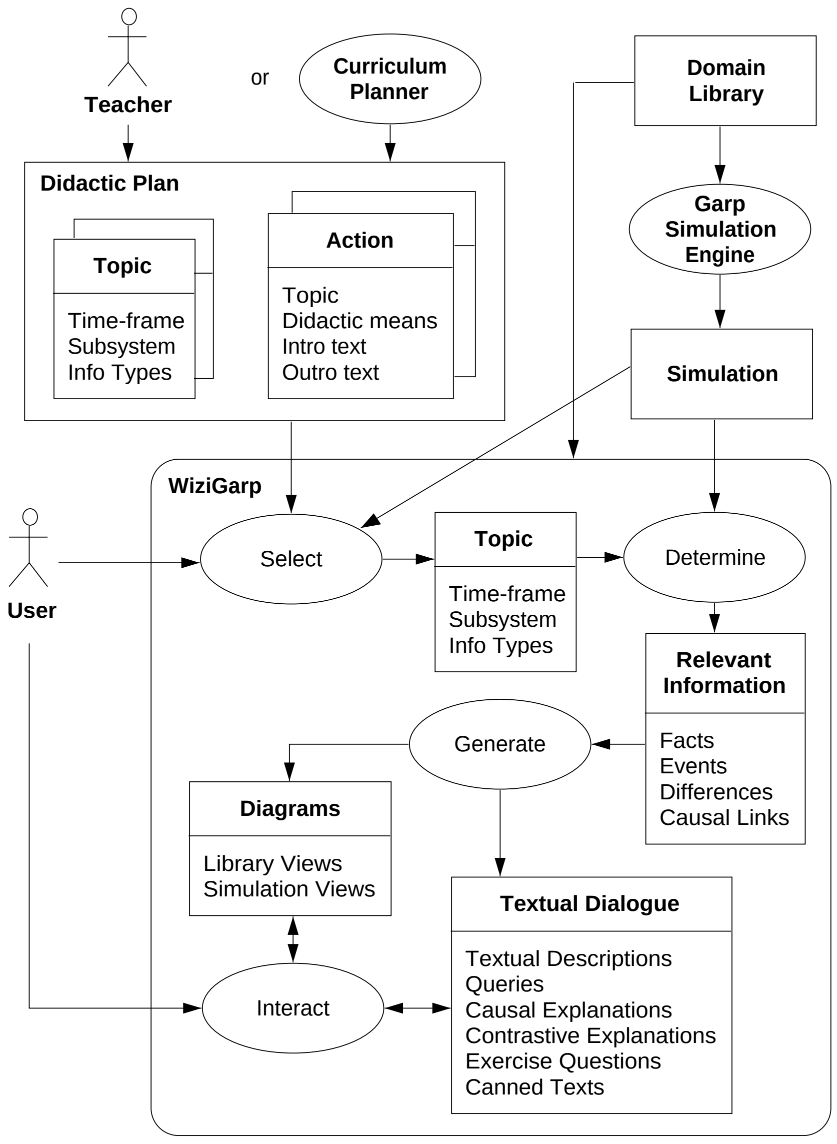


Figure 4.1
The main data flow related to WiziGarp.

The input information resources are the simulation as generated by the GARP simulation engine, and the domain library. Optional input for WiziGarp is a didactic plan, which specifies a list of topics, and a sequence of didactic actions to be performed on those topics. A didactic action specifies which type of output is generated for a particular topic. The types of output that WiziGarp generates correspond to the didactic means discussed in section 4.3.2: interactive diagrams, textual descriptions, queries (from which the student can select one to be answered by the system), causal explanations, contrastive explanations, and exercise questions (generated by the system for the student to answer) and the corresponding answers. Not shown in figure 4.1 are the options in WiziGarp to control the simulation engine. These are identical to the options available in VisiGarp, discussed in section 2.3.1. When WiziGarp is in control, the didactic plan determines which types of didactic means are generated for which kinds of topics, in which order. When the user is in control, he or she can determine all of this, with or without assistance from WiziGarp.

In the next subsections, the different parts of the system architecture will be described in more detail. The parts are a result of a decomposition by functionalities. In section 4.4.1 and 4.4.2, the processes of topic selection and determining what constitutes the relevant information are discussed, respectively. They introduce the most important internal data structures, which will reappear in some of the other processes. Section 4.4.3 deals with the interactive visualizations, while sections 4.4.4-4.4.8 describe the various parts of explanatory dialogue employed. Section 4.4.9 discusses the notion of didactic plan, which concludes the system design.

4.4.1 Topic Selection

The determination of relevant information starts with the choice of a topic, which includes a selection of time frame, subsystem, and information type, as described in section 4.3.1.1. Topics may be already specified in the didactic plan (see section 4.4.9), or they may be selected by the user, with support from WiziGarp. In the latter case, each of the three selections can be made separately, as discussed below.

4.4.1.1 Selection of time-frame

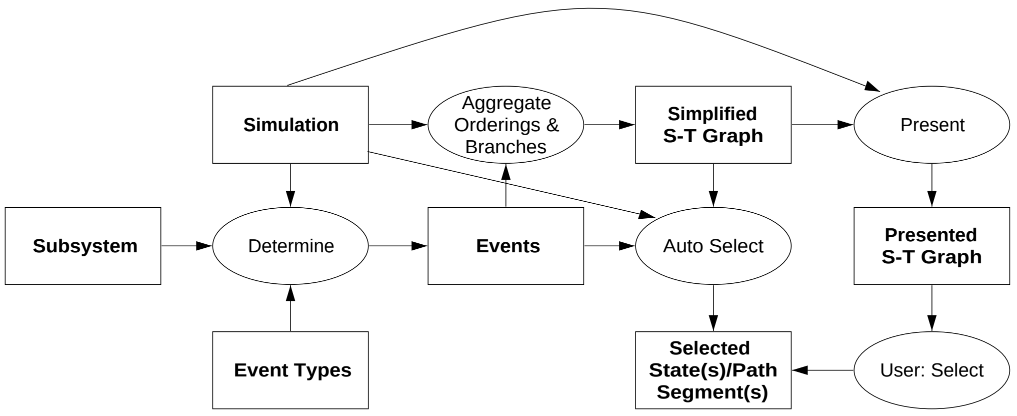


Figure 4.2
Data flow in aggregating the state-transition graph and selecting a time-frame.

In figure 4.2, the data flow in aggregating the state-transition graph and selecting a time-frame is laid out in detail. WiziGarp shows the user either the original state-transition graph of the simulation (as indicated by the curved arrow in the figure), or a simplified graph, after aggregating orderings and branches using the techniques described in chapter 3. This is indicated by the longer route to *Present S-T Graph*, which involves determining which events are relevant given a selection of types of events and, optionally, a subsystem focus. Exactly how the relevant events are determined is discussed in section 4.4.2. From the presented graph, the user may choose a particular (set of) state(s) or path segment(s) to focus on a particular time frame of the simulation. However, in a more system-directed interaction, it is also possible that a particular time-frame is selected by WiziGarp automatically, without presenting a graph to the user first. This possibility is indicated by the module called *Auto Select*. Heuristics are incorporated in WiziGarp for this purpose, based on which states remain after aggregation.

For the user, there are two views of the state-transition graph in which a time-frame can be selected: the *simulation view*, which is basically the main screen of WiziGarp's predecessor VisiGarp, shown in figure 2.8, and the *miniature state-transition graph*, which is integrated in the new main screen of WiziGarp, discussed in detail in section 4.4.3.1. The simulation view contains the original (large) version of the state-transition graph. It is possible to select one or multiple state(s), either directly in the graph or by entering them in a list format in the text entry at the bottom-leftside of the screen. Buttons are provided for selecting *All* or *None*. When the select mode is set to *Path* (the default setting is *State*), WiziGarp will try to find a path through all of the selected states. If it succeeds, it will select the path; if not, a warning message is presented.

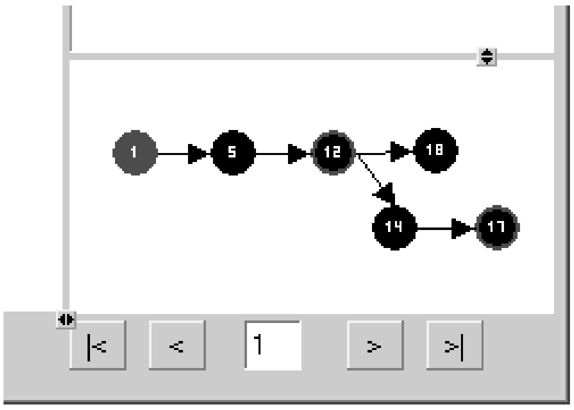


Figure 4.3
The miniature state-transition graph.

The miniature state-transition graph, displayed in figure 4.3, is a tool which offers support for selecting a particular state. This can be done by clicking on a state in the graph itself, by clicking on one of the buttons for Next (>), Previous (<), First (| <), or Last (> |) state, or by entering a state number in the number field. In principle, the Next button moves one state forward, but when reaching an end state, it will also explore alternatives, proceeding in a depth-first manner until all states have been visited. The Back button moves one state back, unless the current state is a begin state, in which case nothing happens. The First button moves back to the first begin state which precedes the currently selected state; when the current selection is already a begin state, pressing the button will move to state 1. The Last button moves to the first end state which can be reached from the current state, but if repeatedly pressed, it will move to remaining end states as well, in a depth-first manner. When a selection of states was made in the simulation view, this will be treated as a preselection in the miniature state-transition graph. This way, clicking the Next, Previous, First or Last button moves to the appropriate state in the preselection, rather than the whole graph. This is particularly useful when the state-graph is too large to explore in full. Preselected states are indicated by a red border, while the currently selected state is completely red. From the aggregation menu, options are accessible to perform an aggregation process, or to revert back to the original state-transition graph.

4.4.1.2 Selection of subsystem(s)

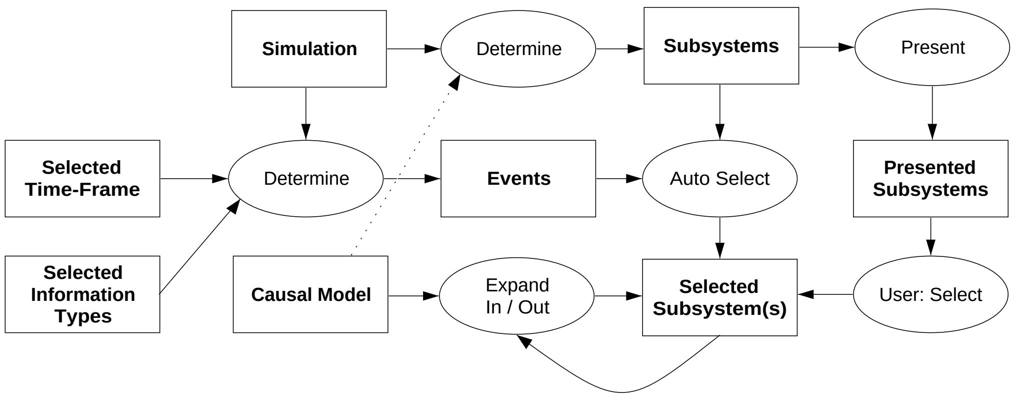
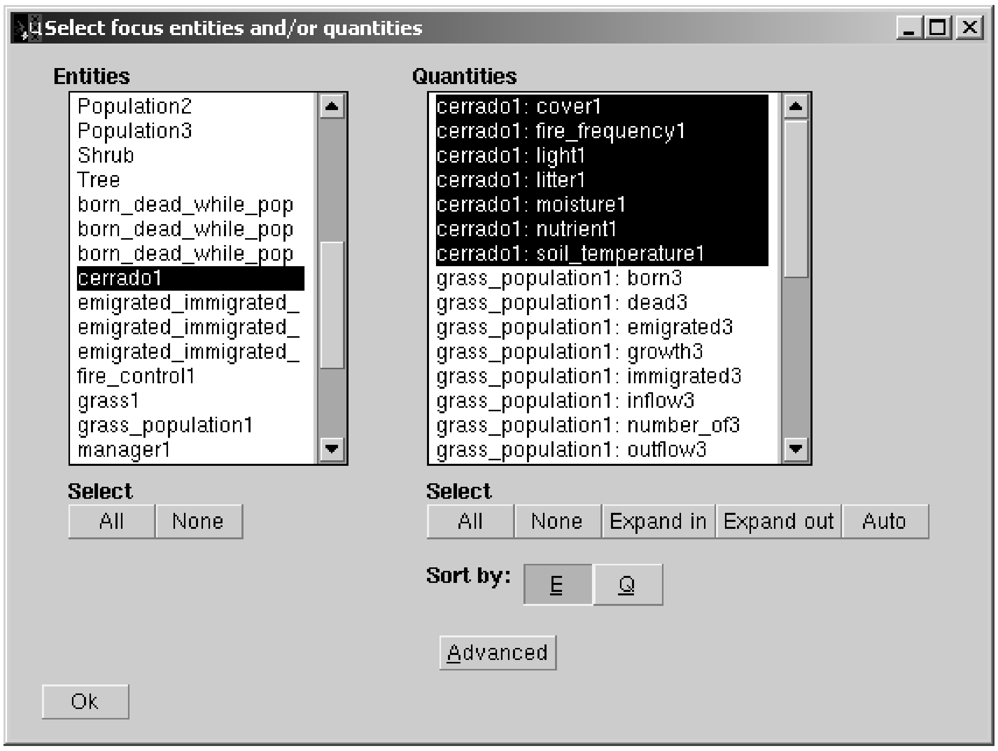


Figure 4.4
Data flow in selecting a subsystem.

The data-flow scheme for selecting a subsystem is depicted in figure 4.4. First, WiziGarp determines which subsystems are available in the simulation. Currently, this is a trivial process, because it considers every entity and quantity a possible subsystem. A more knowledge-intensive way to determine subsystems would be to analyze the causal model for cliques or other structures as candidate subsystems. In any case, the candidate subsystems are presented to the user, or fed into an auto selection mechanism, which results in a selected subsystem: a set of entities and/or quantities. The expand in/out functionality takes the current selection and uses the causal model to extend the selection backwards, or forwards. For the auto select mechanism, WiziGarp uses a heuristic measure based on which subsystems experience the most events. Which events are considered is determined by the selection of time-frame and information types, if available. In terms of the interface, this is implemented as follows. The user can specify a subsystem of interest, by clicking the Focus button, which is available in the main screen, the dependencies view, and the global view. WiziGarp then presents a focus selection window, which is displayed in figure 4.5. This window contains two list browsers, one for entities, and one for quantities to be selected. When an entity is selected in the left list browser, its quantities are automatically selected in the right list browser, to clarify which quantities will be included in the subsystem focus. This automatic selection can be further adjusted by hand, however, by deselecting selected items and selecting others in the quantities list browser. Further selection options are *expand in*, *expand out*, and *auto*. The *Expand in* function takes the currently selected quantities, finds out which quantities have causal dependencies *to* one of the selected quantities, and includes these influencing quantities in the selection. The *Expand out* function works in a similar way, but expands the selection by including the quantities which are affected *by* one of the selected quantities. The *auto* function selects quantities automatically using a heuristic measure based on the number of events occurring which involve each quantity, *i.e.*, the *N* quantities to which the highest number of events happen are included in the selection. The constant *N* is by default set to five.



Entities:
Population2
Population3
Shrub
Tree
born_dead_while_pop
born_dead_while_pop
born_dead_while_pop
cerrado1
emigrated_immigrated_
emigrated_immigrated_
emigrated_immigrated_
fire_control1
grass1
grass_population1
manager1

Quantities:
cerrado1: cover1
cerrado1: fire_frequency1
cerrado1: light1
cerrado1: litter1
cerrado1: moisture1
cerrado1: nutrient1
cerrado1: soil_temperature1
grass_population1: born3
grass_population1: dead3
grass_population1: emigrated3
grass_population1: growth3
grass_population1: immigrated3
grass_population1: inflow3
grass_population1: number_of3
grass_population1: outflow3

Figure 4.5
Focus selection for quantities and entities.

Advanced Options: Selection of Model Fragments. Within the Focus menu (for the global view only), there is a button 'Advanced' which offers advanced options concerning the selection, filtering and display of model fragment facts. When the button is clicked, a new window appears (shown in figure 4.6), which shows the MF is-a hierarchy (as described in section 2.3.13). In this hierarchy, the user can select MF types of interest, which may then be used to filter the model fragment events. Below the MF is-a selection hierarchy, there are two option menus: MF Filter and MF Names.

- The MF Filter option has two possible settings: MF (for model fragment types) or Q (for quantities). When the filter type is set to MF, only the model fragment facts are displayed for those model fragment types which have been selected in the focus MF isa-hierarchy. When the filter type is set to Q, only the model fragment facts are displayed which concern at least one of the selected quantities in the focus selection. In that case, the selection of MF types in the hierarchy is neglected.

- The MF Names option concerns the display of model fragment facts. A small reduction of information is possible using the setting 'short', rather than the default 'long'. The short format leaves out the variable names of the entities to which the model fragment applies. This saves space horizontally, at the expense of the entity information.

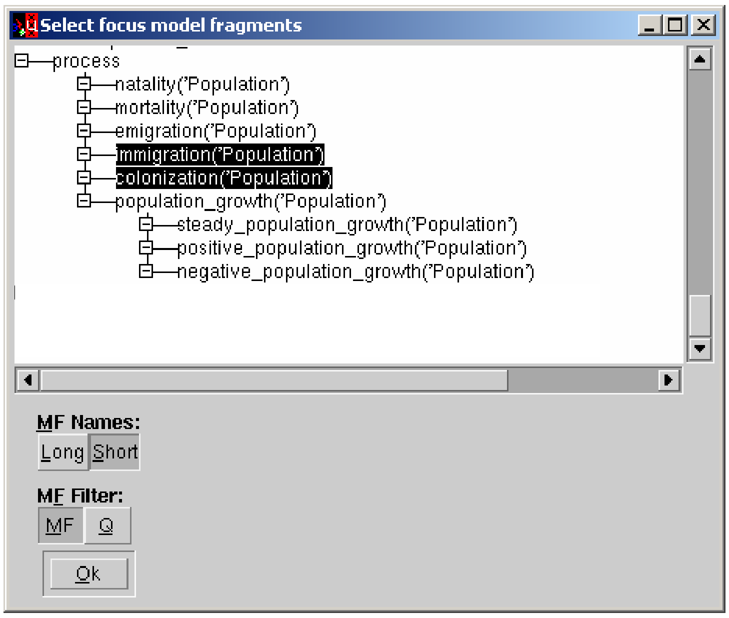

Select focus model fragments
process
natality('Population')
mortality('Population')
emigration('Population')
immigration('Population')
colonization('Population')
population_growth('Population')
steady_population_growth('Population')
positive_population_growth('Population')
negative_population_growth('Population')
MF Names:
Long
Short
MF Filter:
MF
Q
Ok

Figure 4.6
Focus selection for model fragments.

4.4.1.3 Selection of information types

The user can select the types of information he or she is interested in. This can be done separately for the types of facts, and for the types of events. Because events are based on facts, it would be possible to link these event types automatically to the corresponding fact types, but they are offered as separate options to increase flexibility. The types of facts and events which can be selected are shown in table 4.1. The left column lists the types of facts which are represented in GARP, plus the facts concerning the status of the causal effects described in section 3.5.1.1. The right column lists the event types which are recognized as described in section 3.5.2 and section 3.5.4. The horizontal lines in the table show how the different categories map to each other. The user can make a choice of these

Table 4.1
The types of facts and events which can be selected.

Type of Facts	Type of Events
E-R Structure Quantities	E-R Structure & Q
Values & Derivatives	Values & Derivatives Derivatives
Dependencies	Dependencies (In)equalities Correspondences
Causal Effects	Causal Effects
Model Fragments	Model Fragments

types by selecting the desired items from a toggle menu, which allows multiple items to be selected.

4.4.2 Determination of Relevant Information

When a topic (including a specification of time-frame, subsystem and information types) has been selected, either by the user, or by an automated curriculum planner, WiziGarp can start determining what information is relevant. Figure 4.7 offers a detailed view of the internal data flow involved in this determination process. The topic is used to filter the facts from a simulation (first by state, then by information type, then by subsystem), to arrive at the *relevant facts*. The comparison process takes the

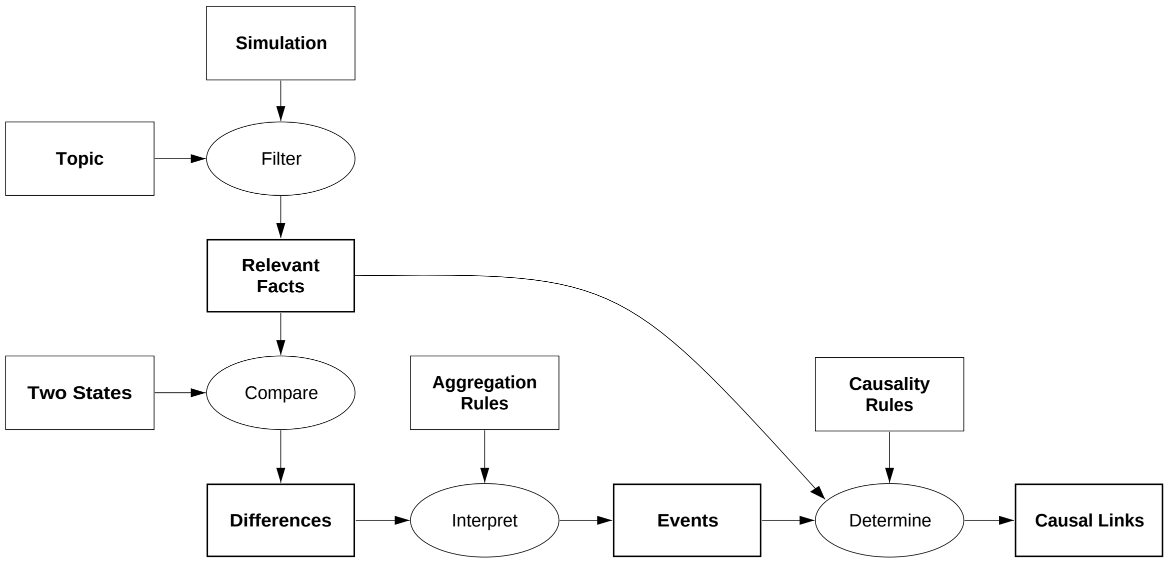


Figure 4.7
Data flow in determining relevant information.

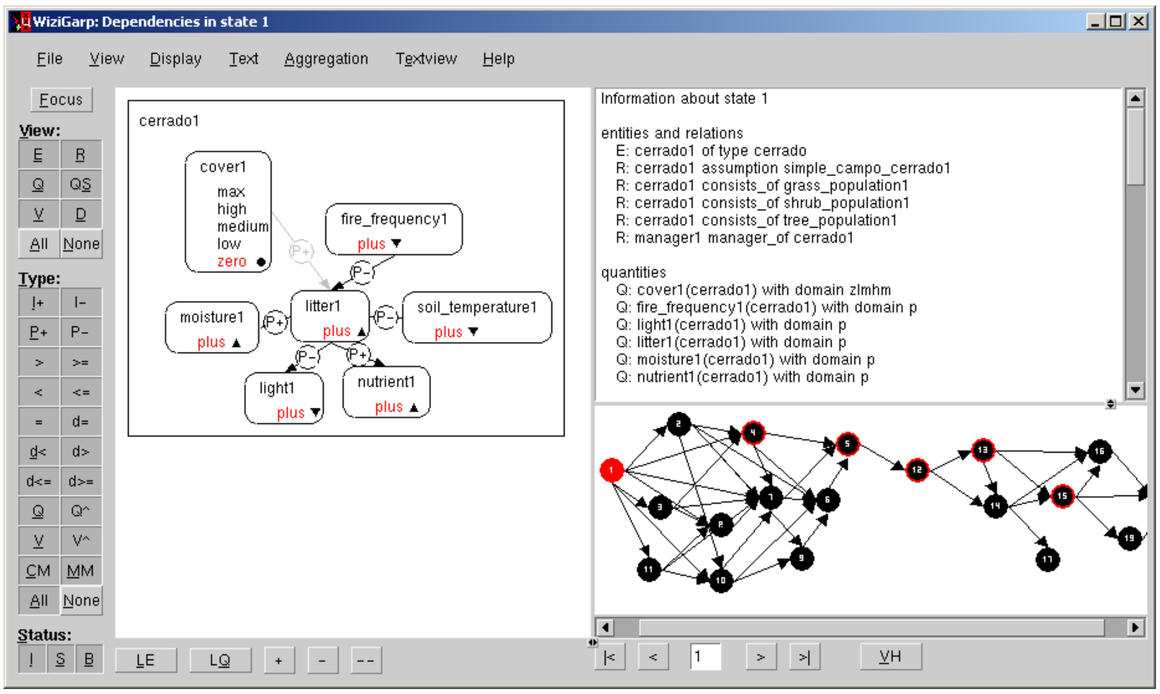
relevant facts from two states, and calculates the *differences*: a list of facts which hold for the first, but not the second state, and another list of facts which hold for the second, but not the first state. If the differences are determined for consecutive states, they can be processed further, to see if they can be interpreted as *events*. This process takes the aggregation rules, as discussed in chapter 3, as extra input. Finally, *causal links* can be determined between facts and/or events, by trying to match explanandum facts or events to the causality rules, which specify how a fact or an event may have come about. Note that each of the data structures which is displayed in a box with thick lines (*i.e., relevant facts, differences, events, and causal links*) may serve as an endpoint for the generation process, and be presented as output.

4.4.3 Interactive Visualizations

Concerning the interactive visualizations, WiziGarp is based on VisiGarp, with the following differences. Most importantly, the main screen has been changed to include a combination of information types. Second, two views have been added: the global view and the event history view. For the rest, all views that were already present in VisiGarp are still included in WiziGarp. Some of these views have been slightly improved.

4.4.3.1 The Main Screen

The main screen is a combination of a state-based visualization à la the dependencies screen in VisiGarp, a miniature version of the state-transition graph, and a text window for textual descriptions and explanatory dialogue (see figure 4.8). In VisiGarp, the main screen only displayed the state-transition



Information about state 1

entities and relations
E: cerrado1 of type cerrado
R: cerrado1 assumption simple_campo_cerrado1
R: cerrado1 consists_of grass_population1
R: cerrado1 consists_of shrub_population1
R: cerrado1 consists_of tree_population1
R: manager1 manager_of cerrado1

quantities
Q: cover1(cerrado1) with domain zlmhm
Q: fire_frequency1(cerrado1) with domain p
Q: light1(cerrado1) with domain p
Q: litter1(cerrado1) with domain p
Q: moisture1(cerrado1) with domain p
Q: nutrient1(cerrado1) with domain p

Figure 4.8
The main screen of WiziGarp, showing information in state 1 of the cerrado simulation.

graph, which is in itself rather abstract. Based on observations during the VisiGarp evaluation studies, this is not an optimal starting point for the interaction for the target audience of students without knowledge about qualitative simulation. In WiziGarp, the combination of three views gives easier access to the content of a simulation, while providing flexible navigation. First of all, the state-based visualization is included to shows the actual domain-specific content of a state, or initially, the input model. Second, the text view is added to support students' understanding of facts, events, and comparisons between states, and to allow follow-up questions for explanations of facts and events. Third, the miniature state-transition graph is included to allow navigation in an integrated way, without having to switch views.

The state-based visualization. The state-based visualization shows, by default, all entities (E), structural relations (R), quantities (Q), quantity spaces (QS), values (V), derivatives (D), and all dependencies for the selected subsystem. If no subsystem was specified in the topic selection, all will be displayed (with the risk of superimposing graphics upon others). The subsystem can be changed by clicking the focus button. The options E, R, Q, QS, V, and D, are present as toggle buttons under the View heading on the left of the screen, to hide/show the corresponding types of information. Similarly, each type of dependency can be hidden or shown using the corresponding toggle button under the Type heading.

The text view. The text view is shown next to the state-based visualization, on the right. What is displayed in the text view is determined by the textview-setting, which can take on one of four values, chosen from the Textview-menu.

- Facts: S. In the *facts* setting, the text view shows facts (of the selected information types) for the currently selected state S.
- Events: S1 $\rightarrow$ S2. In the *events* setting, the text view shows events (of the selected information types) for the state-transition S1 $\rightarrow$ S2, when the currently selected state is S2, while the previous one was S1.
- Compare: S1 vs S2. In the *compare* setting, the text view shows a contrastive explanation, comparing facts (of the selected information types) between the currently selected state S2 and the previously selected state S1.
- Auto: Events or Compare. In the *auto* setting, the text view behaves either like in the event setting, or the compare setting, depending on whether a state-transition exists between the currently selected state S2 and the previously selected state S1. If yes, then events are shown; if no, a comparison is made.

The selected information types for facts and events can be set using the menu option Fact Types, resp. Event Types, within the Aggregation-menu.

Organization of the text. The text view starts with a description of the textview-setting, before the actual content follows. The facts (or events) are organized first by information type, which is shown as a header, after which the facts (or events) are indented, to support searching by information types. For each information type, the list of relevant facts is ordered alphabetically, to facilitate searching for a particular quantity. Each fact (or event) is shown on a separate line, in the list browser format, so that it can be individually selected.

For the compare setting, there are two additional headers, just after the information type header:

```
In state S1 but not in S2:
...
In state S2 but not in S1:
...
```

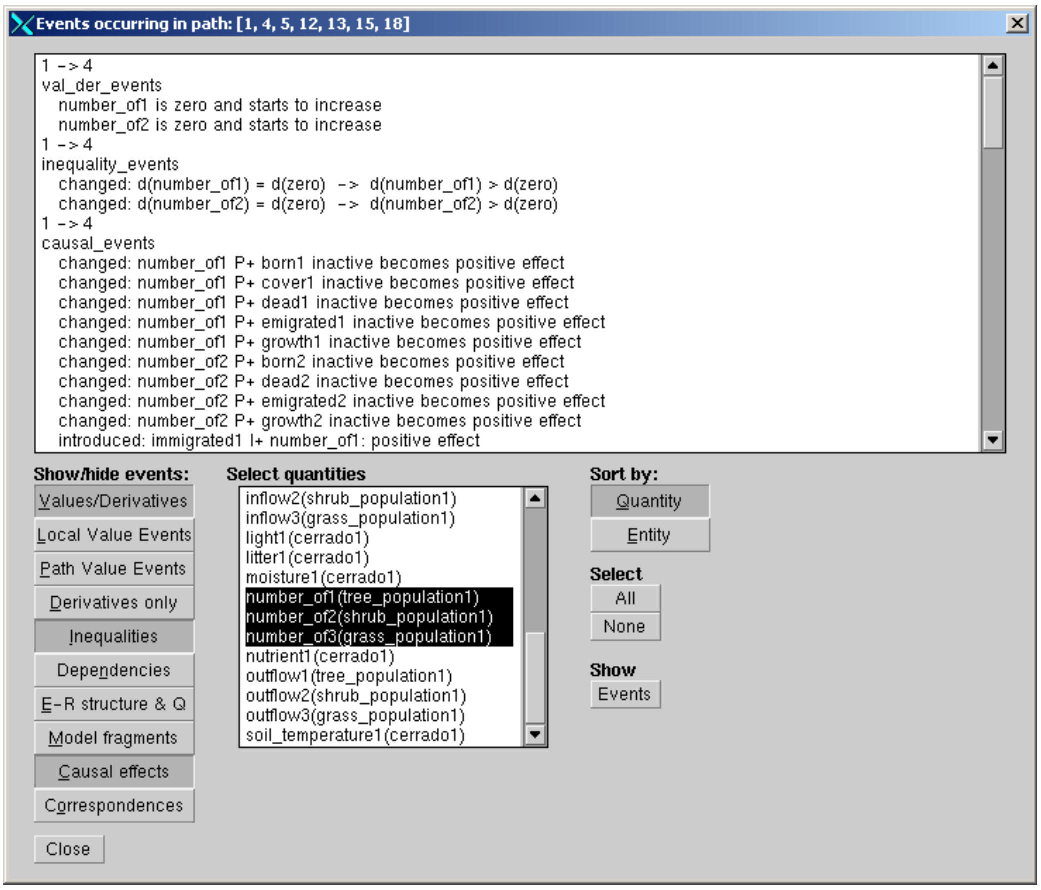
Because the list of facts (or events) may be long (depending on the topic selection), the text view has scroll-bars, which can also be controlled with keyboard shortcuts: the arrow up and down keys.

The Miniature State-Transition Graph. The miniature state-transition graph is a navigational tool, which allows easy switching between states. The user can either select a state directly, or switch between (consecutive) states using the *next, previous*, *first*, and *last* buttons. After each switch, the state-based visualization is updated. This produces a kind of interactive animation, which makes it easier to detect changes between states.

Window layout. The size of each of the three views can be adjusted horizontally and vertically, in order to allow an optimal use of the display size. For example, if the user wants to see more of the dependencies within a state in the graphical view, he or she can expand it horizontally, at the expense of the text view and miniature state-transition graph. To create a nice window layout, the text view and miniature state-transition graph are set to have the same width, but the division between them can be adjusted vertically to allow more space for the text or, alternatively, for the state-transitions graph.

4.4.3.2 The Event History View

The event history view shows a list of descriptions of the events of selected types for the currently selected path in the state-transition graph, as shown in figure 4.9. The selection of event types is done



```
1 -> 4
val_der_events
   number_of1 is zero and starts to increase
   number_of2 is zero and starts to increase
1 -> 4
inequality_events
   changed: d(number_of1) = d(zero)  ->  d(number_of1) > d(zero)
   changed: d(number_of2) = d(zero)  ->  d(number_of2) > d(zero)
1 -> 4
causal_events
   changed: number_of1 P+ born1 inactive becomes positive effect
   changed: number_of1 P+ cover1 inactive becomes positive effect
   changed: number_of1 P+ dead1 inactive becomes positive effect
   changed: number_of1 P+ emigrated1 inactive becomes positive effect
   changed: number_of1 P+ growth1 inactive becomes positive effect
   changed: number_of2 P+ born2 inactive becomes positive effect
   changed: number_of2 P+ dead2 inactive becomes positive effect
   changed: number_of2 P+ emigrated2 inactive becomes positive effect
   changed: number_of2 P+ growth2 inactive becomes positive effect
   introduced: immigrated1 I+ number_of1: positive effect
```

Select quantities:

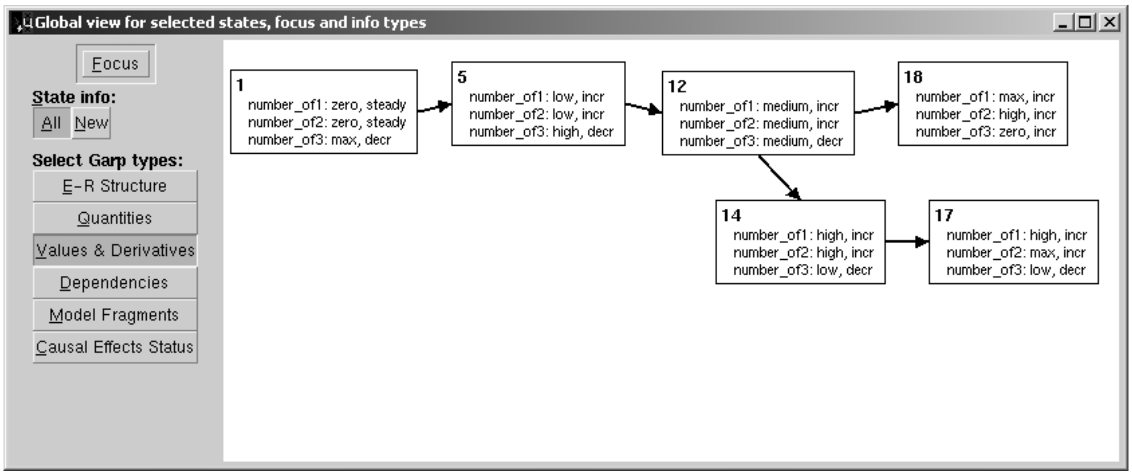
```
inflow2(shrub_population1)
inflow3(grass_population1)
light1(cerrado1)
litter1(cerrado1)
moisture1(cerrado1)
number_of1(tree_population1)
number_of2(shrub_population1)
number_of3(grass_population1)
nutrient1(cerrado1)
outflow1(tree_population1)
outflow2(shrub_population1)
outflow3(grass_population1)
soil_temperature1(cerrado1)
```

Figure 4.9
The event history view, showing events for a path from state 1 to 18 in the cerrado simulation.

by clicking the toggle buttons on the left side of the screen. Quantities can be selected to focus on events happening to the selected quantities only.

Organization of the text. The event history is organized first by state-transition, then by event type, and finally, alphabetically ordered, to support search for a particular (type of) event. Each event is shown on a separate line, in the listbrowser format, as in the WiziGarp main screen text view.

4.4.3.3 The Global View



```
1
number_of1: zero, steady
number_of2: zero, steady
number_of3: max, decr

5
number_of1: low, incr
number_of2: low, incr
number_of3: high, decr

12
number_of1: medium, incr
number_of2: medium, incr
number_of3: medium, decr

18
number_of1: max, incr
number_of2: high, incr
number_of3: zero, incr

14
number_of1: high, incr
number_of2: high, incr
number_of3: low, decr

17
number_of1: high, incr
number_of2: max, incr
number_of3: low, decr
```

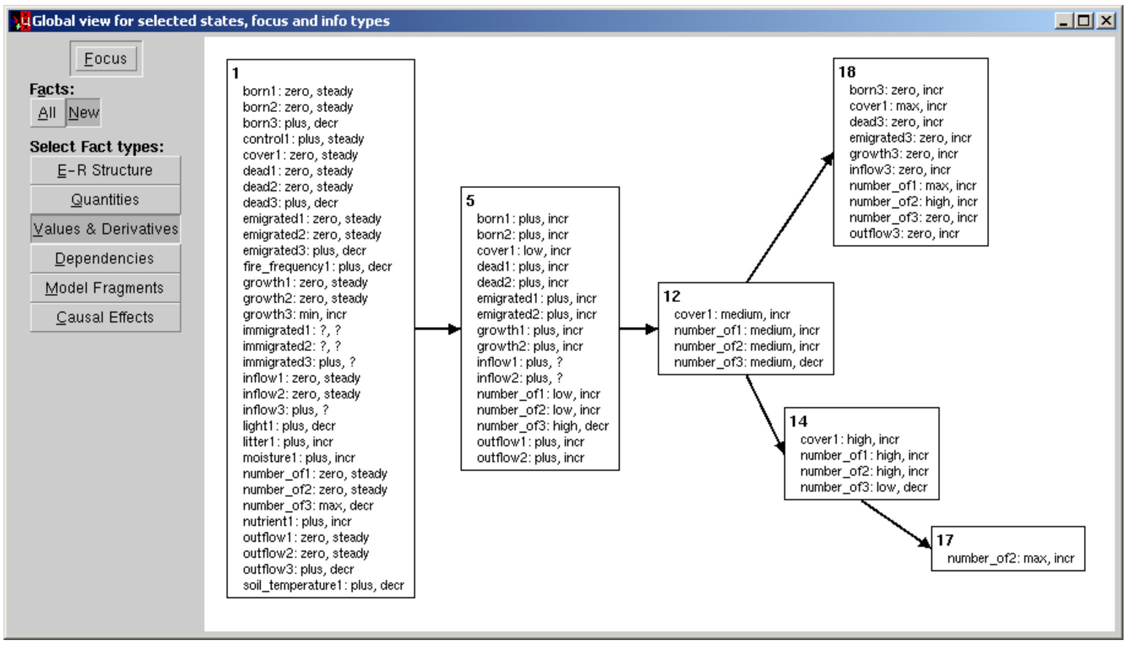
Figure 4.10
Global view showing the values and derivatives for the quantities number_of1, 2 and 3.

The global view is a combination of the state-transition graph and the text view for displaying facts. An example of the global view is shown in figure 4.10. Instead of showing only the state number in a small circle, as in the state-transition graph, the global view shows a state as a larger, rectangular box, with the number shown in boldface in the top-left corner, for easy reference. Furthermore, the box contains a list of fact descriptions, for (a selection of) the facts which hold in that particular state. Which facts are shown depends on the topic selection, which can be adjusted from within in the global view. By clicking the Focus button, a subsystem can be selected. The information types can be chosen by clicking on the toggle buttons on the left of the screen to select particular types of facts. In addition, a further reduction of information is made possible by choosing an option under the State info heading: *all* or *new*. The default is *all*, but when *new* is chosen, only the facts are shown which have just been introduced in that state (see figure 4.11 for an example). To this end, all state facts are filtered by checking whether there is a predecessor state in which the fact was not true. This seems a useful indication of when a fact is relevant - everytime it becomes true, compared to any of its predecessors. However, a list of facts takes up quite a lot of space, and multiple lists of facts (a list per state) even more. To allow a further reduction in the amount of information to be displayed, not all states are included in the global view, but only the ones which are selected in the state-transition graph. The transitions can be inspected by clicking on them, in which case a small event window will pop up, listing the events that occur in that transition in the format of the event history view described in section 4.4.3.2.

4.4.3.4 Other Improvements

Several improvements have been made to views existing in VisiGarp, which are discussed per view.

Improvements to the dependencies view. In the dependencies view, which is included in the main screen, but also still available as a separate screen, the following amendments are made (*cf.* figure 2.15 and figure 4.8).



```
1
  born1: zero, steady
  born2: zero, steady
  born3: plus, decr
  control1: plus, steady
  cover1: zero, steady
  dead1: zero, steady
  dead2: zero, steady
  dead3: plus, decr
  emigrated1: zero, steady
  emigrated2: zero, steady
  emigrated3: plus, decr
  fire_frequency1: plus, decr
  growth1: zero, steady
  growth2: zero, steady
  growth3: min, incr
  immigrated1: ?, ?
  immigrated2: ?, ?
  immigrated3: plus, ?
  inflow1: zero, steady
  inflow2: zero, steady
  inflow3: plus, ?
  light1: plus, decr
  litter1: plus, incr
  moisture1: plus, incr
  number_of1: zero, steady
  number_of2: zero, steady
  number_of3: max, decr
  nutrient1: plus, incr
  outflow1: zero, steady
  outflow2: zero, steady
  outflow3: plus, decr
  soil_temperature1: plus, decr

5
  born1: plus, incr
  born2: plus, incr
  cover1: low, incr
  dead1: plus, incr
  dead2: plus, incr
  emigrated1: plus, incr
  emigrated2: plus, incr
  growth1: plus, incr
  growth2: plus, incr
  inflow1: plus, ?
  inflow2: plus, ?
  number_of1: low, incr
  number_of2: low, incr
  number_of3: high, decr
  outflow1: plus, incr
  outflow2: plus, incr

12
  cover1: medium, incr
  number_of1: medium, incr
  number_of2: medium, incr
  number_of3: medium, decr

18
  born3: zero, incr
  cover1: max, incr
  dead3: zero, incr
  emigrated3: zero, incr
  growth3: zero, incr
  inflow3: zero, incr
  number_of1: max, incr
  number_of2: high, incr
  number_of3: zero, incr
  outflow3: zero, incr

14
  cover1: high, incr
  number_of1: high, incr
  number_of2: high, incr
  number_of3: low, decr

17
  number_of2: max, incr
```

Figure 4.11
Global view showing only the new values and derivatives, for all quantities in the simulation.

- The size of button and menu labels on the screen in the dependency view is reduced, by using (shorter) abbrevations. This leaves more space for the actual content, thereby increasing the data-ink ratio [Tufte, 1983].
- It is now also possible to hide the quantities, just like the entities, relations, quantity spaces, values and derivatives, using the toggle button Q on the left side of the screen.
- The status of causal dependencies can be shown or hidden using the toggle buttons under the Status heading: I (for inactive), S (for submissive), and B (for balanced). When turned on, these dependencies are shown in a particular colour to allow recognition of their status: grey (for inactive), blue (for submissive), and yellow (for balanced). The default colour for effective causal dependencies is black.
- There are two extra toggle buttons under the Type heading: CM (for causal model), and MM (for mathematical model). The first shows/hides the I+/I-/P+/P- dependencies with one click; the second shows/hides the mathematical dependencies (all others).
- There is a Focus button on the top-left of the screen, which allows choosing a subsystem of interest, after which the dependency view is updated.

Improvements to the value history view. The value history view has not been changed except that the current focus selection is already selected in the quantity selection listbrowser. This selection can be changed, without changing the overall focus, however.

Improvements to the transition details view. The transition details view makes use of the textual description of facts (discussed in section 4.4.4), which offers improved readability over the original GARP format.

Improvements to the model fragment views. The model fragment text view also makes use of the textual description of facts.

4.4.4 Generation of Textual Descriptions

Textual descriptions are generated to communicate facts or events. To allow selection of individual facts and events, each fact or event has its own description. The verbalizations of the different types of facts are shown in table 4.2. The format is very close to the format used in the graphical representations described in chapter 2. A longer description would perhaps be clearer for first-time users, but there is a trade-off between the use of space and clarity. Because facts simply denote the information elements present in a particular state, without any necessary further interpretation, they are kept as short as possible. Users who don't know what the format means are referred to the user guide.[2]

The entity, relationship, and quantity facts are preceded by a capital E, R, and Q, respectively, to clarify their type. This was deemed necessary because the names of entities, relationships and quantities are supplied by a model-builder, and are not guaranteed to be clear by themselves. In the description of the dependencies there are no arrows, as in the graphical representations, but this is not necessary because all dependencies are read from left to right. In the textual descriptions, the quantities are always on the appropriate side of the dependency symbol, which is not always the case in the graphical representations. For the model fragment facts, there are two alternative verbalizations. The complete version of the model fragment name includes (within brackets) the entities that it applies to, which is especially useful when a model fragment holds for multiple (combinations of) entities at the same time. The abbreviated version, without a reference to the entities, is more convenient when the list of entities involved is rather long.

The verbalizations of the value and derivative *events* are shown in table 4.3. An event denotes something that happens, which requires more interpretation than a fact. To support a natural interpretation, the event descriptions are slightly more verbose than those for facts, although also here an attempt was made to keep them short (they are not all full natural language sentences). In table 4.3, the following variables are used: Q for quantity, V for value, D for derivative, S for state.

In table 4.4, the verbalizations are listed for the other kinds of events. In this table, the following variables are used: Q for quantity, V for value, D for derivative, S for state, MF for model fragment, E for entity, CR for causal relationship, Ineq for (in)equality, Dep for dependency, CorrDep for a correspondence, EffStatus for the status of a causal effect, and Descr for a fact description, as specified in table 4.2.

[2]Available from http://hcs.science.uva.nl/projects/GARP/software.html

Table 4.2
Textual descriptions of the different types of fact.

System representation	**Verbalization**
E-R Structure:	
$instance(cerrado1, cerrado)$	E: cerrado1 of type cerrado
$has\_attribute(manager1, manager\_of, cerrado1)$	R: manager1 manager_of cerrado1
Quantities:	
$cover(cerrado1, cover1, continuous, zlmhm)$	Q: cover1(cerrado1) with domain zlmhm
Values & Derivatives:	
$value(cover1, unk, zero, zero)$	cover1: zero, steady
Dependencies:	
$prop\_pos(cover1, number\_of1)$	number_of1 P+ cover1
$inf\_neg\_by(fire\_frequency1, control1)$	control1 I- fire_frequency1
$dir\_v\_correspondence(dead1, zero, number\_of1, zero)$	zero(number_of1) Vˆ zero(dead1)
$equal(growth1, min(inflow1, outflow1))$	growth1 = inflow1 - outflow1
$d\_equal(inflow2, born2)$	d(inflow2) = d(born2)
Correspondences:	
$q\_corr(Q1, Q2)$	Q1: $V1$ corresponds to Q2: $V2$ (Q)
$dir\_q\_corr(Q1, Q2)$	Q1: $V1$ corresponds to Q2: $V2$ (Qˆ)
$v\_corr(Q1, V1, Q2, V2)$	Q1: $V1$ corresponds to Q2: $V2$ (V)
$dir\_v\_corr(Q1, V1, Q2, V2)$	Q1: $V1$ corresponds to Q2: $V2$ (Vˆ)
Causal Effects:	
$eff\_status(causal\_rel(Q1,'I-', Q2), neg, effect)$	control1 I- fire_frequency1: negative effect
$eff\_status(causal\_rel(Q1,'P+', Q2), none, none)$	cover1 P+ litter1: inactive
Model Fragments:	
$system\_structures(existing(grass\_population1))$	existing(grass_population1) OR existing

Table 4.3
Verbalization templates for the value and derivative events.

Internal representation	**Verbalization**
Values & Derivatives: S_1: $dQ = zero$, S_2: $dQ = plus/min$	Q starts to increase/decrease at V in state S_2
S_1: $Q =?$, S_2: $Q = V$	Q becomes known to be V in state S_2
S_1: $dQ = plus/min$, $Q = V1$, S_2: $Q = V2$	Q increases/decreases: V1 $->$ V2 in state S_2
S_1: $dQ = plus/min/zero$, S_2: $dQ =?$	Q not longer known to incr/decrease/be steady in state S_2
S_1: $dQ = plus/min$, S_2: $dQ = zero$, $Q = V$	Q reaches V and stops incr/decreasing in state S_2
S_1: $Q = V$, S_2: $Q =?$	Q was V but becomes unknown in state S_2
S_1: $dQ = plus/min$, $Q = V$, S_2: $dQ = zero$, $Q = V$	Q stops increasing within interval V in state S_2
S_1: $dQ =?$, $Q = V1$, S_2: $Q = V2$	Q changes: V1 $->$ V2 in state S_2
$S_{1,2,3}$: $dQ = plus/min$, S_2: $Q = V$	Q crosses the point V while incr/decreasing in state S_2
$S_{1,2,..,n}$: $dQ = plus/min$, S_{n-1}: $Q = V$	Q passes through the interval V while incr/decreasing along states $[S_1, ..., S_n]$
S_1: $Q =?, dQ =?$	Q is unknown in state S_1 (value and derivative unknown)
S_1: $Q = V, dQ =?$	Q is V in state S_1 (value known, derivative unknown)
S_1: $Q = V, dQ = zero$	Q is V and steady in state S_1
S_1: $Q = V, dQ = plus/min$	Q is V and already increasing/decreasing in state S_1
S_1: $Q = V1$, S_2: $Q = V2$	Q increases to the point/interval V2 in state S_2
$local\_extreme(Q, V, plus/min,$ $[S_1, S_2, S_3])$	Q has a local maximum/minimum of V in state S_2
$critical\_value(Q, V, plus/min,$ $[S_1, S_2, S_3])$	Q temporarily stops incr/decreasing at V in state S_2
S_1: $dQ = plus/min$, S_2: $dQ = zero$, $Q = V$	Q stops increasing/decreasing at point V in state S_2
$fluctuates(Q, V1, V2, Path)$	In summary, Q fluctuates between V1 and V2
$lowest\_val(Q, V, Path, States)$	The lowest value of Q in this path is V, in states $[S_1, ..., S_n]$
$highest\_val(Q, V, Path, States)$	The highest value of Q in this path is V, in states $[S_1, ..., S_n]$

Table 4.4
Verbalization templates for the other types of event.

Internal representation	**Verbalization**
E-R Structure & Quantities:	
$added(StructuralFact)$	introduced: StructuralFactDescr
$removed(StructuralFact)$	removed: StructuralFactDescr
(In)equalities:	
$changed(Ineq1, Ineq2)$	changed: Ineq1Descr $->$ Ineq2Descr
$added(Ineq)$	introduced: IneqDescr
$removed(Ineq)$	removed: IneqDescr
Dependencies:	
$added(Dep)$	introduced: DepDescr
$removed(Dep)$	removed: DepDescr
Correspondences:	
$q\_corr\_event(...)$	Q1: $V1a->V1b$ corresponds to Q2: $V2a->V2b$ (Q)
$dir\_q\_corr\_event(...)$	Q1: $V1a->V1b$ corresponds to Q2: $V2a->V2b$ (Qˆ)
$v\_corr\_event(...)$	Q1: $V1a->V1b$ corresponds to Q2: $V2a->V2b$ (V)
$v\_corr\_event(...)$	Q1: $V1a->V1b$ corresponds to Q2: $V2a->V2b$ (Vˆ)
$added(q\_corr(CorrDep))$:	Q1: $V1a->V1b$ corresponds to introduced: CorrDep
Causal Effects:	
$changed(EffS1, EffS2)$	changed: CR: EffS1Descr becomes EffS2Descr
$added(EffStatus)$	introduced: CR: EffStatusDescr
$removed(EffStatus)$	removed: CR: EffStatusDescr
Model Fragments:	
$replaced(MFList1,$ $MFList2)$	replaced: MFList1Descr by: MFList2Descr
$added(MFList)$	introduced: MFListDescr
$removed(MFList)$	removed: MFListDescr

4.4.5 Generation of Contrastive Explanations

Contrastive explanations are defined here as an exposition of the differences between two behaviours. Behaviours can be compared if they belong to the same contrast class. Three kinds of contrast classes are distinguished: states, paths, and quantities. The first two kinds are examples of comparing time-frames; the third kind is an example of comparing subsystems.

4.4.5.1 Contrastive explanations for states

Two states are compared in terms of facts, to show what the states have in common, and how they differ from each other for selected fact types. This can be done for any two states in a simulation, not just two successive states. For example, a begin state can be compared to an end state to see which initial facts are still true, which have been removed and which have been introduced. Two end states, or two successor states of a particular state can be compared to investigate how these alternatives differ from each other. The comparison between two states S1 and S2 consists of three lists of facts:

- L1: The list of facts which are unique to state S1
- L2: The list of facts which are unique to state S2
- L3: The list of facts which are common to both states

However, when the context already shows the facts within one of the states (*i.e.*, in an accompanying visualization), L3 may be left out of the comparison. The contents of the three lists are determined using the standard method of *difference* in set theory. An example of a contrastive explanation for states is presented in the next chapter, in section 5.2.

4.4.5.2 Contrastive explanations for paths

Two paths are compared in terms of value and derivative events, to show how the behaviour of quantities is similar or different. The comparison between two paths P1 and P2 consists of three lists of events:

- L1: The list of events which are unique to path P1
- L2: The list of events which are unique to path P2
- L3: The list of abstracted events which are common to both paths

To determine the list of common events, the method of standard set difference is modified. Because the representations of events include the transitions in which they occur, they cannot be matched directly. Furthermore, to focus on the commonalities, events are matched when they can be abstracted in a meaningful way, as described in section 3.5.5.6. This method of matching with abstraction is incorporated into set difference determination as follows:

Start with L1 and L2 as the complete lists of events in path P1 and P2, respectively. L3 is empty, initially. For any event E1 in L1, search for a possible match (with or without abstraction) with an event E2 anywhere in L2. If such a match occurs, E1 and E2 are abstracted into the abstracted event E3. E3 is put in list L3, and E1 and E2 are removed from L1 and L2, respectively.

After the three lists have been determined, they are presented using verbalization templates such as described in section 4.4.4. Because the similarities are often most interesting, L3 is presented first, after which L1 and L2 follow. An example is given below, in which the behaviour of the quantity number_of3 (the population size for the grass population) is compared between two paths leading to the two different end states.[3]

```
Comparison for quantity Q = number_of3(grass_population1)

P1 = [1,2,4,5,12,13,15,18] vs. P2 =[1,3,8,7,6,5,12,14,17]

In both paths,
    number_of3 is already decreasing in state 1
    number_of3 passes through the interval high while decreasing along
        states [4,5,12] in P1 and [3,8,7,6,5,12] in P2, respectively
    number_of3 crosses the point medium while decreasing in state 12
    the highest value of number_of3 is max, in states [1,2,4] in P1,
        and [1,3] in P2, respectively

In path P1,
    number_of3 passes through the interval low while decreasing along
        states [12,13,15]
    number_of3 has a local minimum of zero in state 15
    starts to increase at zero in state 18
    the lowest value of number_of3 is zero, in states [15,18]

In path P2,
    number_of3 reaches the interval low while decreasing in state 14
    the lowest value of number_of3 is low, in states [14,17]
```

The example shows that *number_of3* exhibits similar behaviour in path P1 and P2 until state 13 and 14, respectively. In both paths, *number_of3* starts off already decreasing, passes through the interval *high*, and crosses the value *medium*. The highest value reached in both paths is *max*. The lowest value reached differs–for P1 it is *zero*, while for P2 it is *low*. To reach its lowest value *zero* in P1, *number_of1* had to pass through the interval *low*, another event which does not occur in P2. Furthermore, *number_of3* exhibits a local minimum and starts increasing again in P1, while it keeps decreasing and only reaches the value low in P2. When there are multiple lists of states represented in an event, this shows which states in P1 match which states in P2.

4.4.5.3 Contrastive explanations for quantities

The behaviour of two quantities is compared in terms of value and derivative events. In order to compare the behaviour of two quantities, a particular path must be selected for both quantities. It is often useful to compare two quantities along the same path in a simulation, but for advanced users, it is also possible to compare the behaviour of quantity Q1 along path P1 with the behaviour of quantity Q2 along path P2. The first option is regarded as a special case of the second, in which P2 = P1. The result of the comparison consists of three lists of events:

[3]The verbalization of the contrastive explanations for paths (and those for quantities) has not yet been implemented, so this functionality is not yet integrated into the graphical interface of WiziGarp. Therefore, no screen shot is provided, but instead a manual verbalization of these contrastive explanations.

- L1: The list of events which are unique to Q1 along path P1
- L2: The list of events which are unique to Q2 along path P2
- L3: The list of abstracted events which are common to both behaviours

The analysis and presentation of these lists of events are performed analogously to the contrastive explanations for paths. An example is given below, with a comparison of the behaviour of number_of1 and number_of2 for a particular path.

```
Comparison for path P1 = [1,2,4,5,12,14,15,16,18]

Quantity number_of1(tree_population1) vs.
number_of2(shrub_population1):

Both quantities:
    are zero and steady in state 1
    start to increase at value zero in state 4 and 2, respectively
    pass through the interval low while increasing along
        states [4, 5, 12]
    cross the point medium while increasing in state 12
    have a lowest value of zero, in states [1, 2, 4]

number_of1:
    passes through the interval high while increasing along
        states [12, 14, 15, 16]
    reaches the point max while increasing in state 16
    has a highest value of max in states [16, 18]

number_of2:
    reaches the interval high while increasing in state 14
    has a highest value of high in states [14, 15, 16, 18]
```

The example shows that *number_of1* and *number_of2* behave identically until state 12, after which subtle differences occur. The lowest value reached by both quantities is *zero*. The highest value reached differs–for *number_of1* it is *max*, while for *number_of2* it is *high*. To reach its highest value *max*, *number_of1* had to pass through the interval *high*, another event which does not occur for *number_of2*.

4.4.6 Query Generation and Selection

In figure 4.12, the data flow is shown for the generation and selection of a user query, *i.e.*, a request for an explanation. The user can select a quantity, entity, model fragment, fact or event, as input for the query generation process. Matching the input to the query types yields a number of possible queries, which are relevant to the selected topic(s). These queries are shown in a popup menu on the screen. The user is then able to select a particular query of interest, for which WiziGarp will then generate an appropriate answer.

The query types for each type of object are shown in table 4.5. For quantities, entities and model fragments, the queries are simply instantiated with the name of the selected quantity, entity or model model fragment. For facts and events, the relevant queries need to be determined dynamically. The

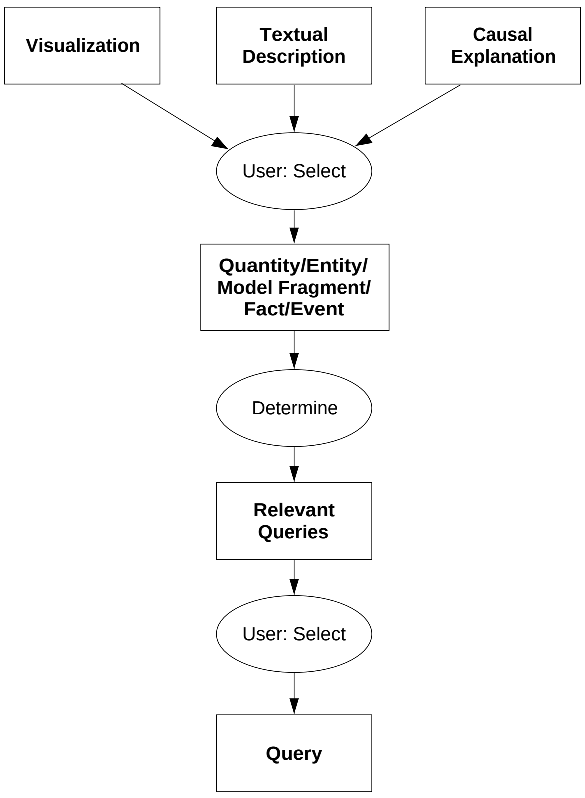


Figure 4.12
Data flow in the generation and selection of user queries.

first, why-query, is always generated, but the rest is only generated when the fact or event mentions a quantity, entity or model fragment.

In figure 4.13, the data flow involved in answering queries is shown. The queries ’What is quantity Q?’, ’What is entity Ent?’ and ’What is model fragment MF?’ are answered by retrieving canned texts which are provided by the model builder. The show-requests for value history, event history, and model fragment are replied to in the form of the value history view, event history view, and the model fragment details view, respectively. The precise formulation of the why-queries ’Why is fact F true?’ and ’Why does event E occur?’ is adapted to the particular type of fact or event. They are answered by generating a causal explanation for the fact or event in question. This is discussed in detail in section 4.4.7.

4.4.7 Generation of Causal Explanations

A causal explanation is generated for a fact or an event. This forms the answer to a causal query. For a fact in a particular state, a causal explanation shows why the fact is true in terms of other facts in the same state or preceding events which have led to the fact in question. For an event, a causal explanation shows why the event occurs, in terms of simultaneous or preceding events, or facts in a start state. In the latter case, the start state already contains the cause for the event, *i.e.*, a quantity may be already increasing, which leads to a value change event later.

Since GARP does not include events, explanatory links between facts and events must be de-

Table 4.5
Query types for quantities, entities, facts, and events.

Quantity	What is quantity Q? Show value history for quantity Q Show event history for quantity Q
Entity	What is entity Ent?
Model fragment	What is model fragment MF? Show model fragment MF
Fact	Why is fact F true? (actual formulation differs per fact type) What is quantity Q? (for every Q mentioned in F) What is entity Ent? (for every Ent mentioned in F) What is model fragment MF? (for every MF mentioned in F) Show model fragment MF (for every MF mentioned in F)
Event	Why does event E occur? (actual formulation differs per event type) What is quantity Q? (for every Q mentioned in E) What is entity Ent? (for every Ent mentioned in E) What is model fragment MF? (for every MF mentioned in E) Show model fragment MF (for every MF mentioned in E)

termined by WiziGarp. This is done on demand for an explanandum fact or event by investigating potential explanatory links to facts or events which might explain the explanandum.

4.4.7.1 Explanatory links for the different kinds of facts

To explain a fact, which holds in a particular state S, first an attempt is made to find other facts in that state which explain it. If no such fact can be found, there may be events preceding state S which explain the current state of affairs, including the explanandum fact. If still no explanation can be found, the fact may have been assumed by GARP's simulation engine. Exactly which kinds of facts and or events may be responsible differs per type of fact.

Value and derivative facts.

Why is Q1 increasing in S1? This is answered by one or more of the following facts:

```
[Qi R Q1: positive effect in state S1]
+[Qj R Q1: negative submissive effect in state S1]*
[d(Q1) = Formula]
```

The second, marked with * is not really a cause, but rather a submissive counteracting cause which by itself cannot explain the fact in question. It is included in the explanation together with the dominant causes to give a complete view of the effects which counteract each other.

Why is Q1 steady in state S? This can be explained by dependencies, or a derivative specification. An explanation by dependencies includes one or more of the following:

```
[Qi R Q1: inactive]
[Qi R Q1: balanced]
[d(Q1) = Formula]
```

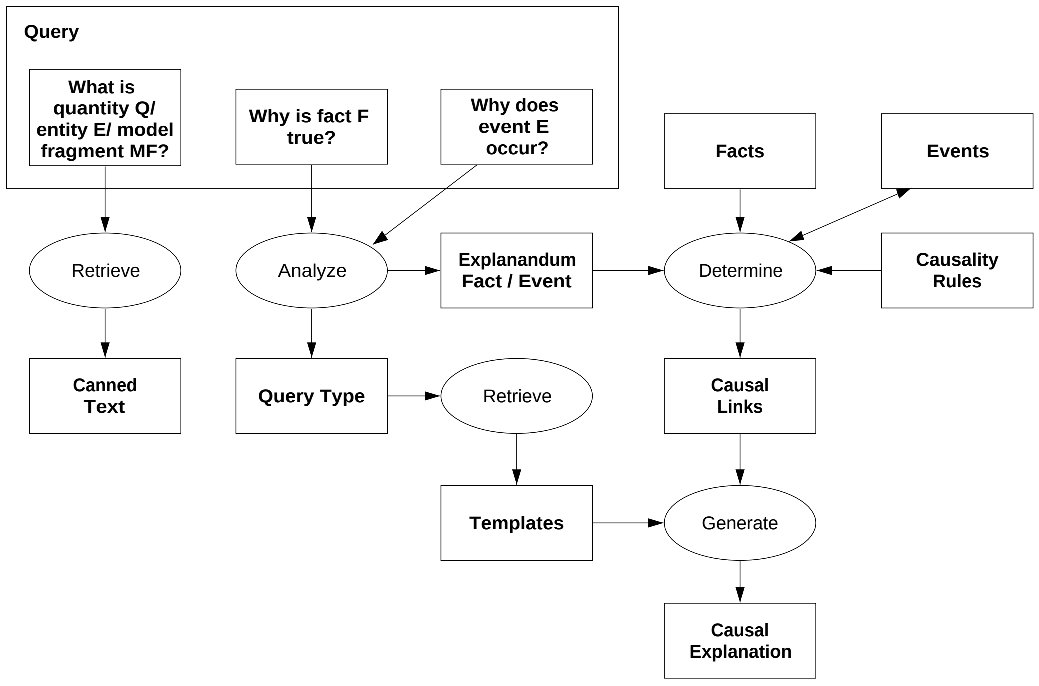


Figure 4.13
Data flow regarding the answering of user queries.

Why does Q1 have the value V1 in state S? This can be explained by a value specification, an (in)equality, a correspondence, a value event, or a value assumption. An explanation by a value specification consists of the following:

```
The value becomes known due to the following value specification(s):
     [ValueSpecification]
This is a result of the following MF(s):
     [MFList]
```

An explanation by (in)equality consists of all (in)equalities which involve quantity Q1 and value V1.

An explanation by correspondence consists of all correspondences which involve quantity Q1 and value V1.

An explanation by a value event includes the relevant value events for Q1 involving V1, which precede S1.

An explanation by a value assumption consists of the following:

```
The value becomes known due to the following value specification(s):
     [ValueSpecification]
This was an assumed condition of the following MF(s):
     [MFList]
```

(In)equality facts.

Why does the dependency Ineq hold in state S? This is explained by model fragment results, (in)equality events, or model fragment conditions.

An explanation by model fragment results includes the list of model fragments which are active in state S, and include the dependency Q1 R Q2 in their results.
The (in)equality holds as a result of the following MF(s): [MF]

An explanation by (in)equality events includes all relevant events involving Q1 R Q2 occurring before S1.

An explanation by model fragment conditions is possible due to the assumption mechanism in GARP, which can add conditions of MFs as assumptions, if not in contradiction with other information in a state. It consists of the following:
The (in)equality holds as an assumed condition of the following MF(s): [MF]

Causal effects facts.

Why does Q1 have EffectStatus effect on Q2 in state S? This is answered by the value and derivative of Q1 in state S, as these are used to determine the effect status. However, if the effect status is *balanced*, the explanation also includes the other dependencies which are involved in creating the balance.

Why is there an effect of Q1 R Q2 in state S? This is explained by the list of model fragments which are present in state S, and include the dependency Q1 R Q2 in their results.

Model fragment facts.

Why is (are) the model fragment(s) MFList active in state S? This is explained by all the conditions of the model fragments in MFList.

Rest case. If no causal explanation can be found in the information known to GARP, the following answer is presented: 'No explanation for fact F could be generated'.

4.4.7.2 Explanatory links for the different kinds of events

Because events imply a dynamic view of what happens, they are usually explained by other events. However, in some cases, events are explained by facts in the state in which the event originates.

Value and derivative events. Value and derivative events are very important events to explain in a simulation. There are multiple possibilities, as can be seen below for each type of question.

Why does Q1 increase from V1 to V2 in S1 → S2? This is answered by one or more of the following facts (the first two) or events (the rest):

```
 [Qi R Q1: positive effect in state S1]
+[Qj R Q1: negative submissive effect in state S1]*
 [Q1: V1->V2 corresponds to Q2: V3->V4 (Q)]
 [Q1: V1->V2 corresponds to Q2: V3->V4 (Q^)]
 [Q1: V1->V2 corresponds to Q2: V3->V4 (Qrev)]
 [Q1: V1->V2 corresponds to Q2: V3->V4 (V)]
```

```
    [Q1: V1->V2 corresponds to Q2: V3->V4 (V^)]
    [Q1: V1->V2 corresponds to Q2: V3->V4 (Vrev)]
    [introduced: Q1 </=</=/>=/>  Formula]
    [introduced: d(Q1) = Formula]
```

The second, marked with * is not really a cause, but rather a submissive counteracting cause which by itself cannot explain the event in question. To be complete, it is included in the explanation together with the dominant causes.

Why does Q1 start increasing in state S? This can be explained by dependencies, or a derivative specification. An explanation by dependencies includes one or more of the following:

```
    [introduced: Qi R Q1: positive effect]
    [changed: Qi R Q1: inactive becomes positive effect]
    [changed: Qi R Q1: balanced becomes positive effect]
```

Why does Q1 stop increasing in state S? This can be explained by dependencies, or a derivative specification. An explanation by dependencies includes one or more of the following:

```
    [removed: Qi R Q1: positive effect]
    [changed: Qi R Q1: positive effect becomes inactive]
    [changed: Qi R Q1: positive effect becomes balanced]
```

Why does the value of Q1 become known to be V1 in state S? This can be explained by a value specification, a value assumption, an (in)equality event, or a correspondence. An explanation by a value specification consists of the following:

```
The value becomes known due to the following value specification(s):
     [ValueSpecification]
This was a result of the following MF(s):
     [MFList]
```

An explanation by a value assumption consists of the following:

```
The value becomes known due to the following value specification(s):
     [ValueSpecification]
This was an assumed condition of the following MF(s):
     [MFList]
```

An explanation by an (in)equality event consists of all (in)equalities which involve quantity Q1 and value V1. An explanation by correspondence consists of all correspondence events which involve quantity Q1 and value V1.

Why does Q1 change from V1 in state S1 to V2 in state S2? This is explained by causal effects, correspondence, (in)equality, or value specification. The explanation consists of all of these, if present.

Why does Q1 pass through interval V1 while increasing/decreasing along states [S1, ..., Sn]? This is explained by the event that started the increase/decrease of Q1. This is found by searching backwards for a start_of_increase/decrease event in paths preceding state S1.

Why does Q1 have a maximum/minimum of V1 in state S? For a local extreme event (maximum/minimum), the explanation consists of two parts, which correspond to the two events which make up a local extreme. The first part is an explanation of the event 'stop_increasing/decreasing'. The second part is an explanation of the event 'start_decreasing/increasing'.

(In)equality events.

Why is the dependency Ineq introduced in state S? This is explained by the list of model fragments which are also introduced in state S, and include the dependency Q1 R Q2 in their results.

The (in)equality is introduced as a result of the following MF(s): [MF]

If no such model fragment can be found, this event is explained by assumed model fragment conditions. This is possible due to the assumption mechanism in GARP, which can add conditions of MFs as assumptions, if not in contradiction with other information in a state.

The (in)equality is introduced as an assumed condition of the following MF(s): [MF]

Why is the dependency Ineq removed in state S? This is explained by value and/or derivative events in the same transition for the quantities involved in the (in)equality Ineq.

Why has Q1 R1 Q2 changed into Q1 R2 Q2 in state S? The explanation for this event consists of all value and derivative specifications in state S for the quantities involved in Ineq1 and Ineq2.

Causal effects events.

Why does Q1 R Q2 change from EffectStatus1 to EffectStatus2 in state S? The answer to this query depends on the type of causal relation R, and the resulting status. If R is an influence relation (I+/I-), a change in its effect status is explained by value change events, notably changes from zero to a positive or negative value, or vice versa. If R is a proportional relation (P+/P-), a change in its effect status is explained by a derivative event, such as *start to increase/decrease*, or *stops increasing/decreasing*. If the effect status is, or was, *balanced*, explanation by value and derivative events for quantity Q1 is not sufficient, however. For these cases, the explanation also includes the events which show how other dependencies affecting Q2 are involved in creating, or removing the balance.

Why is the Status effect of Q1 R Q2 introduced in state S? This is explained by the list of model fragments which are also introduced in state S, and include the dependency Q1 R Q2 in their results.

Why is there no longer an effect of Q1 R Q2 in state S? There are two possible reasons for the removal of an effect, which are checked in the following order: the effect was a result, or an assumed condition of particular model fragment(s) which have been removed. In both cases, the list of model fragments involved is returned as the explanation.

Model fragment events.

Why is (are) the model fragment(s) MFList introduced in state S? This is explained by the *trigger conditions* of the model fragments in MFList. These are the conditions ((in)equalities, in most cases) which have become true in state S, but were not yet true in the predecessor(s) of S.

Why is (are) the model fragment(s) MFList removed in state S? This is explained by the *trigger conditions* of the model fragments in MFList which were true, but have just become false in state S.

Why is (are) the model fragment(s) MFList1 replaced by MFList2 in state S? This is explained by the combination of the explanations for the removal and introduction of model fragments described above.

Rest case. If no causal explanation can be found in the information known to GARP, the following answer is presented: 'No explanation for event E could be generated'.

4.4.8 Question Generation

For the generation of exercise questions from WiziGarp to the student, a separate module has been devised, called QUAGS (which stands for QUestions About Garp Simulations) [Goddijn, 2002, Goddijn et al., 2003]. The QUAGS question generation module is capable of asking different types of questions, addressing (changes in) values and derivatives, as well as causal and mathematical dependencies. To this end, a generic taxonomy of question types has been devised, which takes into account factors such as the topic of the question, the simulation context, and the kind of reasoning required from the student. When applying this taxonomy to simulations of moderate to high complexity (such as the running example of the cerrado simulation), this results in an enormous number of possible questions. To be of practical use, selection and ordering mechanisms are necessary to ensure that only the most relevant questions remain. When given no constraints, QUAGS searches for the most notable developments in the simulation and generates a set of questions about them. In this case, the number of questions generated is reduced by heuristics to favour the most interesting ones, fitting them together where possible, without too much overlap between questions. QUAGS can also be given more specific constraints (by the user, or in the didactic plan), such as a specific time-frame, part of the system, or a particular type of questions. This will generate, if possible, a (set of) question(s) specifically targeted at the constraints. The details of how this works are discussed in the remainder of this section.

4.4.8.1 Question Types and Criteria

In order to control the generation of relevant questions, five criteria are associated to each question type. These criteria are listed below, together with their possible values.

Perspective (P): The overall view on the simulation:
Simulation run-through (sr): changing values

Causal model (cm): reasons of change

Information Type (IT): What kind(s) of information will be used. This is specified as a set consisting of zero or more of the following items:
Causal Relations ($I+$, $I-$, $P+$, $P-$)

(In)equalities ($>, >=, =, =<, <$)

Correspondences ($Q, Q\hat{}, V, V\hat{}$)

Values

Derivatives ($d =$)

Calculations (Value formulas involving $>, >=, =, =<, <$)

Behaviour Status (BS): Which kinds of behaviour should be considered:

Real (re): representing the behaviour

Submissive (su): dominated by other effects, therefore not contributing

All (al): makes no distinction on this subject

InfoState (S): Whether the needed information is available in the current, or another state:

Present (pr): in the current state

Other (ot): in another state

Answer Method (AM): The method required to find the answer, from easy to hard:

Report (rp): the question indicates where the necessary information element can be found

Search singular (ss): the learner must search for a single information element

Search plural (sp): the learner must search for multiple information elements

Explain singular (es): the learner must reason about an information element

Explain plural (ep): the learner must reason about multiple information elements

Based on these criteria, the questions have been categorized into 23 different types. These generic question types are listed in the tables 4.6-4.8, using example questions from an imaginary simulation. Figure 4.14 shows the dependencies between the five quantities in the causal model of this simulation. In table 4.6, the questions are shown for the *causal model* perspective. Because these questions are

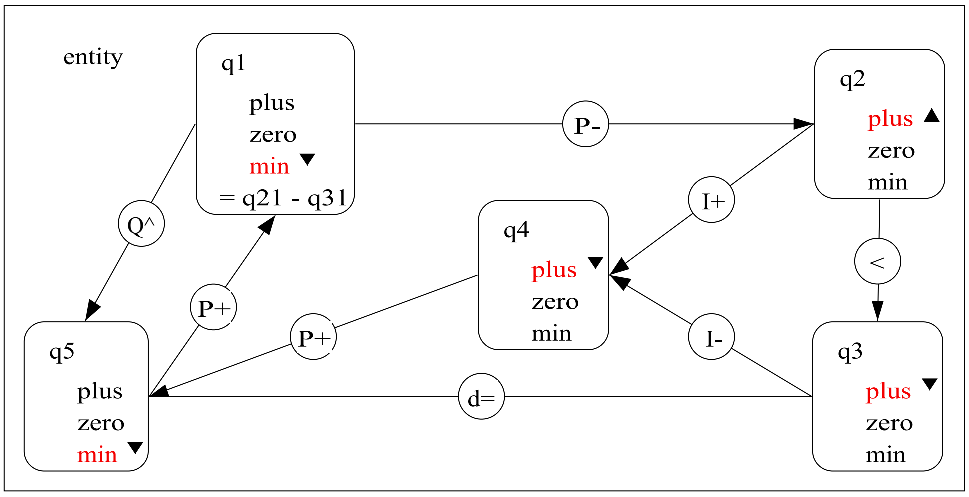


Figure 4.14
Dependencies in an imaginary simulation.

about dependencies and not about changes over states, all the necessary information can be found in one state. Therefore, the *infostate* criterion is *present* for all these questions. The other criteria differ per question type. For the meaning of AM and BS, see the discussion of the criteria above. The last column (Information Type) refers to the ontological types involved in the question, especially the types of dependencies. The answer for each question (in the context of the example) is presented in the line below each question.

Table 4.6
Questions with perspective = causal model.

Question/Answer	**AM**	**BS**	**Information Type**
Q: Which quantity is greater, Q2 or Q3? **A:** Q3	rp	re	$>, >=, =, =<, <$
Q: Describe the influence of Q3 on Q4 **A:** Q3 has a negative influence on Q4	rp	al	$I+, I-, P+, P-$
Q: What kind of connection is there between the values of Q5 and Q1? **A:** All their values correspond, they will always have the same value	rp	al	$Q, Q\hat{}, V, V\hat{}$
Q: Does Q3 influence Q4 directly? **A:** Yes	rp	al	$I+, I-, P+, P-$
Q: Which quantities have a direct influence on Q4? **A:** Q2 and Q3	ss	al	$I+, I-, P+, P-$
Q: How does Q3 influence Q1? **A:** Q3 has a negative influence on Q4 which propagates its change to Q5, which propagates its change to Q1	sp	re	Chains: $I+$, $I-$, $P+$, $P-$
Q: Does Q5 influence Q2? **A:** Yes	sp	al	Chains: $I+$, $I-$, $P+$, $P-$
Q: We saw that Q5 influences Q2 and vice versa. Does Q5 influences itself via Q2? **A:** No, the end of the path from Q5 to Q2 can't be joined to the start of the path from Q2 to Q5	sp	al	Chains: $I+$, $I-$, $P+$, $P-$
Q: Is the influence from Q2 on Q4 effective? **A:** No it is submissive	es	al	$I+, I-, P+, P-$
Q: Why does Q5 have value min? **A:** Q1 has value min and there is a quantity correspondence between Q5 and Q1	es	re	Values
Q: Why does Q1 have value min? **A:** Q1 is calculated by Q2 min Q3 and Q2 is smaller than Q3	es	re	Values
Q: Why does Q4 decrease, although Q2 is positive and Q2 has a positive influence on Q4? **A:** Q3 is positive and Q3 has a negative influence on Q4 and Q3 is greater than Q2	ep	su	$I+$, $I-$, $P+$, $P-$, $<, >$
Q: Describe the influence of Q3 on Q2 **A:** Q3 has a positive influence on Q2	ep	al	$I+, I-, P+, P-$

In table 4.7, the questions are shown which have the perspective *simulation run-through*. These questions are all about changing values over states; therefore, the *infostate* criterion is *other* for all these questions.

Table 4.7
Questions with perspective = simulation run-through.

Question/Answer	**AM**	**BS**
Q: What will be the value of Q5 in the next state? **A:** Q5s value will rise from min to zero from this state to state S1	rp	re
Q: Q4 changed from the state S to this state: its value decreased from plus to zero, what else could have happened? **A:** Q4 could have stayed plus going to state S1	rp	re
Q: What can possibly be the value of Q4 in the next state? **A:** Q4 can decrease from plus to zero, going to state(s) S1, or stay plus going to state(s) S2	rp	re
Q: Which quantity will be changed in the next state: Q4 or Q3? **A:** Just Q4, going to state(s) S1	rp	re
Q: Q4 changed from state S to this state; namely its derivative rose from min to zero, what is the effect of this change on Q1? **A:** Q1s derivative rose from min to zero	es	re
Q: What is going to happen with Q1 during the simulation? **A:** Q1 will decrease and reach minimum value and then stay steady ending in state S via states S1 and S2	ep	re

Table 4.8 shows the questions which are applicable to both the *causal model* and *simulation run-through* perspective. They can be considered as extra *causal model* questions, because the necessary information can be found in the dependencies (as shown in figure 4.14), but they are also useful in combination with *simulation run-through* questions because they are about values.

Table 4.8
Questions applicable to both perspectives.

Question/Answer	**AM**	**BS**	**Information Type**
Q: What is the value of Q5? **A:** The value of Q5 is min	rp	re	Values
Q: Which values can Q5 adopt? **A:** Min, point (zero), and plus	rp	al	Quantity spaces
Q: Why does Q4 decrease? **A:** Q3 is positive and Q3 has a negative influence on Q4	es	re	$I+$, $I-$, $P+$, $P-$
Q: Why does Q5 decrease? **A:** Because the derivative of Q5 is equal to the derivative of Q3	es	re	$d=$

4.4.8.2 The Topic of Questions

The criteria defined above can be used as input to QUAGS to generate only the relevant types of questions. To further direct the generation process, the topic of the questions can also be specified in terms of time-frame and subsystem.

The time-frame of questions. To ensure that QUAGS generates a meaningful set of questions, the time-frame specified by the user or the didactic plan (as described in section 4.4.1.1) is modified if necessary, depending on the perspective criterion, as follows.

When the perspective is *simulation run-through* (or left undefined), the questions should include questions about changes over states. This requires an awareness of which states succeed each other. If the states given as input do not form a connected graph, intermediate states are added to to make the graph connected. If no states are specified at all, the shortest path(s) between start and end state(s) are taken. This heuristic aims to determine interesting states which capture the essential behaviour from begin to end, while avoiding needless repetition or ’superfluous’ detail.

When the perspective criterion is *causal model*, no graph is needed, because the information for the questions can be found in a single state. Therefore, the specified states are used directly. Only if no states are specified, again, the shortest path(s) between start and end state(s) are taken as the time-frame for generating questions.

The subsystems involved in questions. To direct the generation towards questions about a particular subsystem of interest, QUAGS offers a fine control method, involving three things: system scope, subject quantities, and forbidden subject quantities. The *system scope* defines the part of the system for which questions may be generated. The scope may be specified in terms of entities or quantities. It defines the pool of information from which QUAGS may draw to generate questions. If no scope is entered, QUAGS considers the whole system, unless the subject quantities or forbidden subject quantities are specified.

The *subject quantities* directly influence the generation process, as all questions must be about at least one of these quantities. For this reason, selection of the subject quantities determines for a large part which questions will be posed. If no quantities are defined, QUAGS uses two heuristics for determining the subject quantities. The main heuristic is that quantities that change most during the selected part of the simulation (the selected states) are considered important. This involves inspecting the actual qualitative value and (in)equality statement changes.[4] In addition to this heuristic, QUAGS considers start quantities as important, because they are the causes of the described behaviour in the simulation. A start quantity is not influenced by the system itself and is the instigator of some process in the system.

The *forbidden subject quantities* may be specified to restrict the generation of questions. If some quantities fall under the system scope but may not be explicitly named in the questions or answers, they can be indicated as forbidden. This is a useful method for a curriculum planner when there are important reasons to (temporarily) disregard particular quantities. If no forbidden subject quantities are entered, QUAGS leaves this restriction empty.

WiziGarp matches the focus quantities and/or entities in the current topic specified by the user or didactic plan (as described in section 4.4.1.2) automatically to QUAGS’ *system scope*. The subject quantities and forbidden subject quantities are by default left undefined, but they may be specified in the didactic plan for a finer control of the question generation process.

[4]The implementation of QUAGS partly preceded the implementation of the event recognition techniques discussed in chapter 3; therefore, the notion of *events* is not used explicitly in QUAGS, although the method is based on the same idea of looking at the meaningful differences between states.

4.4.8.3 The Generation Process

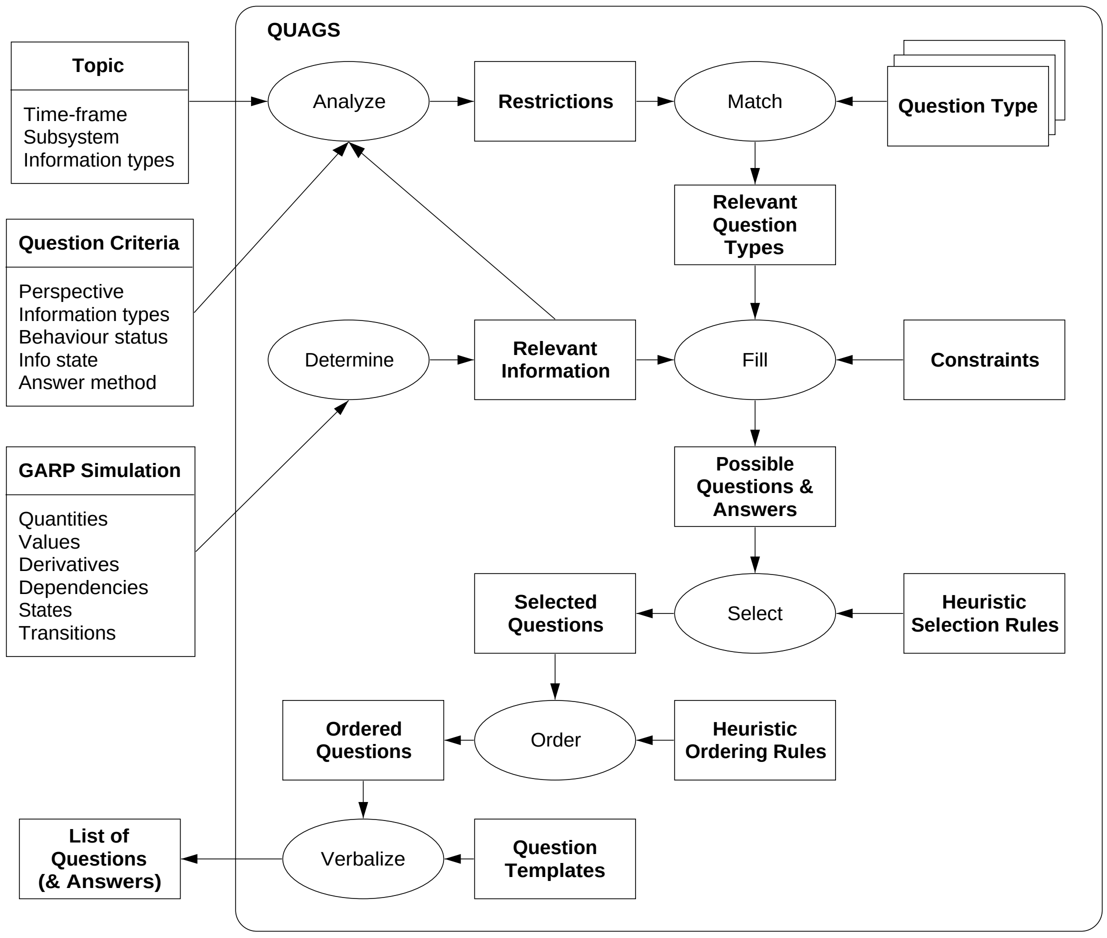
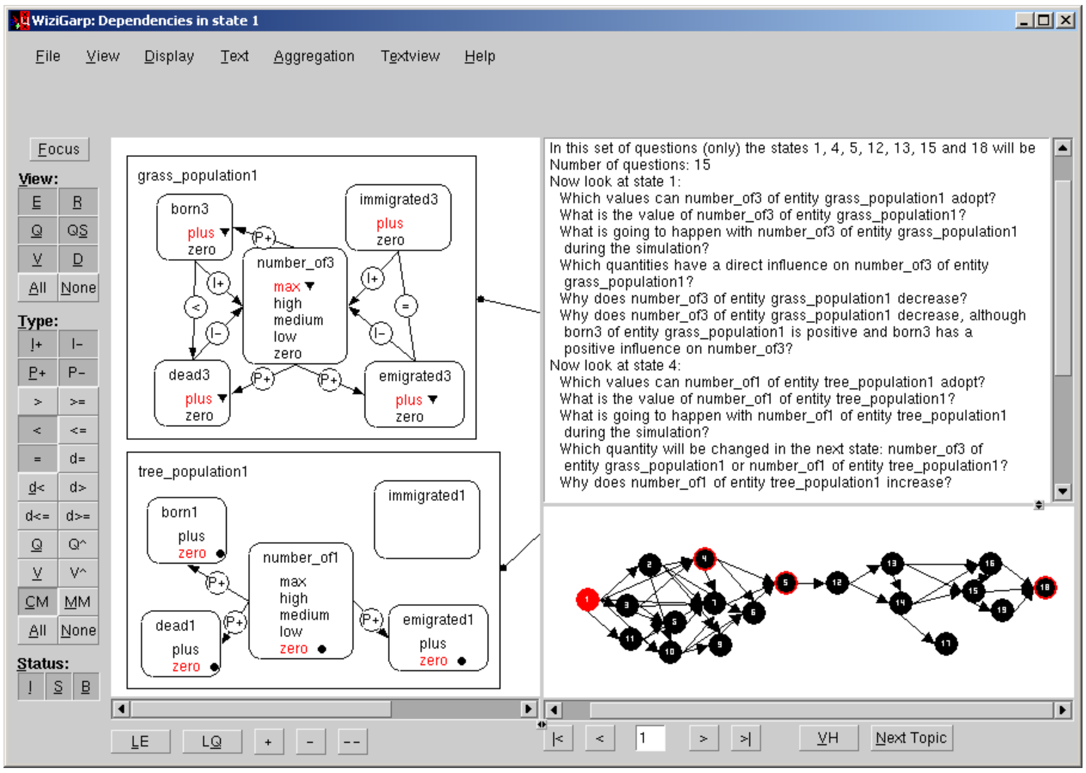


Figure 4.15
The process of generating questions by QUAGS.

QUAGS generates the questions in six main steps. The whole generation process is depicted in figure 4.15. First, the given restrictions are analyzed: the selected criteria, time-frame, and subsystem specifications. If not all of these are specified in the input, QUAGS determines the relevant information from the simulation to arrive at suitable restrictions. If *all* question type criteria are undefined, QUAGS restricts the *Behaviour Status* criterion to *Real*. This yields the most general question types. Second, the *possible* question types are matched to the criteria, resulting in the relevant question types. Third, the relevant question types are filled with the relevant information from the simulation, if all their constraints are met. The constraints, which are specific for each question type, prescribe which situations may deliver a possible interesting question. By means of these constraints the amount of possible questions is decreased. For example, a constraint may be that a quantity must have a non-zero derivative (so it is changing), in order to qualify for a question about change. Constraints may also be about the state-graph, for example demanding that a state has at least two successor states when a question is about branching. In this stage, also the answer to the questions is determined. Fourth, a selection is made from the set of questions generated so far. It is necessary to limit the number of questions as this can potentially grow very large. The goal is to make a diverse set with just a few questions of each type, as too many questions of the same sort would lead to a loss of user interest. Therefore, questions will be dropped when some other question is better suited. One general mechanism is that questions about quantities with a high grade of change (which is also used in the determination of subject quantities described above) are considered more important than questions about quantities with a lower grade. Another mechanism used is that some questions are only, or especially meaningful when combined with other questions, possibly from other types. For example, the question about a quantity is relevant when there is another question that asks why the quantity has this value, or what is going to happen with this value during the simulation. Besides deciding which questions should be selected, QUAGS must also determine when the questions should be posed, because most questions are available in several states. Some questions trigger information that can be valuable in later stages; therefore these questions are posed in the first state possible. Fifth, the selected questions are ordered such that they lead the learner from the start to the end of the simulation. To this end, the questions are grouped by state, and these groups are ordered following the sequence in which they occur in a path through the simulation. Finally, the questions, which are still abstract structures, are verbalized so they can be presented to the user. This happens by filling the natural language templates of the question types (as shown in table 4.6-4.8) with the information from the states in the simulation.

4.4.8.4 The Interface to QUAGS

The questions generated by QUAGS are shown in the text view within the main screen of WiziGarp, as displayed in figure 4.16. This can be done on request from the user or by specifying the generation

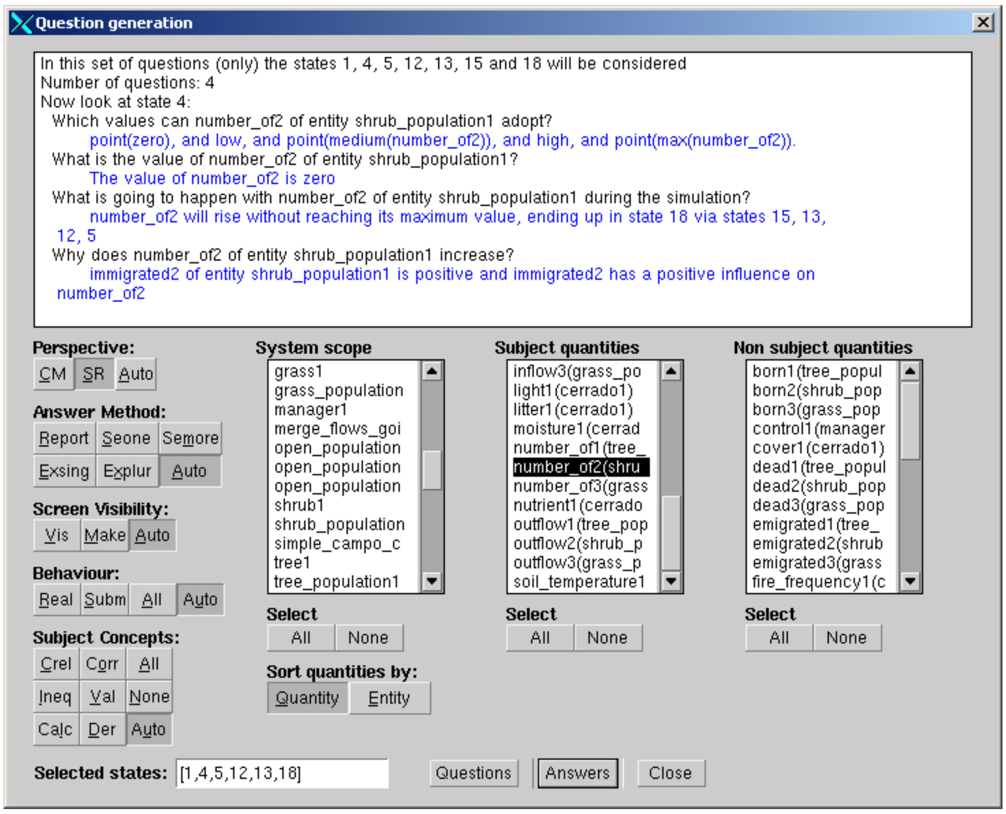


```
In this set of questions (only) the states 1, 4, 5, 12, 13, 15 and 18 will be
Number of questions: 15
Now look at state 1:
  Which values can number_of3 of entity grass_population1 adopt?
  What is the value of number_of3 of entity grass_population1?
  What is going to happen with number_of3 of entity grass_population1
   during the simulation?
  Which quantities have a direct influence on number_of3 of entity
   grass_population1?
  Why does number_of3 of entity grass_population1 decrease?
  Why does number_of3 of entity grass_population1 decrease, although
   born3 of entity grass_population1 is positive and born3 has a
   positive influence on number_of3?
Now look at state 4:
  Which values can number_of1 of entity tree_population1 adopt?
  What is the value of number_of1 of entity tree_population1?
  What is going to happen with number_of1 of entity tree_population1
   during the simulation?
  Which quantity will be changed in the next state: number_of3 of
   entity grass_population1 or number_of1 of entity tree_population1?
  Why does number_of1 of entity tree_population1 increase?
```

Figure 4.16
A screenshot of WiziGarp incorporating questions from QUAGS.

of questions as an action in the didactic plan. The user can request questions by selecting *Questions* from the *Text view* menu. When the didactic plan specifies that a topic should be handled in the form of questions, the questions are generated automatically when the didactic action is performed. In either case, a set of questions is shown for the currently selected topic, including a specific time-frame, subsystem, and types of information. The questions are introduced by referring to the states in the time-frame considered, and grouped visually by state. The subsystem is not mentioned explicitly, because this matches what is shown in the dependencies figure part of the main screen, left of the text view.

To allow users to control QUAGS directly, an additional interface has been implemented with controls for all question criteria and other restriction methods. This interface is integrated into WiziGarp as the Questions view, displayed in figure 4.17. This screenshot shows which questions are generated for a selection of subject quantities with all question criteria set to *Auto*, for the cerrado simulation. The Questions view may not be very practical for students because it has so many controls, but it is a powerful tool for expert users to investigate the kinds of questions which can be generated under different circumstances. WiziGarp does not currently allow the user to answer the questions within


Question generation

```
In this set of questions (only) the states 1, 4, 5, 12, 13, 15 and 18 will be considered
Number of questions: 4
Now look at state 4:
  Which values can number_of2 of entity shrub_population1 adopt?
        point(zero), and low, and point(medium(number_of2)), and high, and point(max(number_of2)).
  What is the value of number_of2 of entity shrub_population1?
        The value of number_of2 is zero
  What is going to happen with number_of2 of entity shrub_population1 during the simulation?
        number_of2 will rise without reaching its maximum value, ending up in state 18 via states 15, 13,
   12, 5
  Why does number_of2 of entity shrub_population1 increase?
        immigrated2 of entity shrub_population1 is positive and immigrated2 has a positive influence on
   number_of2
```

Selected states: [1,4,5,12,13,18]

Figure 4.17
A screenshot of the expert interface to QUAGS, the Questions view.

the interface. The user is expected to think about the questions by oneself, or work them out on paper. As mentioned at the bottom of the text view, a user can request to see the answers to the questions, by selecting *Questions and Answers* from the *Text view* menu. When the goal is to assess students' ability to answer such questions, additional work is necessary, because the current setup allows students to ’cheat’ by looking at the answers before thinking about the questions themselves.

4.4.8.5 Evaluation of QUAGS

QUAGS has been evaluated by domain experts to find out whether the questions generated are indeed adequate. Three experts were consulted who constructed complex models using GARP. The evaluation consisted of two parts: a questionnaire and an evaluation form. The questionnaire consisted of three parts focussing on the simulations created by the experts. The first part was about the full simulation, whereas the second and third part focussed on specific aspects of the state-graph. In all parts, the experts were asked to write down an ordered set of tutoring questions about their simulations and to indicate the most important quantities. They were also asked to give the answers to the questions, and to explain these answers. Then, the experts were allowed to interact with QUAGS, and asked to evaluate the taxonomy of question types that QUAGS can generate. The experts indicate in the evaluation form that they found the taxonomy of question types largely complete, considering the goals that QUAGS aims to cover. This is confirmed by the analyses of their questions. Of all questions posed by the experts, 80% is available in QUAGS. 50% of the questions posed by the experts are automatically generated by QUAGS when no additional constraints are given. Another 30% is selected by QUAGS if it is given the same quantities as *subject quantities* as the experts did. This difference originates from the fact that the heuristics in QUAGS are based on *changing values* (which the experts agreed with), but the experts had also reasons concerning *causality* to select quantities. Finally, 20% of the expert questions are not generated by QUAGS. Most of these questions concern situations in which *nothing happens*. For example, a question type such as: ’Why does the value of quantity Q *not* change?’. These situations can indeed be significant. The problem is that the reasons the experts give why these situations are interesting are domain-dependent and can therefore not be implemented as a heuristic in a domain-independent program. QUAGS has quite a lot of question types which are not mentioned at all by the experts. These question types can be divided into two groups. In the first group we find questions with the answer method criterion *report*. These questions are primarily the most basic ones whereas the experts tend to pose more complex questions, possibly assuming that the basic facts are already known. The second group of questions is about the status of causal relations, for example the question why some relation does not have the expected effect. These statuses are not explicitly mentioned in the simulation (at the time of this evaluation study, WiziGarp was still under development), which may be the reason why they are not mentioned by the experts.

Further details about the evaluation of QUAGS, including all comments made by the experts, can be found in Goddijn [2002] and Goddijn et al. [2003].

4.4.9 The Didactic Plan

The didactic plan describes in detail what is presented and how. To separate the what from the how, a didactic plan essentially consists of two lists: a list of topics, which specify the didactic goals, and a list of didactic actions, which specify which didactic means to use. To make an analogy with the way a book is structured and how it is used, the list of topics can be regarded as the chapters in a book, while the action list specifies what a teacher might do with these chapters (e.g., present or skip the material, give exercises, or compare with other chapters), and in which order.

In order to make the didactic plan machine-readable for WiziGarp, a grammar of topics and actions has been defined, which is shown below:

Topic T:

Subsystems: EntityList + QuantityList

Time-frame: StateList

Info types: InfoTypeList

Action A:

Preparatory action: Aggregate state-transition graph based on events of the types specified in the InfoTypeList of a particular topic T (optional)

OR

Didactic action: Present topic T on screen(*main_screen*) in the *text_view* in the form of *facts/ events/ auto/ questions/ questions+answers* with:

Intro-text TXT1 before the topic presentation, and

Outro_text TXT2 after the topic presentation

Additional question criteria for question action A (optional):

Perspective: P

Answer method: AM

Screen visibility: SV

Behaviour: BS

Subject quantities: SQ

Forbidden quantities: FQ

Each topic T in the topic list contains a specification of the subsystems (including entities and/or quantities), time-frame, and information types of interest. T stands for a unique identifier which can be used in the specification of actions. It does not denote the order in which the topics are addressed, because this is specified by the action list. This allows for more flexible planning. For example, a particular topic may be addressed first in the form of facts or events and handled again later in the form of questions (or vice versa). Or a topic may be skipped altogether, in favour of other topics on the topic list. Every didactic action A in the list of actions specifies which topic T it presents, and in what form: facts, events, questions, or questions + answers. In addition, there are preparatory actions which do not present a topic, but aggregate the state-transition graph based on the kinds of events specified in a topic. When a topic is presented, the dependencies view in the main screen is updated to show the subsystem for the first state in the selected time-frame, focussing on the selected information types. When the time-frame includes multiple states, these are preselected in the miniature state-transition graph, so that the user may move from state to state by clicking the state navigation buttons. In the text-view, the facts, events, questions, or questions + answers are shown for the selected topic, following the specified setting. When the auto setting is chosen, WiziGarp will use events whenever possible (*i.e.*, when two consecutive states are investigated directly after each other). When this is not possible, WiziGarp will resort to presentation of facts. It is possible for a teacher (or curriculum planner) to enter an intro-text and/or an outro-text which are displayed before, and after the actual topic presentation, respectively. This is useful to introduce new topics, mark topic-shifts, or give the user instructions on what to do. The variable A stands for a unique identifier which is used by the didactic planner to activate that particular action. Currently, the didactic plans use integers as identifiers to refer to specific actions as A_1 to A_n. The prototype didactic planner handles these actions in sequence, following the order specified by the integers. For the question and question + answer actions, additional question criteria are optional to fine-tune the question generation module QUAGS, as discussed in section 4.4.8. When the current topic action has been completed, the next action will be activated. Currently, the moment at which this happens is controlled by the user, by clicking on the 'Next Topic' button.[5] If WiziGarp has finished working through a didactic plan (there are no more didactic actions, which means no more goals to pursue), the user has the intiative and full control over WiziGarp, until the didactic plan is updated.

In chapter 5, two examples of possible interaction scenarios are presented: one in which the system has the initiative, and presents information and questions according to an elaborate didactic plan, and one in which the student has the initiative and explores the simulation.

4.5 Discussion

This section discusses the advantages and disadvantages of interactive explanation as implemented in WiziGarp in relation to existing work, including issues to be pursued in future work.

4.5.1 Didactic Goals Revisited

The didactic goals specified in section 4.3.1 included the terminology, generic domain knowledge, and simulated behaviour. WiziGarp provides support for students learning the terminology of qualitative reasoning according to GARP by presenting graphical representations of all knowledge types available in GARP simulations, together with textual descriptions. Support for learning generic domain knowledge is available in WiziGarp through the hierarchical views of entities and model fragments, the graphical and textual views of model fragments, and the canned texts about individual concepts (quantities, entities, and model fragments) in the domain. Support for understanding the actual behaviour as simulated is given by WiziGarp in several ways. WiziGarp shows what happens in a simulation both graphically, and textually, by describing facts and events. State-specific views show the contents of a particular state, while path-based views and textual event descriptions support the interpretation of what happens. The state-transition graph and the global view show the possible alternative behaviours that the simulated system may manifest. To avoid users being overwhelmed with detail, WiziGarp's aggregation and focussing mechanisms reduce the amount of information to be communicated, while retaining the most interesting information elements. Reasoning *about* the behaviour is supported as well. Causal explanations are presented in response to user queries about why facts and events occur. These link the explanandum back to other facts or events, ultimately tracing back to an assumption, or the initial situation. They also show how generic domain knowledge, in the form of model fragments, applies to the specific situation. The contrastive explanations support the interpretation of differences between states, and between different behaviours. The exercise questions generated by the QUAGS module support a variety of didactic goals, based on different input criteria. QUAGS generates questions which help students to get familiar with interpreting the terminology and visualizations, but also questions which require the student to make predictions and give explanations based on (multiple) facts and events in the simulation.

The didactic goals are meant to teach a student how the simulated system is structured, how it behaves, and why. If a student masters all of the didactic goals in full detail, he or she will also know a great deal about GARP. Some readers may argue that understanding how GARP works should not

[5]Strictly speaking, this button triggers the next *action* rather than the next *topic*, but this is not expected to cause confusion for students using WiziGarp, as in most cases, this involves a new topic as well.

be necessary for understanding the domain. This is not completely true, however, for two reasons. First, a qualitative simulation terminology is necessary to distinguish the relevant kinds of knowledge involved in common-sense reasoning about system behaviour. The GARP framework used in this study combines results of decades of work in qualitative reasoning [de Kleer and Brown, 1984, Forbus, 1984, Kuipers, 1984, Bredeweg, 1992, Bredeweg and Forbus, 2003] and therefore seems a good basis for such a terminology. Second, prediction as a cognitive task is not so different from the way GARP works, in functional terms. Prediction can be described as *mentally* running a simulation, at least when it is based on causal knowledge (rather than statistical knowledge). People (should) also take as input a description of an initial state, try to apply relevant chunks of knowledge (cf. model fragments), calculate the results, and update the state accordingly. The amount of information and algorithmic details involved in the GARP simulation engine do not always match closely with what goes on in people's minds [de Koning, 1997], but that is exactly one of the reasons for including the aggregation process in WiziGarp. Also for explaining why some facts are true, or why events occur, some details about the working of GARP have to be included, or else information would be missing.

4.5.2 Didactic Means Revisited

The didactic means used in WiziGarp are a combination of visualizations, and text (descriptions of facts, events, causal explanations, contrastive explanations, queries, and questions). Regarding the visualizations, WiziGarp presents several advantages over VisiGarp, by offering new views as well as improvements of existing views. In addition, WiziGarp could be extended to include another type of diagram, called *reason graph*, which explicitly shows the causal relationships between events, as is done in the EXPOUND system [Mallory, 1998].

Regarding the textual didactic means, the descriptions of facts may prove useful for learners who are unfamiliar with the visual language used in the diagrams. The descriptions of events are useful to communicate the output of the aggregation mechanisms specified in Chapter 3, which goes beyond state-based information. The excercise questions may be used to motivate learners to focus their attention and learn from their efforts to answer these questions. The fact that these questions are dynamically generated removes the need to write them manually, which is a great improvement. The queries, which are also dynamically generated, provide users with a way to pose questions about specific facts and events without having to formulate the query beforehand. A type of question that is currently not available in WiziGarp is the negative question 'Why not?'. This kind of questions may arise when a user has expectations which differ from the system's results. An explanation facility for expert systems that addresses this type of question is given by Metzler and Martincic [1998].

The causal explanations provide direct access to the reasons behind events in GARP simulations, which were hard to trace before WiziGarp. To compare this with related work, EXPOUND [Mallory, 1998] also generates causal explanations of events, such as extreme values, as WiziGarp does. EXPOUND generates multi-sentence explanations of such events, whereas WiziGarp generates explanations in the form of descriptions of facts and events in a list browser format. This difference reflects the fact that in EXPOUND, the explanations are regarded more as a final product, which is considered off-line, than in WiziGarp, where follow-up questions and explanations are generated interactively and displayed directly on the screen. Furthermore, EXPOUND only explains value and derivative events, whereas WiziGarp can explain many kinds of facts and events. Finally, WiziGarp provides contrastive explanations which compare two strands of behaviour, which goes beyond contrastive explanations of static material found in previous work [McKeown, 1985].

Nevertheless, there are also several improvements possible. The intro and outro texts are currently handcrafted, which is time-consuming. Automatically generating (parts of) such texts would reduce the effort of creating learning material.

Types of *cognitive tools* that are not incorporated in WiziGarp's didactic means are the notions of *hypothesis scratchpad*, *counterexamples*, and *monitoring tools* [van Joolingen, 1995, 1999, Luckin and Hammerton, 2002]. The hypothesis scratchpad is a means for students to state their ideas about the domain, in terms of relations between variables. When a hypothesis is stated, it can be used to run a simulation to derive the consequences of the hypothesis. In case the student's hypothesis is incorrect, it can be compared to the correct relation to generate counterexamples to the learner's hypothesis. Monitoring tools are meant to support the user's management of the learning experience by keeping track of what has been done, and what still needs to be done.

4.5.3 Generic vs. Domain-specific

The approach taken in this thesis has focused on generic methods and techniques, and is therefore fully domain-independent. This means that it can be used in any domain for which a GARP model library is available. Currently, model libraries exist for the domains of population dynamics and river management in ecology, the working of refrigerators in engineering, the container-piston dynamics in physics, and the blood circulation in biology. Extending the set of model libraries in different application domains may further augment the potential usage of tools such as VisiGarp and WiziGarp. Therefore, the domain-independent nature of the approach used in this work should be considered an important advantage. However, it is an issue to what extent teaching in a domain can be supported using *only* domain-independent methods. Grandbastien observes that teachers often express their teaching expertise 'in close relation with the domain in which they are teaching' [Grandbastien, 1999, p.344]. A domain may require its own specific didactic material (such as pictures, drawings, texts, or video) as well as specific teaching methods. If the goal is to create an effective learning environment for a particular domain, it may be best to try to integrate such materials and methods with the generic methods described here. Sharing a common basic representation method may facilitate learning to use conceptual abstractions and support the use of analogies across different domains [Falkenhainer et al., 1989].

4.6 Conclusion

In order to explain the behaviour of complex systems, qualitative simulations are a rich resource of information. In this chapter, an approach for generating interactive explanatory dialogue has been presented based on the GARP framework for qualitative simulation. The approach consists of didactic goals, didactic means, and didactic styles, and has resulted in the implementation of a prototype interactive learning environment, called WiziGarp. The didactic goals specify what needs to be communicated in order for a student to know the terminology and generic knowledge, and to understand the actual behaviour as represented in the simulation. The didactic means are the media of communication, including diagrams (as already included in VisiGarp, discussed in chapter 2), and textual dialogue. The textual dialogue in WiziGarp consists of textual descriptions of facts and events occurring in the simulation, causal explanations of why these occur in response to user queries, and exercise questions, which are automaticallly generated by a separate question generation module called QUAGS. WiziGarp takes into account the rich knowledge representation of GARP, as well as the techniques for reducing the complexity of qualitative simulations described in chapter 3, resulting in events at various levels of aggregation. Therefore, WiziGarp provides better coverage of explanations of dynamic system behaviour, compared to previous work on (simulation-based) explanation generation. Nevertheless, WiziGarp could be further expanded to include additional didactic means, such as reason graphs, answers to negative questions, a hypothesis scratchpad, counterexamples, and monitoring tools. To provide a focussed interaction, the notion of topic has been specified in detail for qualitative simulations, as a combination of a particular subsystem during a particular time-frame, with a particular set of information types of interest. Furthermore, a didactic plan representation has been defined, which specifies a list of topics and actions for each of these topics. Using different combinations of the didactic means, WiziGarp can support a range of didactic styles, from system-driven tutoring based on a specific didactic plan to free student-initiated exploration, for which no didactic plan is necessary.

5
Scenarios of Explanatory Interaction

"Simple things should be simple. Complex things should be possible."
Alan Kay (1940-), US Turing Award Winner

5.1 Introduction

This chapter presents examples of explanatory interaction with the simulation-based learning environment WiziGarp, which is described in chapter 4, followed by a more general discussion about how to incorporate tools such as WiziGarp in educational settings. Two interaction scenarios between a user and WiziGarp are discussed, to demonstrate how it works in practice, taking advantage of the potential and flexibility of the system architecture. The first scenario is an example of system-initiated tutoring, aimed at a learner new to the domain. The second scenario is an example of student-initiated exploration, which is appropriate for an advanced learner. Both scenarios make use of the GARP model of the Cerrado Succession Hypothesis (CSH) [Salles and Bredeweg, 1997, 2001a]. The Cerrado domain and the purpose of the model are assumed to be introduced to the learner before the interaction with WiziGarp starts, as was done in section 2.5.1.

5.2 Scenario 1: System-initiated tutoring

The first scenario is an example where WiziGarp has the initiative in selecting topics and performing didactic actions using different means. In order to support this, a didactic plan is specified which covers the most important topics in the domain of Cerrado ecology. The plan is aimed at a novice learner, who understands a little about qualitative reasoning, but does not have much knowledge about ecology. The didactic plan is based on the following considerations, which are inspired by research on explanation generation [McKeown, 1985, Acker et al., 1991, Cawsey, 1993, Mallory and Porter, 2000] curriculum planning [Goldstein, 1982, Winkels and Breuker, 1993], and model progression [White and Frederiksen, 1990, Salles and Bredeweg, 2001a].

- Introduce the structure before discussing the behaviour.
- Focus on the most essential quantities and entities.
- Discuss the values and derivatives before the causal effects.
- Minimize the complexity of the state-transition graph.
- When discussing values, select as time-frame begin states, branching points, end states.
- When discussing causal effects, select as time-frame a state in which the effects are active.
- Start with small topics, and increase gradually to larger topics.

- Discuss analogous topics right after each other.
- Start by telling facts; ask exercise questions later.

These domain-independent heuristics have led to the design of a topic list and action list, which together make up the didactic plan. The topic list of the didactic plan is shown in schematized form in table 5.1. The state numbers specified for the time-frame refer to the state-transition graph for the CSH simulation, as shown in figure 2.8. The first topic addresses the structure of the system, which

Table 5.1
The list of topics in the didactic plan.

N	Info Type	Subsystem	Time-frame
1.	E-R Structure	E: [cerrado1, manager1, fire_control1, tree_population1, shrub_population1, grass_population1]	State 1
2.	Val/der	Q: [number_of1, number_of2, number_of3, control1, fire_frequency1, cover1, litter1]	State 1
3.	Val/der	Idem as 2	State 5
4.	Val/der	Idem as 2	State 12
5.	Val/der	Idem as 2	State 17
6.	Val/der	Idem as 2	State 18
7.	Dependencies	E: [tree_population1], Q: [number_of1, born1, dead1, immigrated1, emigrated1]	State 5
8.	Dependencies	Analogous to 7 for shrub_population1	State 5
9.	Dependencies	Analogous to 7 for grass_population1	State 5
10.	Dependencies	E: [cerrado1, tree_population1], Q: [cover1, litter1, fire_frequency1, nutrient1, moisture1, light1, soil_temperature1, born1, dead1, number_of1, immigrated1, emigrated1]	State 5
11.	Dependencies	E: [cerrado1, shrub_population1] and associated Q	State 5
12.	Dependencies	E: [cerrado1, grass_population1] and associated Q	State 5
13.	Dependencies	E: [manager1, cerrado1]	State 5

is stable during the simulation, so it only needs to be discussed for the initial state 1. The selected subsystem contains six entities, which are considered the most essential ones in the model. Then, the behaviour is addressed in terms of values and derivatives in topics 2–6. This is done for seven of the most essential quantities within the selected entities, for several different states in the (aggregated) state-transition graph, including the begin state 1, the branching point 12, and the two end states 17 and 18. State 5 is also included to avoid the need to skip a state right away, and because it becomes important later. After discussing the values, the remaining topics 7–13 address the causal and mathematical dependencies which explain why the quantities of interest change in the way they do. For this topic, state 5 is selected as the time-frame because this is the first state in the (aggregated) state-transition graph in which all relevant processes are active. In this scenario, processes are discussed in terms of causal dependencies, not in terms of the underlying model fragments. The reason is that the dependencies specific for this system are more appropriate for the novice than the model fragments, which represent more complex model ingredients. The model fragments are more interesting for the experienced learner who is interested in this type of generic information, as discussed in scenario 2 (see section 5.3).

The idea of handling analogous topics in sequence is used in the sequence from 2 to 5, and the sequence from 7 to 9. The idea of increasing the size of topics gradually is used by discussing the causal effects per entity first (7–9), and discussing interactions between pairs of entities later (topic 10–13).

An alternative way to handle the material in topic 2–5 in the didactic plan is to define this as one topic with a larger time-frame, i.e., the whole path from state 1 to 17. In that case, the learner could have navigated through the path by clicking on the next state button '>'. This would have resulted in almost the same sequence of screenshots, but in that case, the intro and outro texts would have been the same for the whole path. Because this scenario is tuned to a novice learner, the topic time-frame is kept small, so that every state can be commented on individually.

The action list for the didactic plan is shown in table 5.2. The action list also specifies the view in which the information is presented, as well as intro and/or outro texts for every action. In the current scenario, the main screen of WiziGarp is used as the view for all actions. The intro and outro texts are not shown here to avoid repetition, as they are presented in the screenshots later in this section.

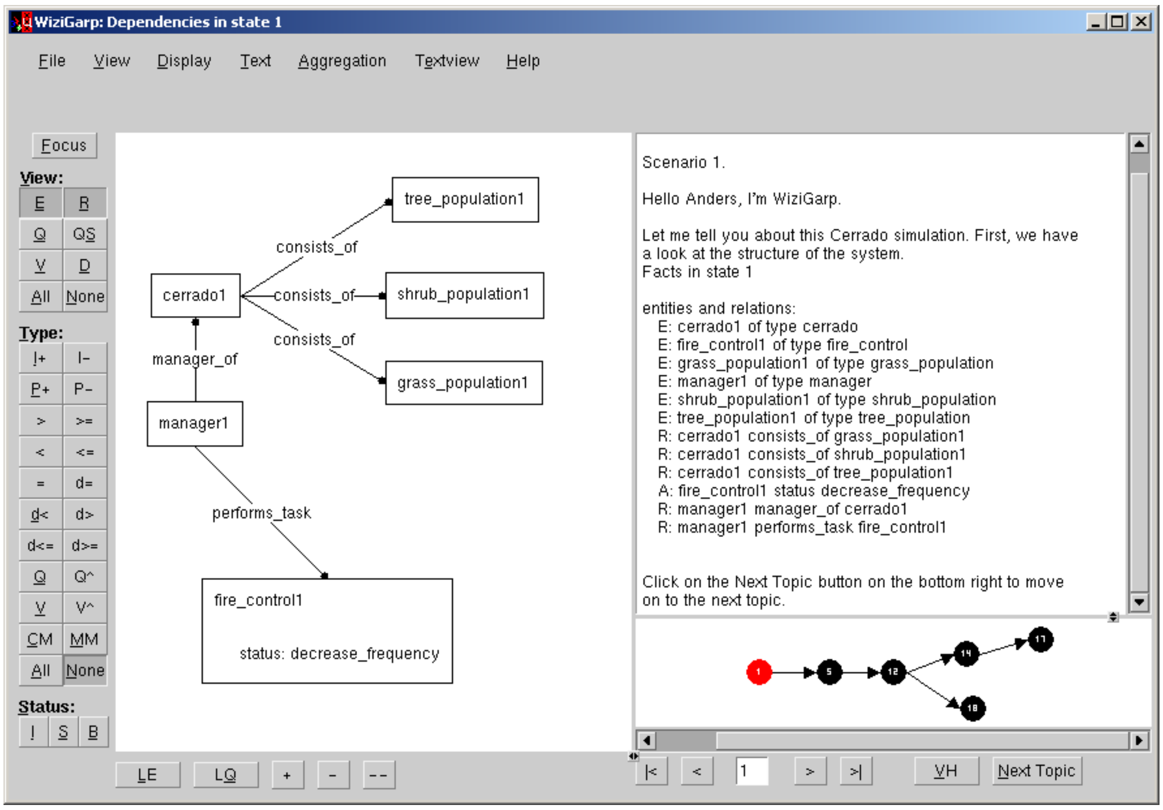
Table 5.2
The list of actions in the didactic plan.

N	Action	Topic No. (Short description)
0.	Aggregate alternative orderings	- (Val/der events)
1.	Present facts	1 (E-R structure in state 1)
2.	Present facts	2 (Val/der in state 1)
3.	Present facts	3 (Val/der in state 5)
4.	Present questions	4 (Val/der in state 12)
5.	Present facts	5 (Val/der in state 17)
6.	Present auto	6 (Val/der in state 18)
7.	Present facts	7 (Dep. tree pop. in state 5)
8.	Present facts	8 (Dep. shrub pop. in state 5)
9.	Present questions	9 (Dep. grass pop. in state 5)
10.	Present questions	10 (Dep. cerrado-trees in state 5)
11.	Present facts	11 (Dep. cerrado-shrubs in state 5)
12.	Present facts	12 (Dep. cerrado-grass in state 5)
13.	Present questions	13 (Dep. manager-cerrado in state 5)
14.	Present none	1 (E-R structure in state 1)

Action 0. Before the presentation begins, action 0 specifies the application of the technique of aggregation of alternative orderings, as defined in section 3.5.5.3. This is done to reduce the complexity of the state-transition graph. The technique abstracts paths which divert and reunite, if they include the same events, albeit in a different order. The type of events considered in the aggregation process is specified in the action definition as *val_der_events*, the value and derivative events, which is the default setting. The aggregation leads to a reduction in the number of states from 19 to 6 and a reduction in the number of paths from 896 to just 2. It is clear that without this reduction, it would be impractical to discuss what happens in all paths of the simulation.

Although the notion of events plays a role in determining what is most important, the resulting information is presented to the novice learner in terms of facts per state rather than in terms of events. Based on the considerations listed on page 153, it is deemed appropriate to start with the notion of state and state-based information before discussing state transitions and events. Thus, the action list continues with the presentation of several topics in the form of *facts*, alternated later on with comparisons between a state and the next (using the *auto* setting, described in 4.4.3.1) and *exercise questions*, to stimulate the learner to think about what happens and why.

In the current didactic plan, the list of didactic actions completely matches the order of the list of topics (except for the last action). As specified in section 4.4.9, this need not be the case. This is exemplified by the last action, which is meant as a closure: it shows topic 1 again, without a presentation of any new facts or questions.


WiziGarp: Dependencies in state 1
File View Display Text Aggregation Textview Help
Focus
View:
E R Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
tree_population1
consists_of
cerrado1
consists_of
shrub_population1
consists_of
grass_population1
manager_of
manager1
performs_task
fire_control1
status: decrease_frequency
LE LQ + - --
|< < 1 > >| VH Next Topic

Scenario 1.

Hello Anders, I'm WiziGarp.

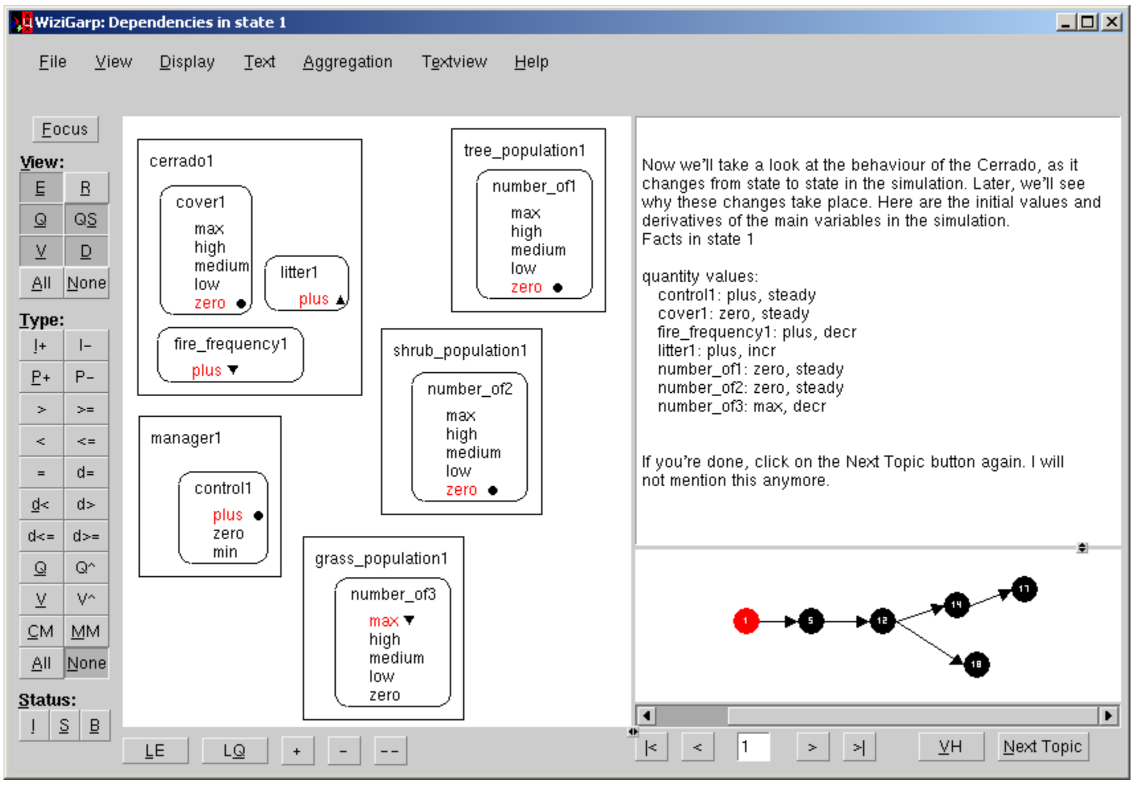
Let me tell you about this Cerrado simulation. First, we have a look at the structure of the system.
Facts in state 1

entities and relations:
E: cerrado1 of type cerrado
E: fire_control1 of type fire_control
E: grass_population1 of type grass_population
E: manager1 of type manager
E: shrub_population1 of type shrub_population
E: tree_population1 of type tree_population
R: cerrado1 consists_of grass_population1
R: cerrado1 consists_of shrub_population1
R: cerrado1 consists_of tree_population1
A: fire_control1 status decrease_frequency
R: manager1 manager_of cerrado1
R: manager1 performs_task fire_control1

Click on the Next Topic button on the bottom right to move on to the next topic.

Figure 5.1
The structure of the Cerrado domain, in terms of entities, relations and attributes.

Action 1. The presentation starts by displaying the main entities (E) in the Cerrado domain together with the relations between entities (R) and attribute relations within entities (A), as shown in figure 5.1. The model includes more entities (20 in total) than the ones specified in topic 1. The ones left out represent assumptions (*e.g.*, that the tree population is an open population) rather than real entities, or are entities that do not incorporate quantities. After filtering out these (assumption) entities, the following facts remain. The Cerrado consists of three populations: tree_population1, shrub_population1 and grass_population1. Manager1 is the manager of the Cerrado and performs the task of fire control, by decreasing its frequency. These facts are visible both in the diagram on the left and in textual format in the text view on the right. In addition, the text view also indicates the type of each entity instance, *i.e.*, grass_population1 is an entity of type grass_population. In this simulation that does not add much information, as the entity instance names are automatically generated by GARP, and already contain the type. If the model builder would supply different names, for example, 'pop1', 'pop2', and 'pop3', for the three populations, and 'Tom' for the manager, then the type indication would be important. On the left side of the diagram, the state of the buttons on the left indicate that only the entities and relations are shown, while everything else (including the quantities and dependencies) is turned off. In the text view, the intro and outro texts (which are specified for this action in the didactic plan) set the stage and tell the learner what to do next. The learner is not obliged to follow these instructions, however. In this scenario, we assume that the learner follows WiziGarp's didactic plan.



Now we'll take a look at the behaviour of the Cerrado, as it changes from state to state in the simulation. Later, we'll see why these changes take place. Here are the initial values and derivatives of the main variables in the simulation.
Facts in state 1

quantity values:
control1: plus, steady
cover1: zero, steady
fire_frequency1: plus, decr
litter1: plus, incr
number_of1: zero, steady
number_of2: zero, steady
number_of3: max, decr

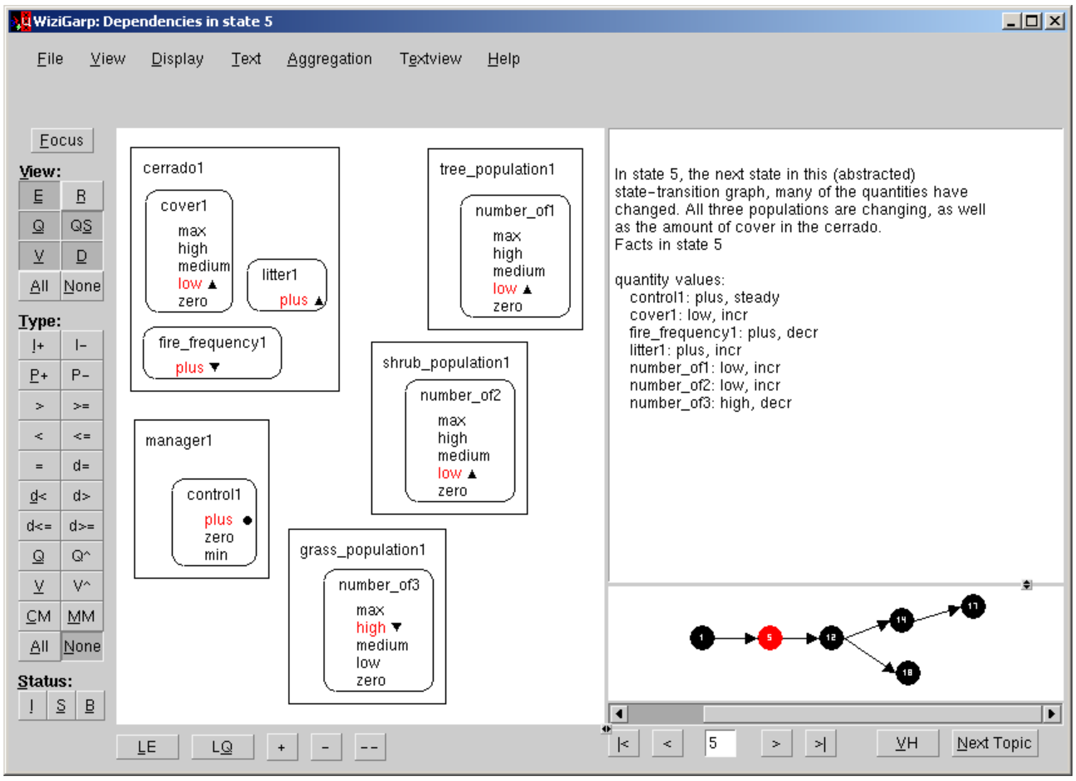
If you're done, click on the Next Topic button again. I will not mention this anymore.

Figure 5.2
The initial values and derivatives in state 1.

Action 2. After the learner clicks on the Next Topic button, WiziGarp shows the main quantities within the entities they belong to, together with their quantity spaces, values and derivatives in state 1, as shown in figure 5.2. As discussed in section 2.3.7, the current value of each quantity is shown in red, with the derivative symbol juxtaposed on the right side of the value. As discussed in section 4.4.3.1, the text view presents a textual description of the same facts in a list format, alphabetically ordered by the name of the quantities.

Everything is steady, except litter1 (increasing), fire_frequency1 (decreasing), and number_of3 (decreasing). Litter1 and fire_frequency1 have a single-valued quantity space, which means that they are changing within the interval plus. The values of control1 and number_of3 are currently at the top of their quantity space, while the values of cover1, number_of1 and number_of2 are currently zero, at the bottom of their quantity space. The entity fire_control1, which was present in the previous screenshot, has not been included anymore in the subsystem specification for the current topic, because it does not contain any quantities.

The intro text gives a short indication of WiziGarp's didactic plan, which is informed by the consideration of showing first what happens with the values, before going into the dependencies to explain why this happens.

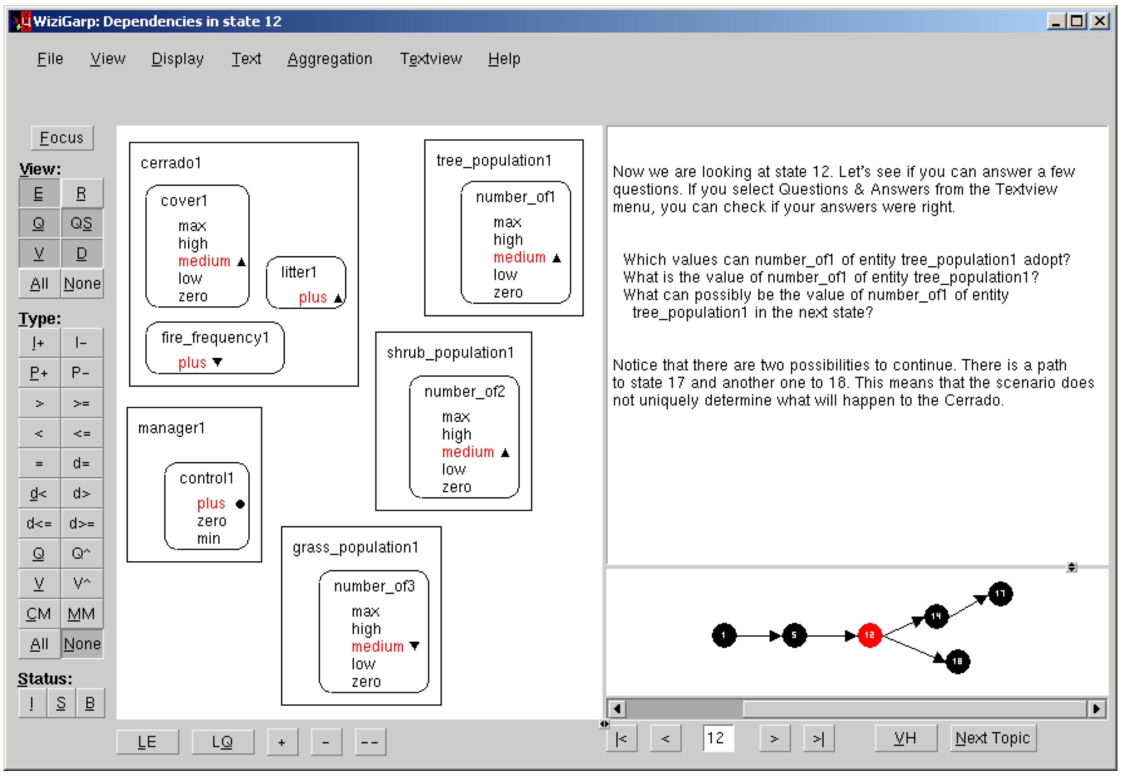


In state 5, the next state in this (abstracted) state-transition graph, many of the quantities have changed. All three populations are changing, as well as the amount of cover in the cerrado.
Facts in state 5

quantity values:
control1: plus, steady
cover1: low, incr
fire_frequency1: plus, decr
litter1: plus, incr
number_of1: low, incr
number_of2: low, incr
number_of3: high, decr

Figure 5.3
The values and derivatives in state 5.

Action 3. Next, WiziGarp moves to state 5, as shown in figure 5.3. The diagram is updated to show the new values and derivatives, and the intro and outro texts are replaced. The intro text draws attention to the fact that all population size quantities are changing now, as well as the amount of cover in the Cerrado. The intro text also mentions that the state-transition graph is abstracted, which is relevant because the transition from state 1 to 5 is an aggregated transition. An expert user would be able to recognize this, because the quantities which were steady in state 1 could not have changed directly to the next value without their derivative changing first. In the current scenario, WiziGarp deliberately glosses over these details, but scenario 2 (described in section 5.3) will show that WiziGarp has the means to explain this in detail.


WiziGarp: Dependencies in state 12
File View Display Text Aggregation Textview Help
Focus
View: E R Q QS V D All None
Type: I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status: I S B
cerrado1
cover1 max high medium low zero
litter1 plus
fire_frequency1 plus
tree_population1
number_of1 max high medium low zero
shrub_population1
number_of2 max high medium low zero
manager1
control1 plus zero min
grass_population1
number_of3 max high medium low zero
1 5 12 14 17 18
LE LQ + - --
|< < 12 > >| VH Next Topic

Now we are looking at state 12. Let's see if you can answer a few questions. If you select Questions & Answers from the Textview menu, you can check if your answers were right.

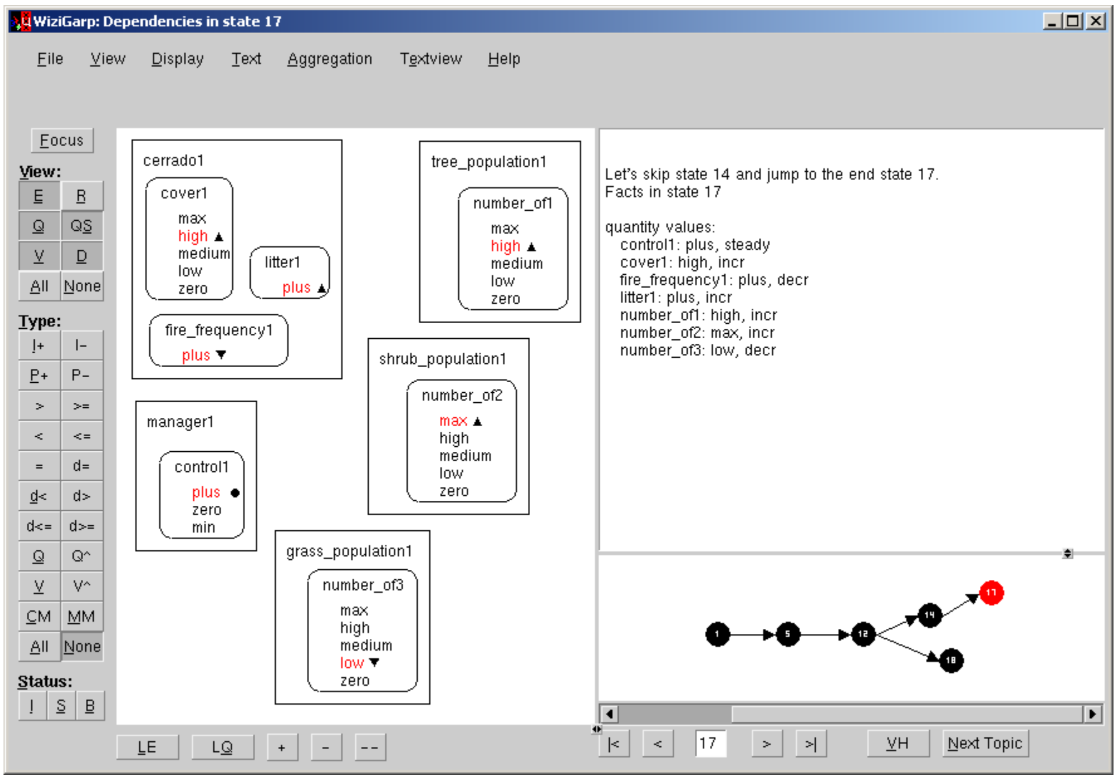
Which values can number_of1 of entity tree_population1 adopt?
What is the value of number_of1 of entity tree_population1?
What can possibly be the value of number_of1 of entity tree_population1 in the next state?

Notice that there are two possibilities to continue. There is a path to state 17 and another one to 18. This means that the scenario does not uniquely determine what will happen to the Cerrado.

Figure 5.4
The values and derivatives in state 12.

Action 4. The next didactic action specifies that exercise questions should be asked to the learner, as displayed in figure 5.4. The exercise questions are generated automatically by the question generation module QUAGS, as described in section 4.4.8. In the additional question criteria for this action, the subject quantities are restricted to one quantity only (number_of1), to avoid multiple questions of the same type. The subject concepts (values and derivatives) and time-frame (state 12) are taken from the current topic; the rest of the question criteria are left unspecified. This leads QUAGS to generate three questions, which are presented in the text view. The intro text announces the questions, and tells how the correct answers can be called up. The answers are easy to determine from the figure, and are left as an exercise to the reader.

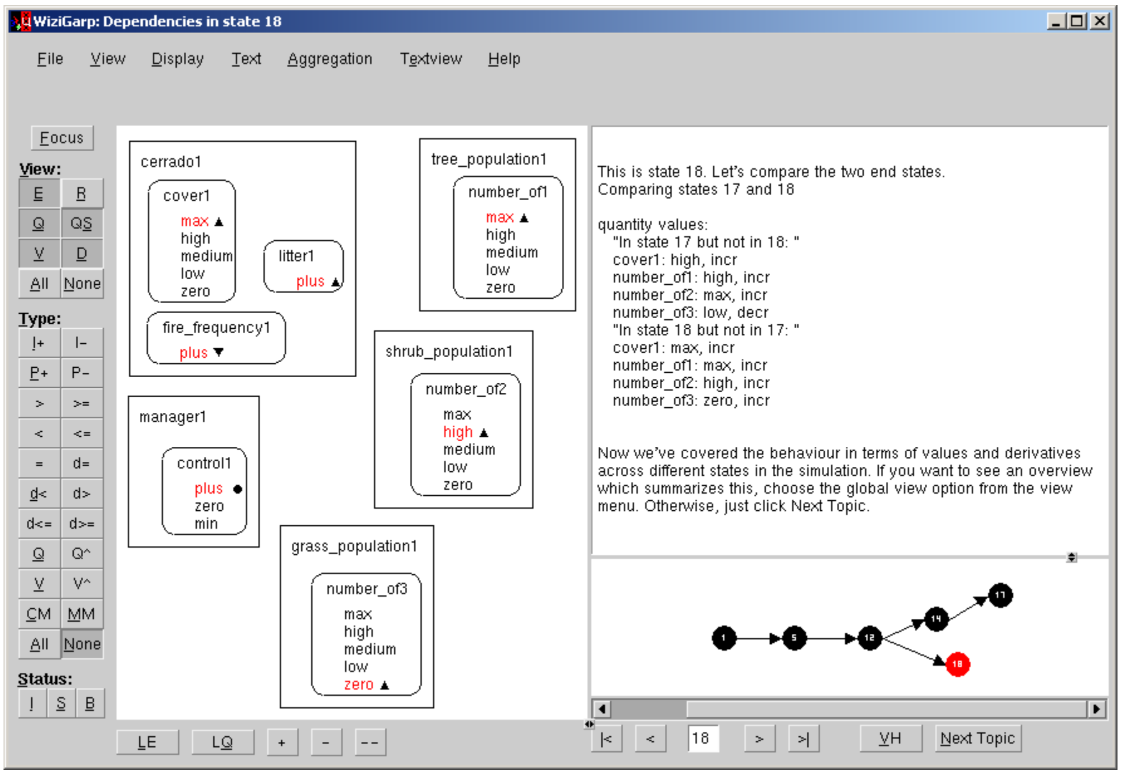
The outro text clarifies that there are two possibilities to continue from state 12. This is due to non-determinism in the qualitative simulation. An expert user may wonder why this occurs. Sometimes, non-determinism is caused by missing information in the scenario, but in this case, it is a deliberate result of the model builders. The two paths represent the idea that either the tree population, or the shrub population grows to its maximum size, but not both, due to limited resources. The mechanism of competition for resources is not modelled explicitly in this model, however. Instead, the occurrence of the two paths is determined by model fragments which represent this, and other assumptions about how the vegetation can change over time. Some of these vegetation type model fragments are shown in scenario 2 (see section 5.3).



```
Let's skip state 14 and jump to the end state 17.
Facts in state 17

quantity values:
   control1: plus, steady
   cover1: high, incr
   fire_frequency1: plus, decr
   litter1: plus, incr
   number_of1: high, incr
   number_of2: max, incr
   number_of3: low, decr
```

Figure 5.5
The values and derivatives in end state 17.

Action 5. Next, WiziGarp jumps to the end state 17 (see figure 5.5), as specified in the didactic plan, thereby skipping state 14. The intro text mentions this to ensure the learner that this is ok. In state 17, the shrub population has reached its maximum size, while the tree population does not increase above the interval high. The vegetation type model fragments mentioned earlier define the scope of possible developments in the Cerrado as ending here, so that the GARP simulation engine does not continue the simulation beyond this state. Otherwise, the grass population could have decreased further.


WiziGarp: Dependencies in state 18
File View Display Text Aggregation Textview Help
Focus
View: E R Q QS V D All None
Type: I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status: I S B
cerrado1
cover1 max ▲ high medium low zero
litter1 plus ▲
fire_frequency1 plus ▼
tree_population1
number_of1 max ▲ high medium low zero
shrub_population1
number_of2 max high ▲ medium low zero
manager1
control1 plus ● zero min
grass_population1
number_of3 max high medium low zero ▲
1 5 12 14 17 18
LE LQ + - --
|< < 18 > >| VH Next Topic

This is state 18. Let's compare the two end states.
Comparing states 17 and 18

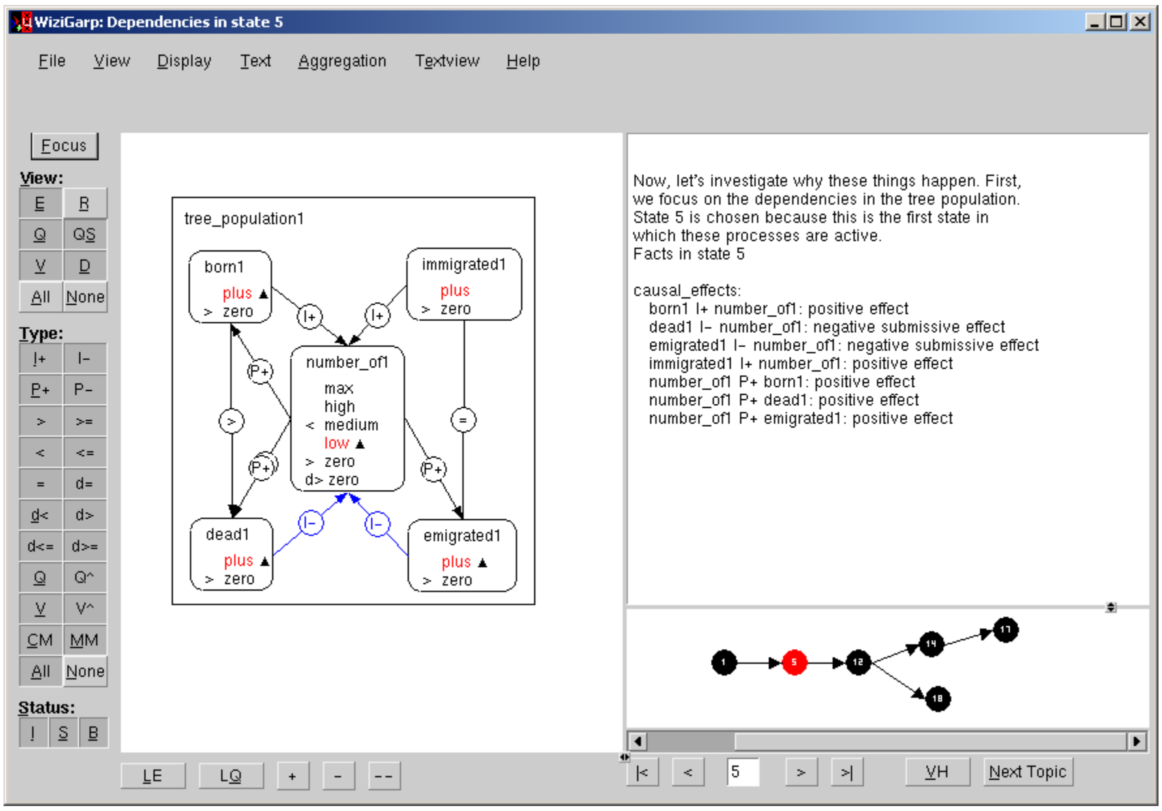
```
quantity values:
  "In state 17 but not in 18: "
  cover1: high, incr
  number_of1: high, incr
  number_of2: max, incr
  number_of3: low, decr
  "In state 18 but not in 17: "
  cover1: max, incr
  number_of1: max, incr
  number_of2: high, incr
  number_of3: zero, incr
```

Now we've covered the behaviour in terms of values and derivatives across different states in the simulation. If you want to see an overview which summarizes this, choose the global view option from the view menu. Otherwise, just click Next Topic.

Figure 5.6
The values and derivatives in end state 18, compared to state 17.

Action 6. Then, the didactic plan specifies that WiziGarp moves to state 18, and contrast it to state 17 using the *auto* setting, as explained in section 4.4.3.1 (see figure 5.6). This allows an easy comparison of the two end states. It shows that number_of1 and number_of2 reach their maximum value in state 18 and 17, respectively. Furthermore, it shows that number_of3 is still decreasing in state 17, but is *increasing* in state 18. This difference is glossed over in the current scenario in favour of other topics, but it is treated in detail in scenario 2, which is described in section 5.3.

The outro text signals a closure of this range of topics involving only values and derivatives, and mentions the possibility to open the global view for a summary of this information. But the learner does not pursue this option, and instead clicks again on 'Next Topic'. Examples of the global view occur in scenario 2 however, *e.g.*, see figure 5.19.


WiziGarp: Dependencies in state 5
File View Display Text Aggregation Textview Help
Focus
View:
E B Q QS V D All None
Type:
I+ I– P+ P– > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
! S B
tree_population1
born1 plus > zero
immigrated1 plus > zero
number_of1 max high < medium low > zero d> zero
dead1 plus > zero
emigrated1 plus > zero
I+ I+ P+ P+ P+ > = I– I–
1 5 12 14 17 18
LE LQ + - --
|< < 5 > >| VH Next Topic

Now, let's investigate why these things happen. First, we focus on the dependencies in the tree population. State 5 is chosen because this is the first state in which these processes are active.
Facts in state 5

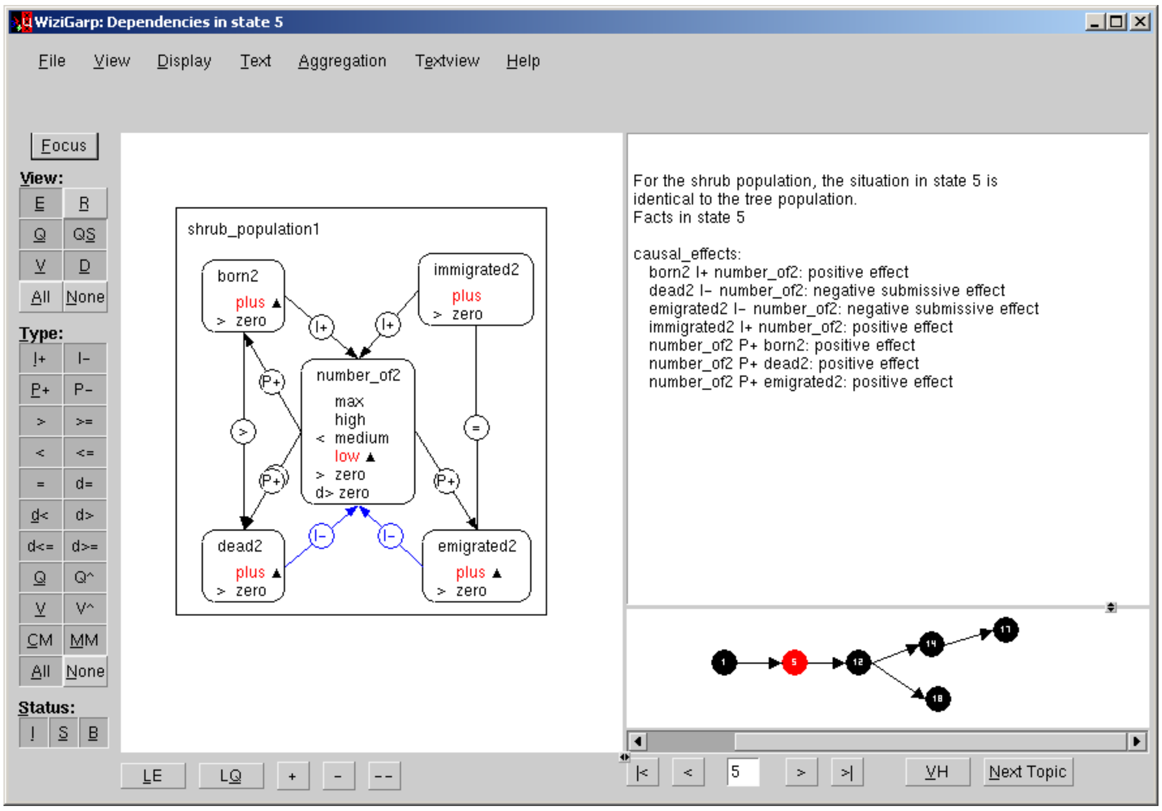
causal_effects:
  born1 I+ number_of1: positive effect
  dead1 I– number_of1: negative submissive effect
  emigrated1 I– number_of1: negative submissive effect
  immigrated1 I+ number_of1: positive effect
  number_of1 P+ born1: positive effect
  number_of1 P+ dead1: positive effect
  number_of1 P+ emigrated1: positive effect

Figure 5.7
The dependencies in the tree population.

Action 7. From now on, the topics deal with the causal and mathematical dependencies in the simulation.[1] The didactic plan specifies that the dependencies for the tree population are treated first (see figure 5.7). The intro text explains why state 5 is chosen as the time-frame.

Within the tree population, the positive effects of dead1 and emigrated1 are submissive with respect to those of born1 and immigrated1. In many cases, it is useful to hide submissive (and inactive) dependencies, as they do not contribute to, and may therefore distract attention from the actual behaviour. In this case, however, the four basic processes in population ecology are an important topic to learn about. Therefore, the status of their effects is shown in the figure and the accompanying text. The dependency hidden behind the P+ between number_of1 and dead1 is a d=-dependency stating that number_of1 and dead1 have the same derivative. This becomes visible when the P+ dependencies are turned off. The positive influence from born1 on number_of1 creates an immediate feedback loop, because the increase in number_of1 is propagated back to born1.

[1]The mathematical dependencies are shown in the figure but not discussed explicitly in order to keep the focus on the causal dependencies. The mathematical dependencies specify, among other things, that all quantities involved are greater than zero, and that born1 $>$ dead1, while immigrated1 $=$ emigrated1. These (in)equalities explain why the processes are active, and why the effects of dead1 and emigrated1 are submissive.

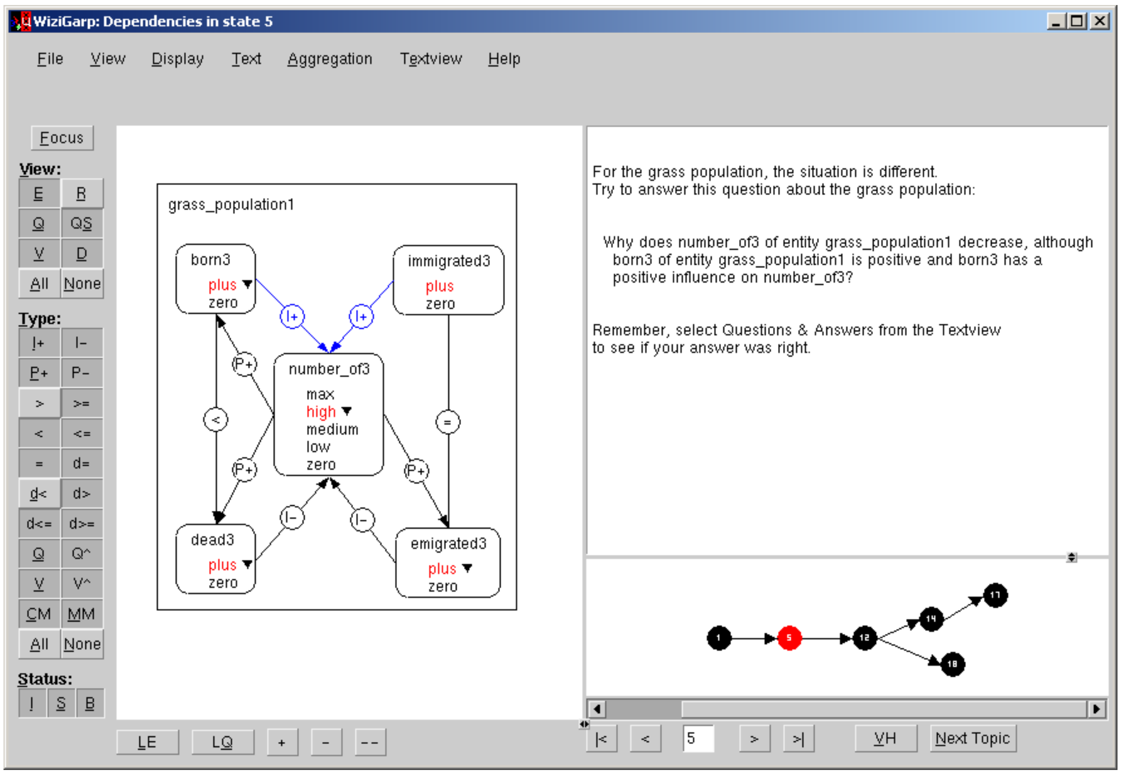


For the shrub population, the situation in state 5 is
identical to the tree population.
Facts in state 5

causal_effects:
  born2 I+ number_of2: positive effect
  dead2 I- number_of2: negative submissive effect
  emigrated2 I- number_of2: negative submissive effect
  immigrated2 I+ number_of2: positive effect
  number_of2 P+ born2: positive effect
  number_of2 P+ dead2: positive effect
  number_of2 P+ emigrated2: positive effect

Figure 5.8
The dependencies in the shrub population.

Action 8. The next topic is the shrub population, as shown in figure 5.8. The situation for the shrub population is identical to the tree population, which is explicitly mentioned in the intro text. For an expert user, this remark could perhaps be mentioned together with the previous topic, so that the shrub population would not require a separate presentation.



For the grass population, the situation is different.
Try to answer this question about the grass population:

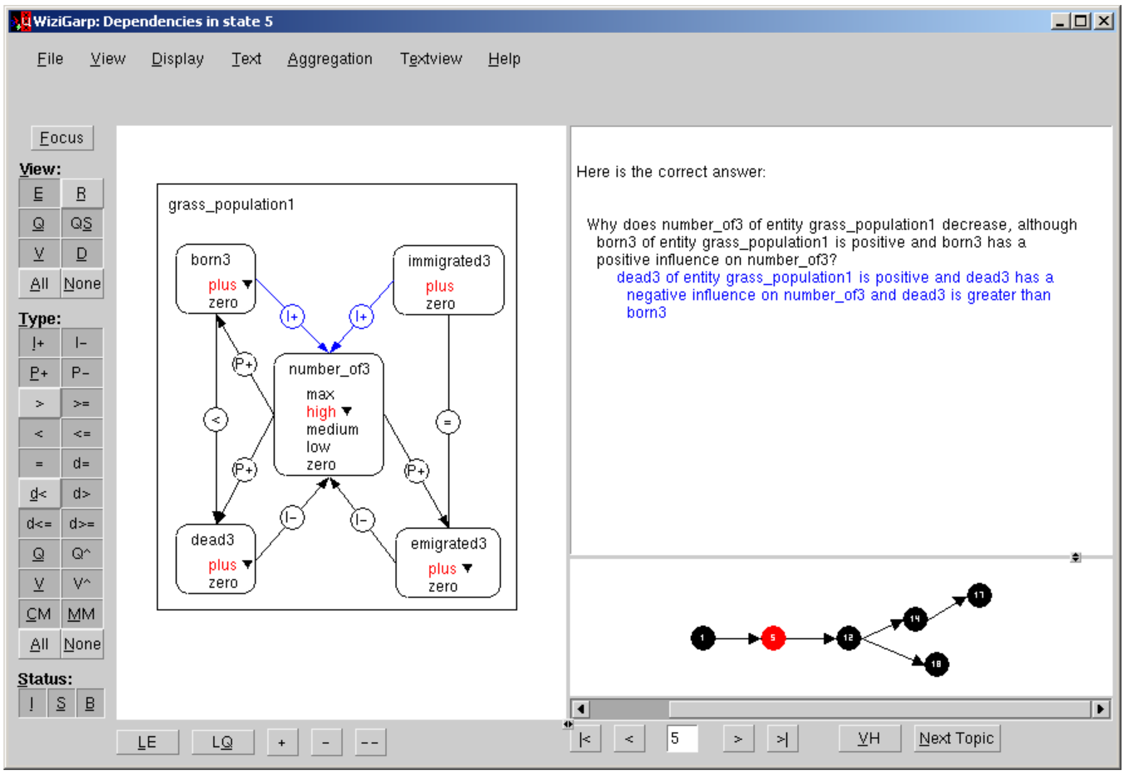
Why does number_of3 of entity grass_population1 decrease, although born3 of entity grass_population1 is positive and born3 has a positive influence on number_of3?

Remember, select Questions & Answers from the Textview to see if your answer was right.

Figure 5.9
A question about the dependencies in the grass population.

Action 9. Next, the focus is on the grass population, as shown in figure 5.9. For this population, the situation is different in that born and immigrated are submissive now, rather than dead and emigrated. But instead of showing this as facts, as before, WiziGarp's didactic plan specifies that this topic should be treated in the form of exercise questions. To this end, QUAGS is activated again. This time, several additional question criteria are necessary to limit the number of questions to the most relevant one. The Perspective criterion is set to *causal_model*, and the Behaviour criterion is set to *submissive* (for a discussion of QUAGS question criteria, see section 4.4.8). To answer the question, the learner must first realize that there are multiple processes acting on number_of3. Second, the learner should notice that, while immigrated3 is equal to emigrated3, born3 is smaller than dead3, so the negative effect of dead3 (I-) must be responsible for number_of3 decreasing.[2]

[2]The buttons for the $>$ and $d\ <$ dependencies are toggled off to hide these inequalities from view. They encode mathematical information which can be derived from other information in the state, *i.e.*, dead3 $>$ born3 can be derived from born3 $<$ dead3, and born3 $d <$ zero can be derived from born3 being decreasing.

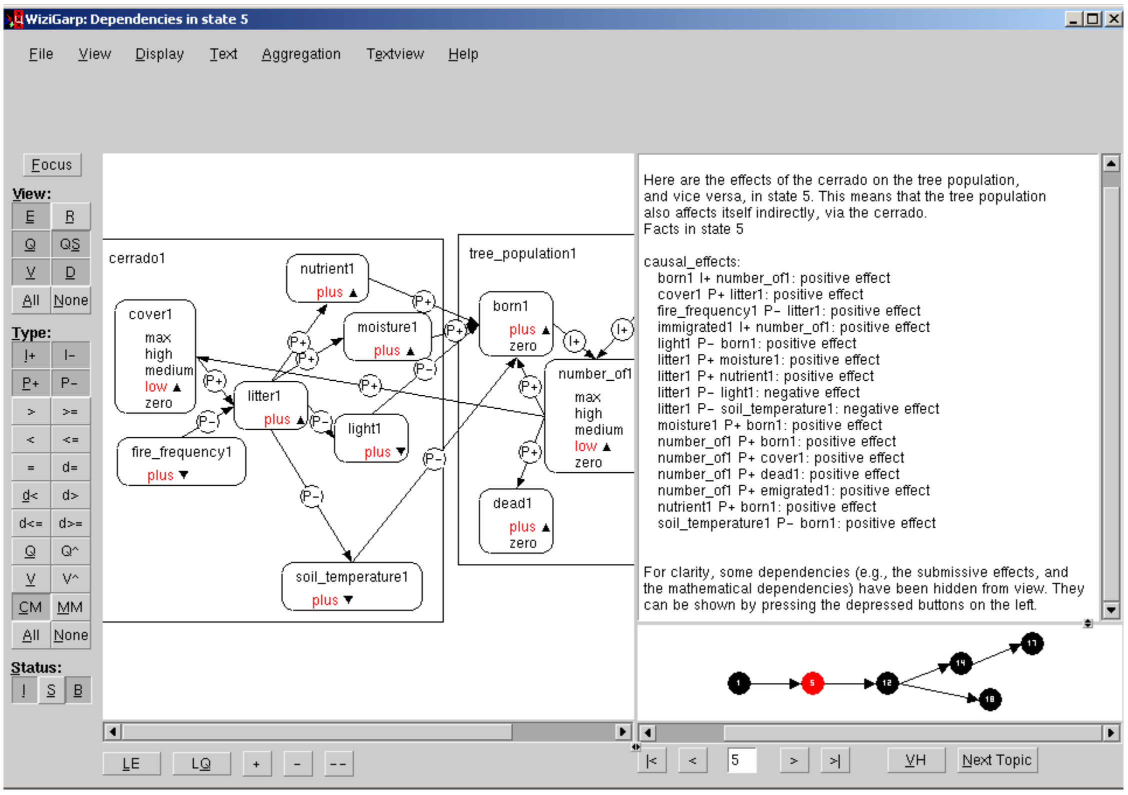


Here is the correct answer:

Why does number_of3 of entity grass_population1 decrease, although born3 of entity grass_population1 is positive and born3 has a positive influence on number_of3?
dead3 of entity grass_population1 is positive and dead3 has a negative influence on number_of3 and dead3 is greater than born3

Figure 5.10
The answer to the question about the grass population.

Action 9 (continued). This question requires some reasoning, but the learner can take the time necessary to think about it. As discussed in section 4.4.8.4, QUAGS does not currently include a way for users to enter their answers, nor to interpret them. Nevertheless, the learner can write down the answer and chooses the Question & Answers option from the Text view menu to see the correct answer, as the outro text in figure 5.9 suggests. When the learner does this, WiziGarp displays the answer in blue, indented below the question, as shown in figure 5.10.


WiziGarp: Dependencies in state 5
File View Display Text Aggregation Textview Help
Focus
View: E R Q QS V D All None
Type: I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status: I S B
cerrado1
nutrient1 plus
moisture1 plus
cover1 max high medium low zero
litter1 plus
light1 plus
fire_frequency1 plus
soil_temperature1 plus
tree_population1
born1 plus zero
number_of1 max high medium low zero
dead1 plus zero
LE LQ + - --
|< < 5 > >| VH Next Topic

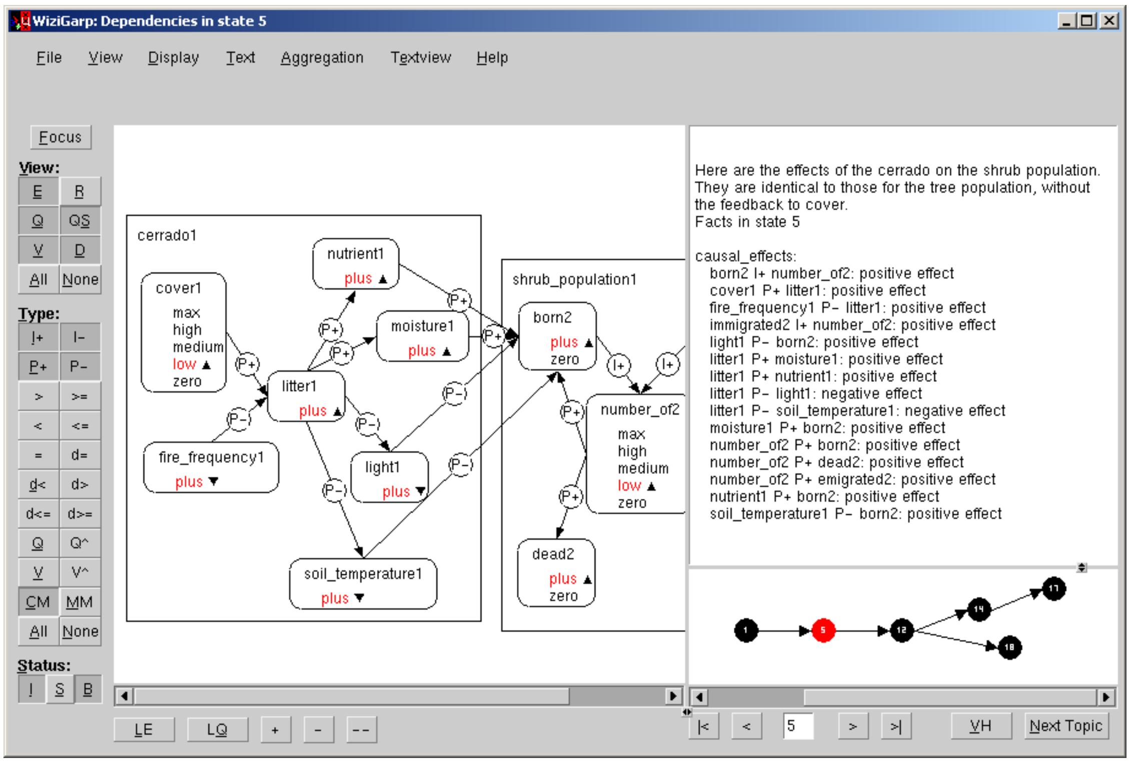
```
Here are the effects of the cerrado on the tree population,
and vice versa, in state 5. This means that the tree population
also affects itself indirectly, via the cerrado.
Facts in state 5

causal_effects:
   born1 I+ number_of1: positive effect
   cover1 P+ litter1: positive effect
   fire_frequency1 P– litter1: positive effect
   immigrated1 I+ number_of1: positive effect
   light1 P– born1: positive effect
   litter1 P+ moisture1: positive effect
   litter1 P+ nutrient1: positive effect
   litter1 P– light1: negative effect
   litter1 P– soil_temperature1: negative effect
   moisture1 P+ born1: positive effect
   number_of1 P+ born1: positive effect
   number_of1 P+ cover1: positive effect
   number_of1 P+ dead1: positive effect
   number_of1 P+ emigrated1: positive effect
   nutrient1 P+ born1: positive effect
   soil_temperature1 P– born1: positive effect


For clarity, some dependencies (e.g., the submissive effects, and
the mathematical dependencies) have been hidden from view. They
can be shown by pressing the depressed buttons on the left.
```

Figure 5.11
The interaction between the Cerrado and the tree population.

Action 10. Now, the topics become larger, and include dependencies for the combination of two entities. First, the interaction between the Cerrado and the tree population is discussed, as shown in figure 5.11. Because this involves many dependencies, the number of dependencies to show is reduced.[3] The submissive effects are hidden from view, as well as the dependencies relating to derivatives and correspondences. As the outro text mentions, they can be turned on by the user, if desired.

The figure shows how the amount of born and dead in the tree population are affected by the amount of nutrients, moisture and light, as well as the soil temperature in the Cerrado. Furthermore, as the intro text indicates, it includes a causal chain with a feedback loop from the number of trees back to the amount of cover in the Cerrado (via P+). The higher the number of trees, the higher the amount of cover (due to the leaves of the trees), and the higher the amount of litter (which includes fallen leaves). More litter means more nutrient and moisture, less light and lower soil temperature. Both the increase of nutrient and moisture and the decrease of soil temperature and light have a positive effect on born (via P+, and P- dependencies, respectively). They have a negative effect on dead, but these effects are submissive, and are therefore hidden from view. Because born is greater than dead, the number of trees is increasing.

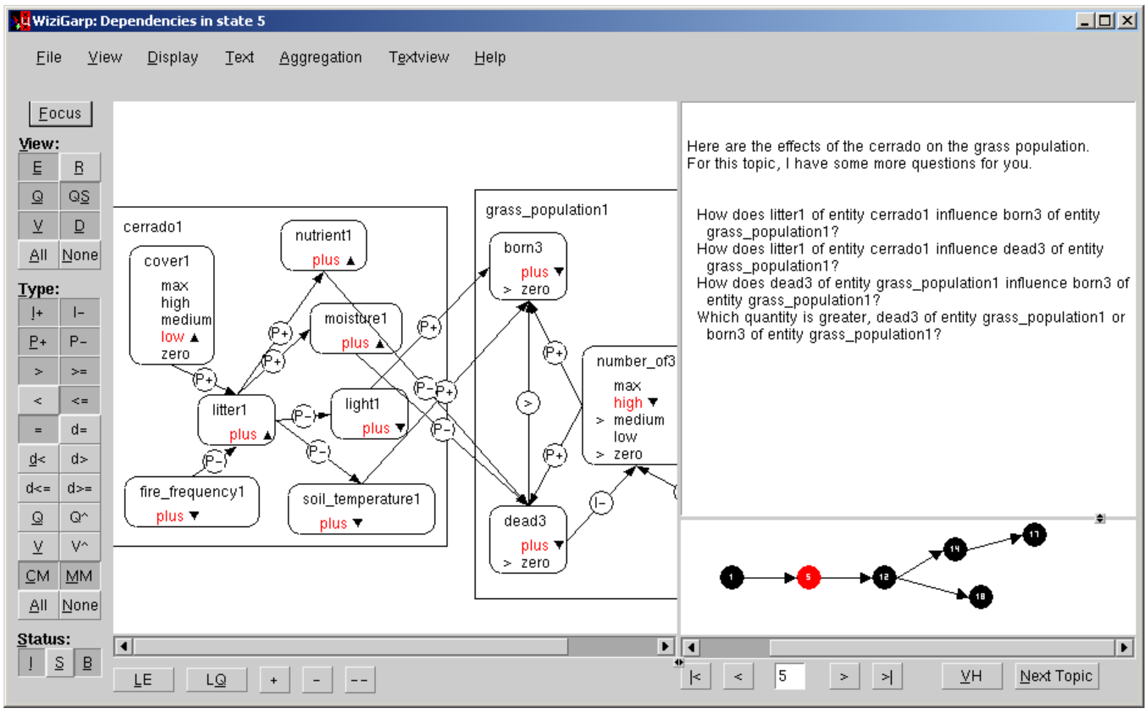
[3]In the current prototype, the didactic plan definition only allows *all* or *no* dependencies to be included in the topic; therefore, the selection of which types of dependencies to show was done manually.


WiziGarp: Dependencies in state 5
File View Display Text Aggregation Textview Help
Focus
View:
E R Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
cerrado1
cover1 max high medium low zero
nutrient1 plus
moisture1 plus
litter1 plus
fire_frequency1 plus
light1 plus
soil_temperature1 plus
shrub_population1
born2 plus zero
number_of2 max high medium low zero
dead2 plus zero
P+ P- I+
1 5 12 14 17 18
LE LQ + - --
|< < 5 > >| VH Next Topic

```
Here are the effects of the cerrado on the shrub population.
They are identical to those for the tree population, without
the feedback to cover.
Facts in state 5

causal_effects:
   born2 I+ number_of2: positive effect
   cover1 P+ litter1: positive effect
   fire_frequency1 P– litter1: positive effect
   immigrated2 I+ number_of2: positive effect
   light1 P– born2: positive effect
   litter1 P+ moisture1: positive effect
   litter1 P+ nutrient1: positive effect
   litter1 P– light1: negative effect
   litter1 P– soil_temperature1: negative effect
   moisture1 P+ born2: positive effect
   number_of2 P+ born2: positive effect
   number_of2 P+ dead2: positive effect
   number_of2 P+ emigrated2: positive effect
   nutrient1 P+ born2: positive effect
   soil_temperature1 P– born2: positive effect
```

Figure 5.12
The effects of the Cerrado on the shrub population.

Action 11. Analogously, the same is done for the shrub population (see figure 5.12). As stated in the intro text, the same dependencies apply, except for the ones that created a feedback loop from the tree population to the cover in the Cerrado. This is because shrubs do not have as many leaves, and are not as high as trees.



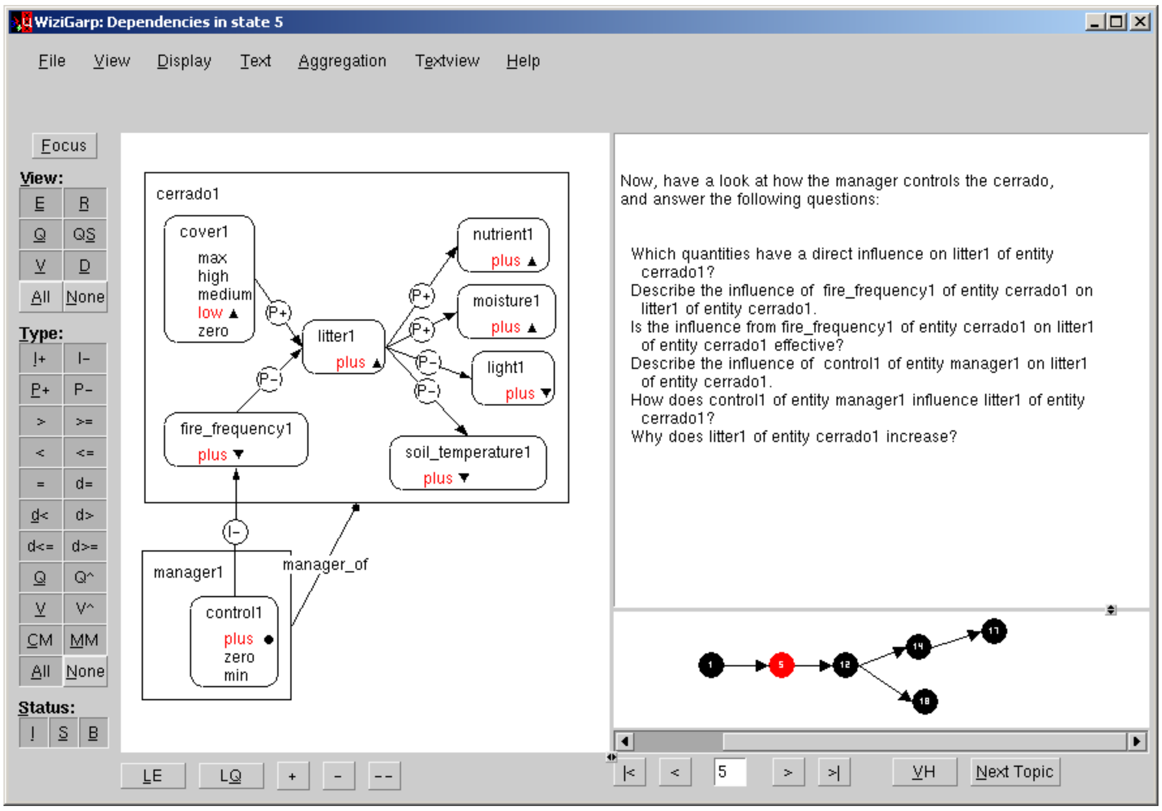
Here are the effects of the cerrado on the grass population.
For this topic, I have some more questions for you.

How does litter1 of entity cerrado1 influence born3 of entity grass_population1?
How does litter1 of entity cerrado1 influence dead3 of entity grass_population1?
How does dead3 of entity grass_population1 influence born3 of entity grass_population1?
Which quantity is greater, dead3 of entity grass_population1 or born3 of entity grass_population1?

Figure 5.13
Questions about the effects of the Cerrado on the grass population.

Action 12. The effects of the Cerrado on the grass population are slightly different (see figure 5.13). Here, the increase of nutrient and moisture have a negative effect on dead (via P- dependencies), and the decrease of soil temperature and light have a negative effect on born (via P+ dependencies). The reason for this is that grass thrives in hot and dry conditions and does not need a lot of nutrients, as shrubs and trees do. For the grass population, born is smaller than dead, which makes number_of3 decrease.

The didactic plan specifies that this topic is treated in the form of questions. When the *auto* settings are used, the question generation module QUAGS generates 37 questions about the combination of Cerrado and grass population. Since this number is too large to present to the learner, the didactic plan specifies extra criteria to fine-tune the question generation process. Setting the topic quantities to *litter1* and *dead1*, the perspective to causal model, and behaviour to 'all' results in an adequate set of four questions, a reasonable number to present to the learner at this stage. To answer some of the questions, the learner must now reason about causal chains of reasonable length. The first two questions involve chains of proportionalities. The third question involves a causal chain starting with an influence. The fourth question involves an inequality statement. The learner does not ask for the answers this time, but moves on to the next topic.

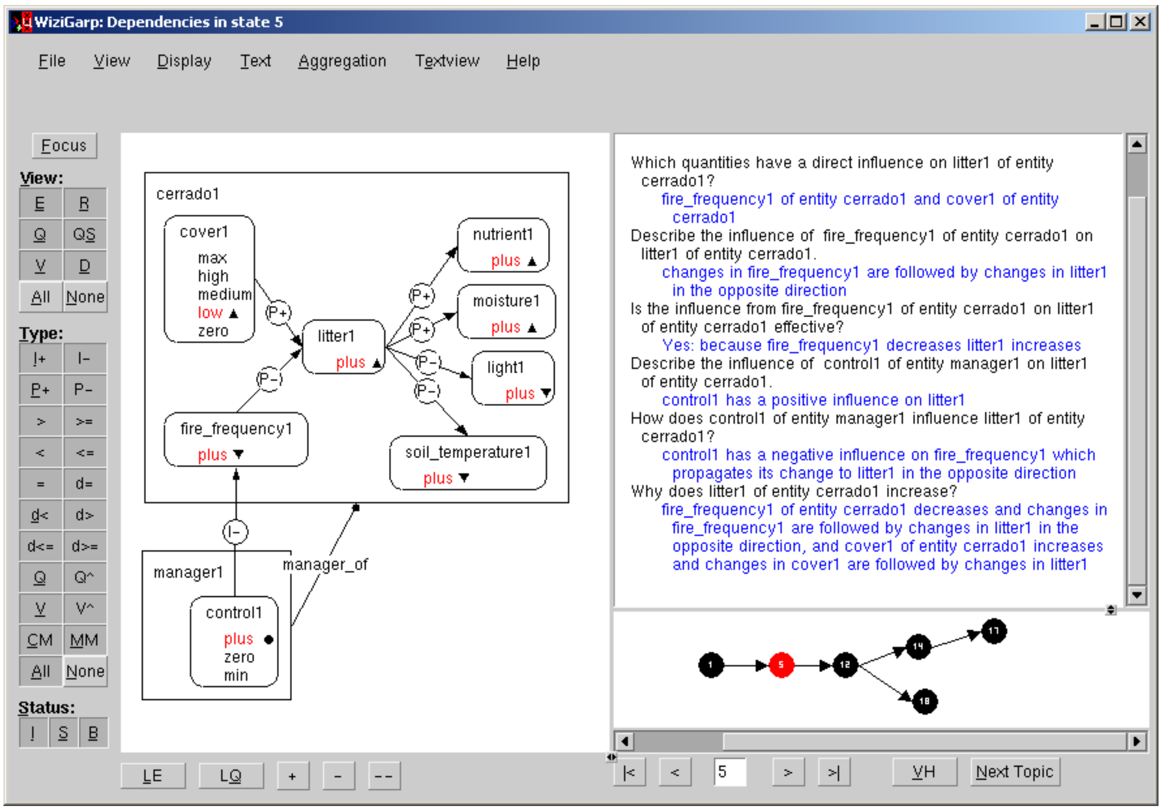


Now, have a look at how the manager controls the cerrado, and answer the following questions:

Which quantities have a direct influence on litter1 of entity cerrado1?
Describe the influence of fire_frequency1 of entity cerrado1 on litter1 of entity cerrado1.
Is the influence from fire_frequency1 of entity cerrado1 on litter1 of entity cerrado1 effective?
Describe the influence of control1 of entity manager1 on litter1 of entity cerrado1.
How does control1 of entity manager1 influence litter1 of entity cerrado1?
Why does litter1 of entity cerrado1 increase?

Figure 5.14
Questions about the influence of the manager on the Cerrado.

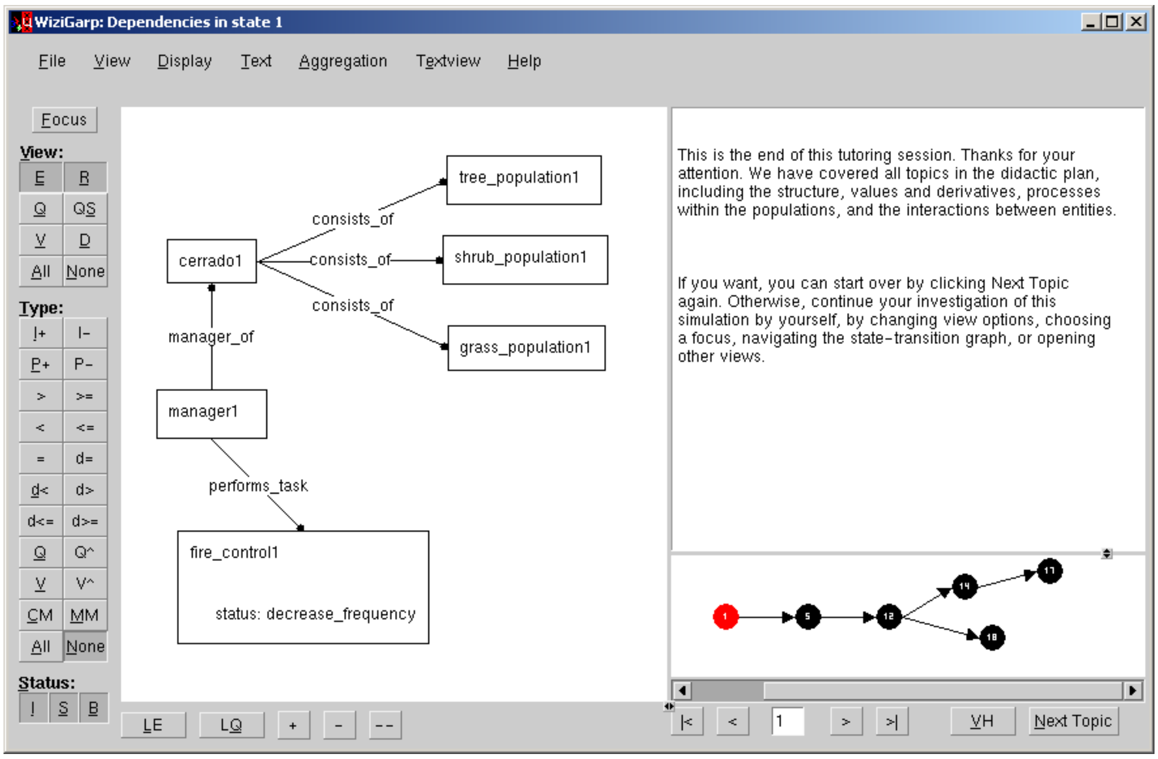
Action 13. Now, the focus changes to the Cerrado itself, plus the manager (see figure 5.14). The quantity control1 starts the causal chain with the negative influence (I-) on fire_frequency1, and propagates further from there. For this topic, QUAGS generates six relevant questions, which WiziGarp presents to the learner. All of the questions deal with the causal dependencies, but require different kinds of answers. The first, second and fourth question draw attention to how litter1 is influenced from two sides. The third question is about the status of one of these effects. The fifth question asks about the exact nature of the causal chains involved, and the sixth question requires the learner to combine some of this information to explain why litter1 increases.



Which quantities have a direct influence on litter1 of entity cerrado1?
fire_frequency1 of entity cerrado1 and cover1 of entity cerrado1
Describe the influence of fire_frequency1 of entity cerrado1 on litter1 of entity cerrado1.
changes in fire_frequency1 are followed by changes in litter1 in the opposite direction
Is the influence from fire_frequency1 of entity cerrado1 on litter1 of entity cerrado1 effective?
Yes: because fire_frequency1 decreases litter1 increases
Describe the influence of control1 of entity manager1 on litter1 of entity cerrado1.
control1 has a positive influence on litter1
How does control1 of entity manager1 influence litter1 of entity cerrado1?
control1 has a negative influence on fire_frequency1 which propagates its change to litter1 in the opposite direction
Why does litter1 of entity cerrado1 increase?
fire_frequency1 of entity cerrado1 decreases and changes in fire_frequency1 are followed by changes in litter1 in the opposite direction, and cover1 of entity cerrado1 increases and changes in cover1 are followed by changes in litter1

Figure 5.15
The answers to the questions about the influence of the manager on the Cerrado.

Action 13 (continued). The learner asks for the answers to the questions, which are shown in figure 5.15. Answering these questions requires reasoning about the status and direction of the effects, *e.g.*, a negative influence (I-) propagated in the opposite direction (P-) results in a net positive effect. The phrase ’in the opposite direction’ is used to draw extra attention to negative effects, as they are often harder to understand. For example, the quantity control1, which has the value plus, has a negative effect on fire_frequency1 (I-), causing it to decrease. This effect is propagated in the opposite direction (P-) to litter1, causing litter1 to increase. Litter1 is also directly affected by cover1. Cover1 is increasing, so this dependency (P+) also has a positive effect on litter1.



This is the end of this tutoring session. Thanks for your attention. We have covered all topics in the didactic plan, including the structure, values and derivatives, processes within the populations, and the interactions between entities.

If you want, you can start over by clicking Next Topic again. Otherwise, continue your investigation of this simulation by yourself, by changing view options, choosing a focus, navigating the state-transition graph, or opening other views.

Figure 5.16
The end of the scenario.

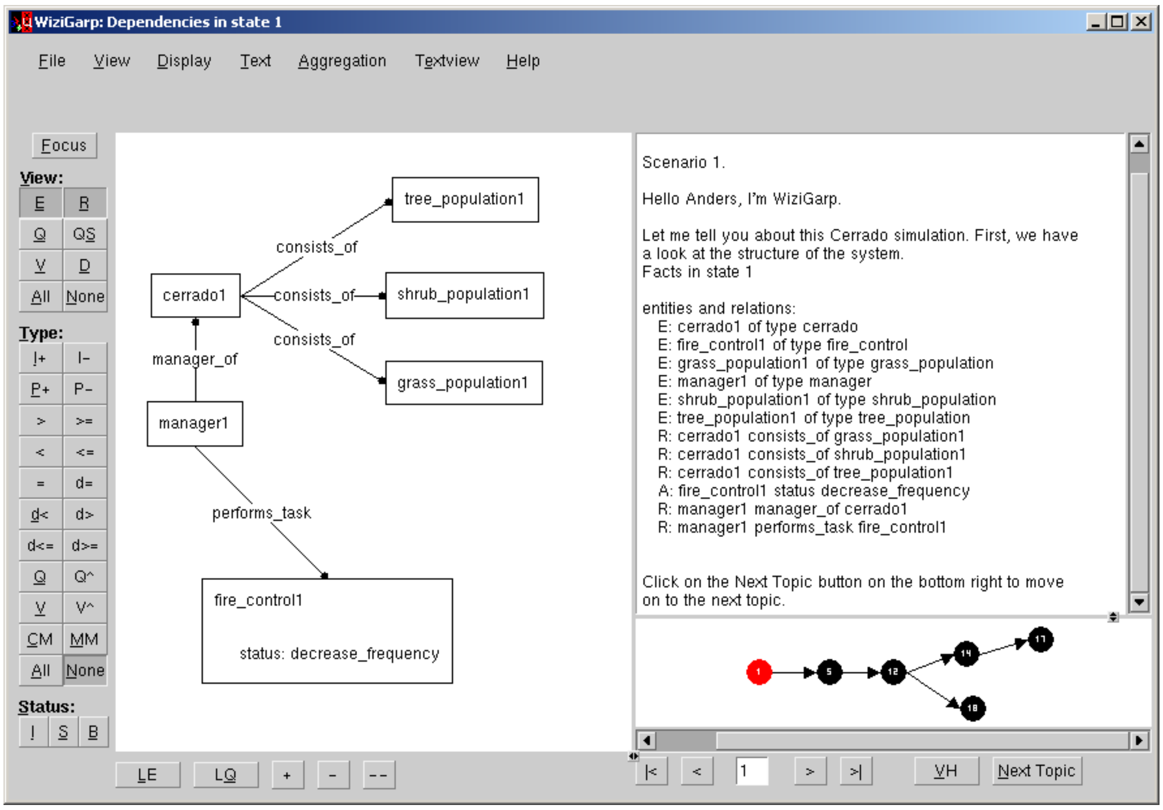
Action 14. After all pairwise interactions between entities have been discussed, the scenario now reaches the end of the didactic plan, as shown in figure 5.16. WiziGarp's intro text thanks the user for the effort and gives a summary of the topics covered. The outro text offers the learner the initiative, by presenting an overview of interface options available in WiziGarp.

This scenario has acquainted the learner with the structure of the Cerrado, the behaviour of the main quantities over the course of the simulation, the four basic processes in population ecology (birth, death, immigration and emigration), and how their effect may differ based on (in)equalities between the quantities involved, and the interaction effects between pairs of entities. However, scenario 2 demonstrates that there are even more topics to explore in the Cerrado simulation.

5.3 Scenario 2: Student-initiated exploration

The second scenario simulates the interaction between WiziGarp and an advanced learner, who already knows something about population ecology, and also knows how to use WiziGarp. In this scenario, the learner explores the simulation by navigating through the visualizations and by asking for explanations, without interference by WiziGarp.

To show the flexibility of WiziGarp, the scenario starts off the same way as in scenario 1 (as shown in figure 5.17). However, after the first screen the didactic plan is not used anymore, since the learner does not click on Next Topic, but follows his own plan.


WiziGarp: Dependencies in state 1
File View Display Text Aggregation Textview Help
Focus
View:
E R Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
tree_population1
consists_of
cerrado1
consists_of
shrub_population1
consists_of
grass_population1
manager_of
manager1
performs_task
fire_control1
status: decrease_frequency
LE LQ + - --
|< < 1 > >| VH Next Topic

Scenario 1.

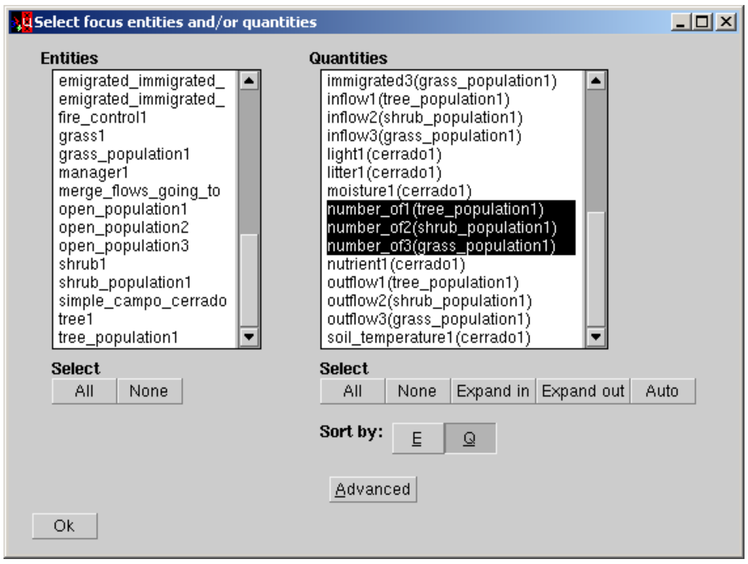
Hello Anders, I'm WiziGarp.

Let me tell you about this Cerrado simulation. First, we have a look at the structure of the system.
Facts in state 1

entities and relations:
E: cerrado1 of type cerrado
E: fire_control1 of type fire_control
E: grass_population1 of type grass_population
E: manager1 of type manager
E: shrub_population1 of type shrub_population
E: tree_population1 of type tree_population
R: cerrado1 consists_of grass_population1
R: cerrado1 consists_of shrub_population1
R: cerrado1 consists_of tree_population1
A: fire_control1 status decrease_frequency
R: manager1 manager_of cerrado1
R: manager1 performs_task fire_control1

Click on the Next Topic button on the bottom right to move on to the next topic.

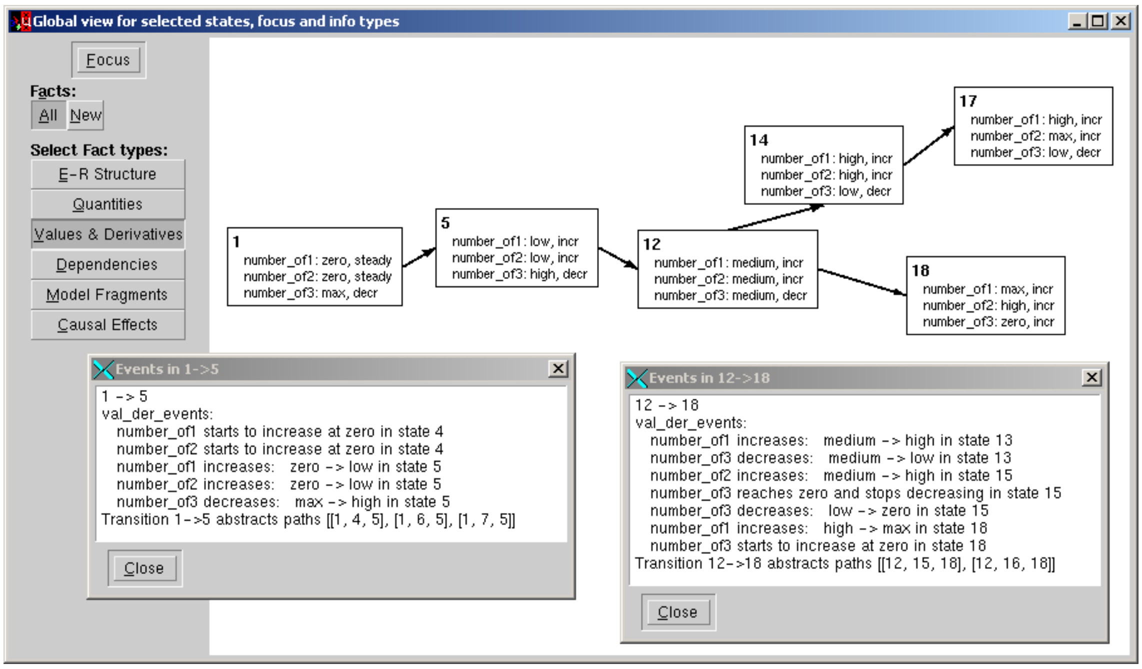
Figure 5.17
Scenario 2 starts off the same way as scenario 1.



```
Entities                     Quantities
emigrated_immigrated_        immigrated3(grass_population1)
emigrated_immigrated_        inflow1(tree_population1)
fire_control1                inflow2(shrub_population1)
grass1                       inflow3(grass_population1)
grass_population1            light1(cerrado1)
manager1                     litter1(cerrado1)
merge_flows_going_to         moisture1(cerrado1)
open_population1             number_of1(tree_population1)
open_population2             number_of2(shrub_population1)
open_population3             number_of3(grass_population1)
shrub1                       nutrient1(cerrado1)
shrub_population1            outflow1(tree_population1)
simple_campo_cerrado         outflow2(shrub_population1)
tree1                        outflow3(grass_population1)
tree_population1             soil_temperature1(cerrado1)
```

Figure 5.18
The learner selects three focus quantities.

The learner clicks on the All button in the view group on the left of the diagram to show all quantities and their quantity spaces, values and derivatives (not shown). Because there is so much information that the diagram can not show this adequately, the learner clicks on the Focus button. This causes the focus selection window to pop up (shown in figure 5.18). To see what WiziGarp suggests as interesting quantities, the learner clicks on the Auto button. WiziGarp automatically selects number_of1, number_of2, number_of3, born3, and dead3, based on the fact that these quantities are involved in the higest number of events in the simulation (as described in section 4.4.1.2). However, the learner decides to focus on a subset of these quantities, *i.e.*. number_of1, number_of2, and number_of3, and deselects the other two. This is the situation as shown in the figure.


Global view for selected states, focus and info types
Focus
Facts:
All
New
Select Fact types:
E-R Structure
Quantities
Values & Derivatives
Dependencies
Model Fragments
Causal Effects
1 number_of1 : zero, steady number_of2: zero, steady number_of3: max, decr
5 number_of1 : low, incr number_of2: low, incr number_of3: high, decr
12 number_of1 : medium, incr number_of2: medium, incr number_of3: medium, decr
14 number_of1 : high, incr number_of2: high, incr number_of3: low, decr
17 number_of1 : high, incr number_of2: max, incr number_of3: low, decr
18 number_of1 : max, incr number_of2: high, incr number_of3: zero, incr
Events in 1->5
Close
Events in 12->18
Close

```
1 -> 5
val_der_events:
  number_of1 starts to increase at zero in state 4
  number_of2 starts to increase at zero in state 4
  number_of1 increases:  zero -> low in state 5
  number_of2 increases:  zero -> low in state 5
  number_of3 decreases:  max -> high in state 5
Transition 1->5 abstracts paths [[1, 4, 5], [1, 6, 5], [1, 7, 5]]
```

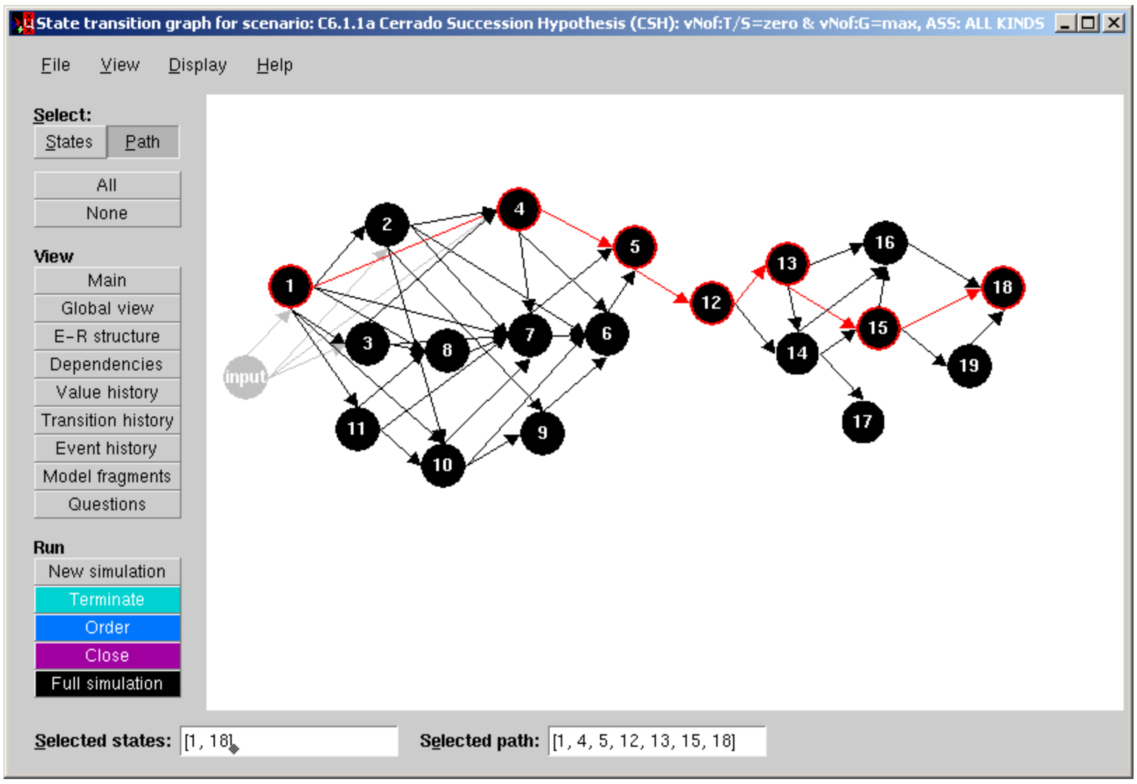
```
12 -> 18
val_der_events:
  number_of1 increases:  medium -> high in state 13
  number_of3 decreases:  medium -> low in state 13
  number_of2 increases:  medium -> high in state 15
  number_of3 reaches zero and stops decreasing in state 15
  number_of3 decreases:  low -> zero in state 15
  number_of1 increases:  high -> max in state 18
  number_of3 starts to increase at zero in state 18
Transition 12->18 abstracts paths [[12, 15, 18], [12, 16, 18]]
```

Figure 5.19
The global view showing the values and derivatives of the three focus quantities.

After clicking OK, which closes the Focus selection window, the learner opens the global view to get an overview of the behaviour of these quantities (see figure 5.19).

The learner notices that number_of1 and number_of2 increase over the course of the simulation, while number_of3 decreases, with two different possibilities towards the end. The learner tries to spot where in the state-transition graph the most interesting things happen. To this end, the learner inspects all transitions by double-clicking them. This causes small event windows to pop up which list the value and derivative events which occur in that particular transition. Since the state-transition graph is aggregated, the event window shows which paths are abstracted into each abstracted transition. After inspecting them all, two transitions seem most interesting to the learner: 1→5, because in this transition, two population sizes start to increase, and 12→18, because here number_of3 stops decreasing and starts increasing again.[4]

[4]If the functionality of contrastive explanations for paths (as described in section 4.4.5.2) would have been fully integrated into the graphical interface of WiziGarp, the learner could have utilized it here to find out how the behaviour from state 12 to 17 differs from the behaviour from state 12 to 18. This would also have led to the conclusion that the most interesting events happen in the selected aggregated transition from 12 to 18.



Selected states: [1, 18]

Selected path: [1, 4, 5, 12, 13, 15, 18]

Figure 5.20
The original state-transition graph.

Because both of these transitions are aggregated transitions, the learner decides to deaggregate the state-transition graph to see the original state-transition graph.

The result is shown in figure 5.20. There are many states and transitions between 1 and 5, and between 12 and 18. It would take too long to investigate them all, so the learner decides to look at one particular path from start to end. He selects state 1 and 18 and clicks the Path button, which causes the path [1,4,5,12,13,15,18] to be selected automatically (and consequently, to be highlighted in red). Since this path includes both a route from 1 to 5, and from 12 to 18, the learner is satisfied with this selected time-frame.

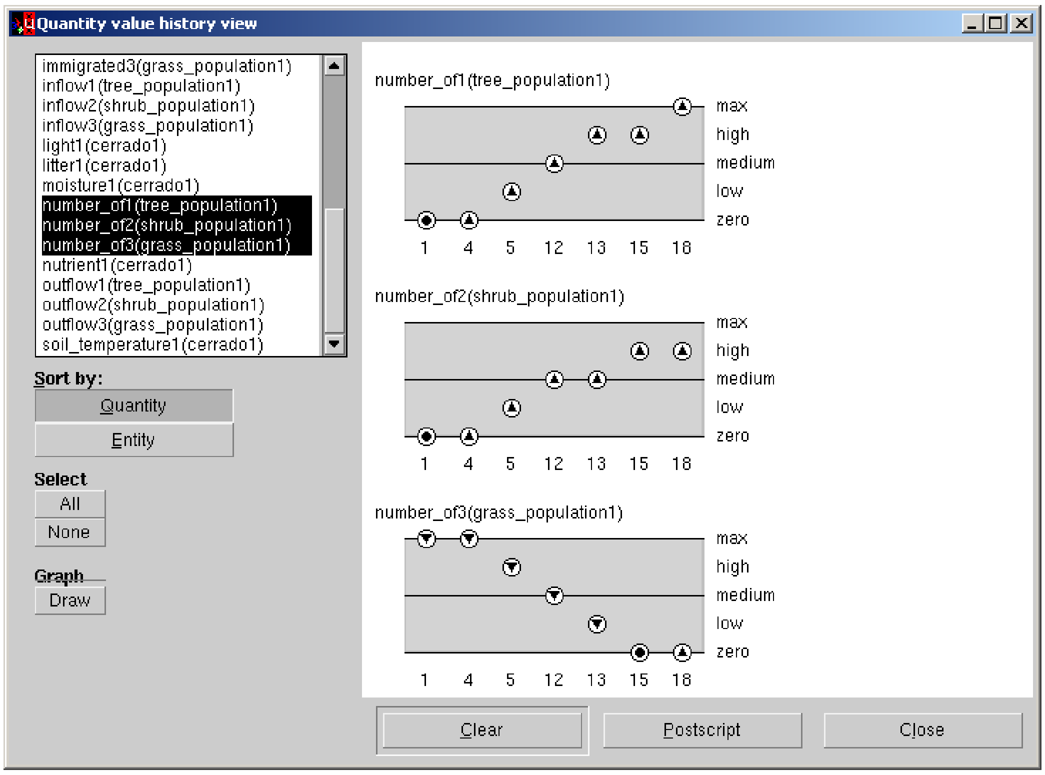
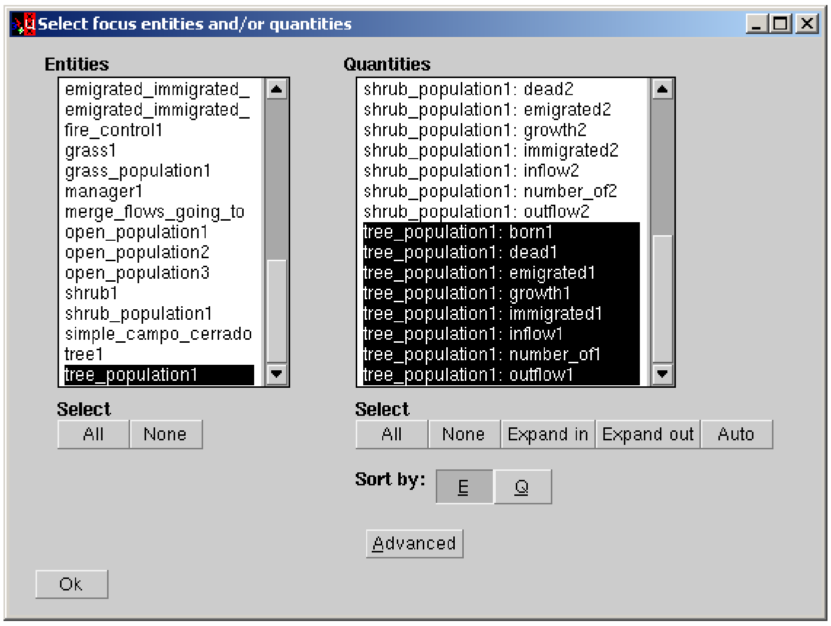

Quantity value history view
immigrated3(grass_population1)
inflow1(tree_population1)
inflow2(shrub_population1)
inflow3(grass_population1)
light1(cerrado1)
litter1(cerrado1)
moisture1(cerrado1)
number_of1(tree_population1)
number_of2(shrub_population1)
number_of3(grass_population1)
nutrient1(cerrado1)
outflow1(tree_population1)
outflow2(shrub_population1)
outflow3(grass_population1)
soil_temperature1(cerrado1)
Sort by:
Quantity
Entity
Select
All
None
Graph
Draw
number_of1(tree_population1)
max
high
medium
low
zero
1 4 5 12 13 15 18
number_of2(shrub_population1)
max
high
medium
low
zero
1 4 5 12 13 15 18
number_of3(grass_population1)
max
high
medium
low
zero
1 4 5 12 13 15 18
Clear
Postscript
Close

Figure 5.21
The value history for the three focus quantities.

To see exactly what happens in this path, the learner opens the value history view. This shows the values for the still selected focus quantities along the path from 1 to 18, as shown in figure 5.21. It presents a more detailed view than figure 5.19, because several states (4, 13, and 15) that were abstracted before are now included.[5]

[5]The reader who is familiar with QR might object against the simulated behaviour of the quantity number_of3 along states 1 and 4, and the quantity number_of2 along states 12 and 13, since these quantities are changing while staying at the same point value (max, and medium, respectively). This behaviour is due to adaptation of the transition rule set for this domain, which relaxes the constraints about point values, essentially making them equivalent to interval values.



Select focus entities and/or quantities

Entities

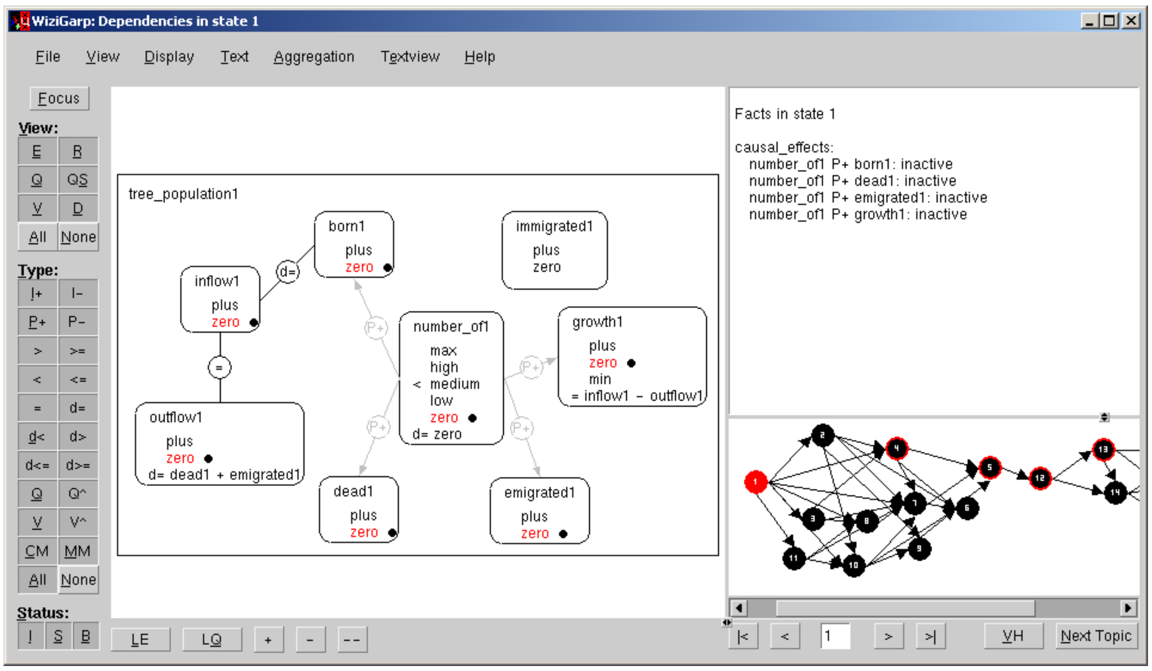
```
emigrated_immigrated_
emigrated_immigrated_
fire_control1
grass1
grass_population1
manager1
merge_flows_going_to
open_population1
open_population2
open_population3
shrub1
shrub_population1
simple_campo_cerrado
tree1
tree_population1
```

Quantities

```
shrub_population1: dead2
shrub_population1: emigrated2
shrub_population1: growth2
shrub_population1: immigrated2
shrub_population1: inflow2
shrub_population1: number_of2
shrub_population1: outflow2
tree_population1: born1
tree_population1: dead1
tree_population1: emigrated1
tree_population1: growth1
tree_population1: immigrated1
tree_population1: inflow1
tree_population1: number_of1
tree_population1: outflow1
```

Figure 5.22
The learner selects the tree population as the focus.

Now that the learner has seen a global overview of what happens to the three populations, he decides to investigate the tree population in detail. Therefore, the learner returns to the main screen, clicks the Focus button, and selects the tree population to focus on. This causes all its quantities to be included automatically, as shown in figure 5.22. The result is shown in figure 5.23.


WiziGarp: Dependencies in state 1
File View Display Text Aggregation Textview Help
Focus
View:
E R
Q QS
V D
All None
Type:
I+ I–
P+ P–
> >=
< <=
= d=
d< d>
d<= d>=
Q Q^
V V^
CM MM
All None
Status:
I S B
LE LQ + - --
tree_population1
born1 plus zero
immigrated1 plus zero
inflow1 plus zero
d=
number_of1 max high < medium low zero d= zero
growth1 plus zero min = inflow1 – outflow1
P+
=
outflow1 plus zero d= dead1 + emigrated1
dead1 plus zero
emigrated1 plus zero
|< < 1 > >| VH Next Topic

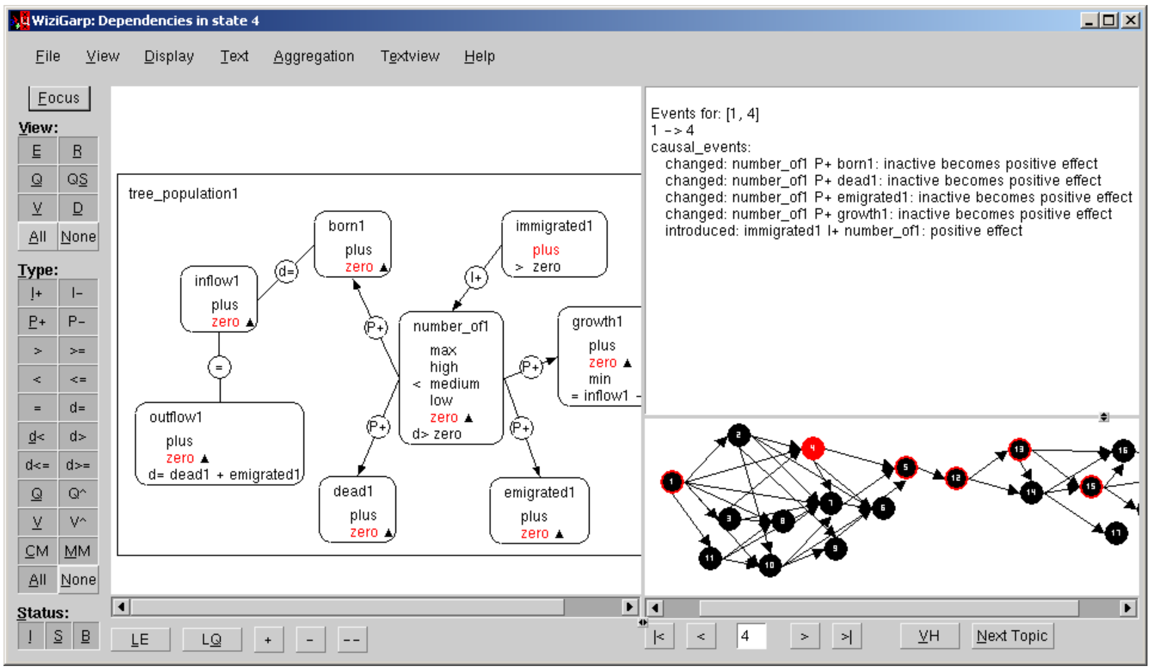
```
Facts in state 1

causal_effects:
   number_of1 P+ born1: inactive
   number_of1 P+ dead1: inactive
   number_of1 P+ emigrated1: inactive
   number_of1 P+ growth1: inactive
```

Figure 5.23
The dependencies within the tree population in state 1.

In figure 5.23, *all* quantities in the tree population are shown, not just a selection, as in scenario 1. The additional quantities, inflow, outflow, and growth are defined in terms of the other quantities.

In the miniature state-transition graph, the states of the selected path are preselected, as explained in section 4.4.1.1 and the text accompanying figure 5.20. This allows the user to easily navigate through the path by clicking the $>$ button. To focus on the causal effects, the learner selects *causal effects* from the Aggregation menu as the type of information of interest (this selection process is not shown in the figure). The text view setting is *auto*, by default. Because this is the first state that is presented, WiziGarp shows the facts for this state. In this state 1, all causal dependencies are inactive (notice the grey P+ dependencies), and all quantities are steady.

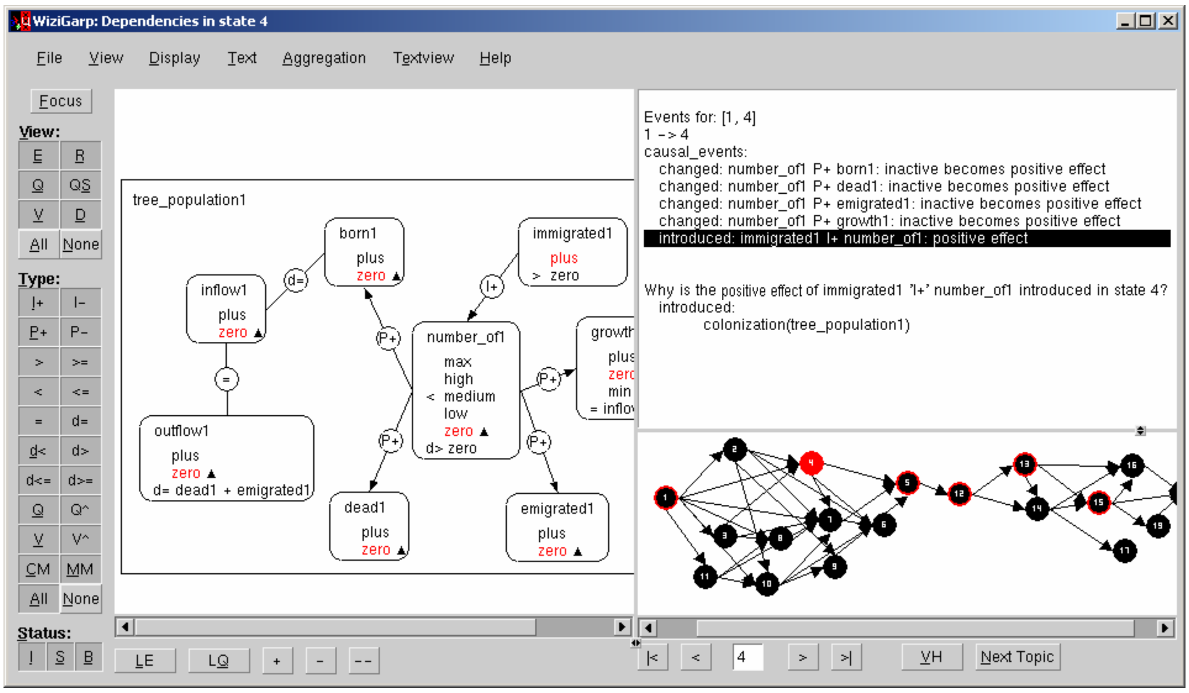
The (in)equalities number_of1 $< medium$ and $d = zero$ may seem superfluous, because its value and derivative are both known (and shown) to be zero. The fact that these (in)equalities are shown explicitly signifies to the expert user that they play a role in GARP's reasoning process, *e.g.*, as a condition for a particular model fragment.


WiziGarp: Dependencies in state 4
File View Display Text Aggregation Textview Help
Focus
View:
Type:
Status:
tree_population1
inflow1 plus zero
born1 plus zero
immigrated1 plus > zero
number_of1 max high < medium low zero d> zero
growth1 plus zero min = inflow1
outflow1 plus zero d= dead1 + emigrated1
dead1 plus zero
emigrated1 plus zero
LE LQ + - --
VH Next Topic

```
Events for: [1, 4]
1 -> 4
causal_events:
   changed: number_of1 P+ born1: inactive becomes positive effect
   changed: number_of1 P+ dead1: inactive becomes positive effect
   changed: number_of1 P+ emigrated1: inactive becomes positive effect
   changed: number_of1 P+ growth1: inactive becomes positive effect
   introduced: immigrated1 I+ number_of1: positive effect
```

Figure 5.24
The dependencies within the tree population in state 4.

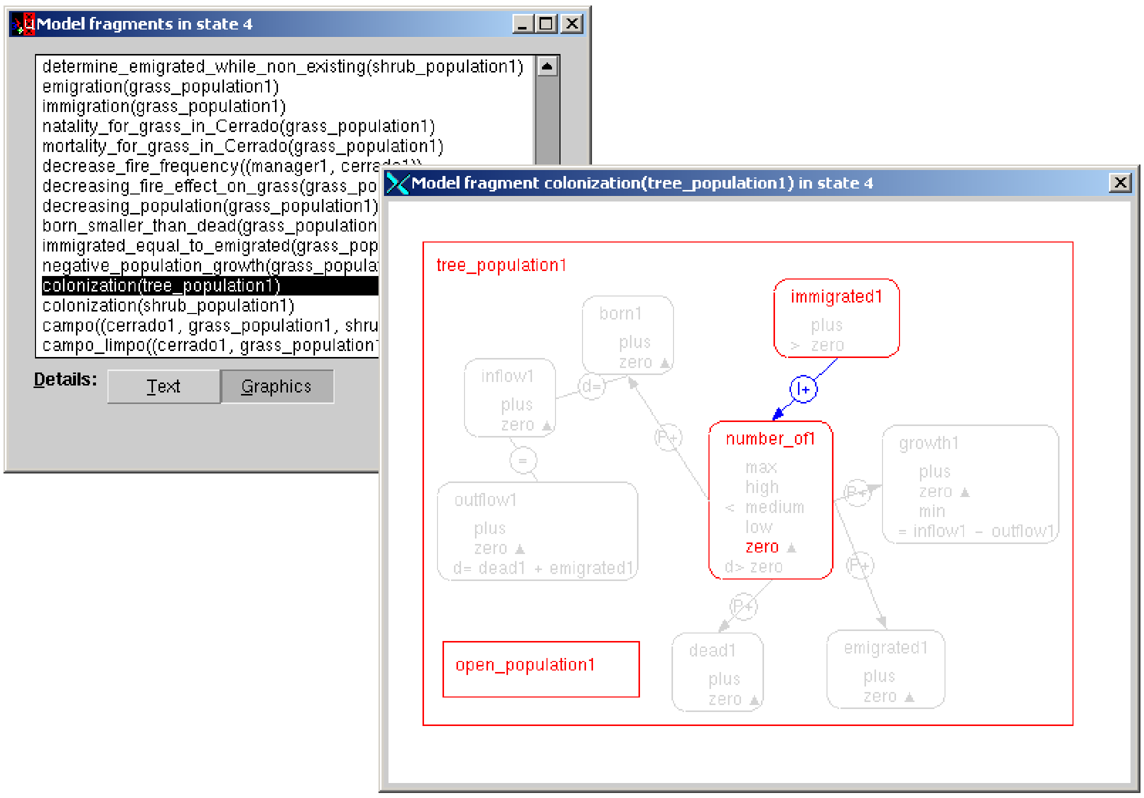
The learner uses the > button to step forward to the next state, which is state 4. The result after WiziGarp has updated the screen is shown in figure 5.24. Now, most quantities have started to increase. Because the text view setting is *auto*, WiziGarp shows the difference between state 1 and 4 in terms of local events (as defined in section 3.5.2). All inactive P+ dependencies now have changed into having a positive effect, and an influence is introduced: immigrated1 I+ number_of1. Due to the way the model is built, initially, inflow1 is zero, and immigrated1 is regarded as an exogenous variable.


WiziGarp: Dependencies in state 4
File View Display Text Aggregation Textview Help
Focus
View:
E R Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
! S B
LE LQ + - --
tree_population1
inflow1 plus zero
born1 plus zero
immigrated1 plus > zero
number_of1 max high < medium low zero d> zero
growth plus zero min = inflow
outflow1 plus zero d= dead1 + emigrated1
dead1 plus zero
emigrated1 plus zero
d= = I+ P+ P+ P+ P+
|< < 4 > >| VH Next Topic

```
Events for: [1 , 4]
1 -> 4
causal_events:
   changed: number_of1 P+ born1: inactive becomes positive effect
   changed: number_of1 P+ dead1: inactive becomes positive effect
   changed: number_of1 P+ emigrated1: inactive becomes positive effect
   changed: number_of1 P+ growth1: inactive becomes positive effect
   introduced: immigrated1 I+ number_of1: positive effect

Why is the positive effect of immigrated1 'I+' number_of1 introduced in state 4?
   introduced:
            colonization(tree_population1)
```

Figure 5.25
The learner asks a query about the tree population in state 4.

The learner understands that this influence causes number_of1 to start increasing and that this effect propagates to the quantities born1, dead1, and emigrated1; because the population size increases, the amount of births, deaths and emigrations also increases. However, the learner wants to know where this influence comes from. Therefore, the learner clicks on this event, and selects the why-question from the question menu that pops up. The question and the answer given by WiziGarp are shown in figure 5.25. WiziGarp determines the answer by matching rules that might explain the introduction of this influence with the events that occur. It finds the rule which explains the event by a model fragment introduction event: the model fragment colonization, which has just been introduced for the tree population, includes the dependency immigrated1 I+ number_of1 in its results. This explains the event as desired.


Model fragments in state 4

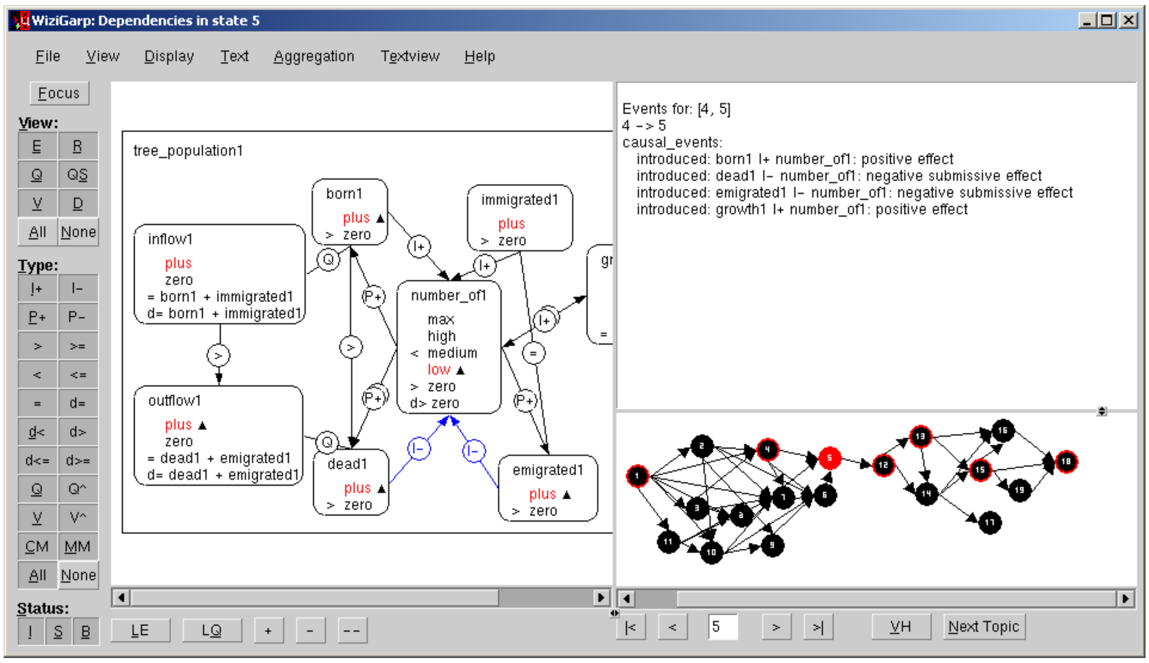
determine_emigrated_while_non_existing(shrub_population1)
emigration(grass_population1)
immigration(grass_population1)
natality_for_grass_in_Cerrado(grass_population1)
mortality_for_grass_in_Cerrado(grass_population1)
decrease_fire_frequency((manager1, cerra
decreasing_fire_effect_on_grass(grass_po
decreasing_population(grass_population1)
born_smaller_than_dead(grass_population
immigrated_equal_to_emigrated(grass_pop
negative_population_growth(grass_popula
colonization(tree_population1)
colonization(shrub_population1)
campo((cerrado1, grass_population1, shru
campo_limpo((cerrado1, grass_population

Model fragment colonization(tree_population1) in state 4

Figure 5.26
The list of model fragments for state 4 and the contents of the colonization model fragment.

To learn more about the model fragment *colonization*, the learner opens the model fragments list view from the view menu, showing the model fragments active in the current state 4. From the list, the learner selects the colonization model fragment for the tree population and opens the graphics detail view to have a look at the contents of this model fragment (shown in figure 5.26). The conditions (which are shown in red in the interface) include an assumption entity, open_population1, and a value statement: number_of1 = zero. The results (shown in blue) only contain the influence from immigrated1 on number_of1. In other words, if the assumption holds that tree_population1 is an open population, and its population size (number_of1) is zero, then the colonization process may become active, introducing a positive influence from the quantity immigrated1.

This way of inspecting the model fragments which are relevant to the situation at hand provides a link from learning about the actual behaviour to the generic domain knowledge in the domain library.


WiziGarp: Dependencies in state 5
File View Display Text Aggregation Textview Help
Focus
View:
E R Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
LE LQ + - --
tree_population1
born1 plus > zero
immigrated1 plus > zero
inflow1 plus zero = born1 + immigrated1 d= born1 + immigrated1
number_of1 max high < medium low > zero d> zero
outflow1 plus zero = dead1 + emigrated1 d= dead1 + emigrated1
dead1 plus > zero
emigrated1 plus > zero
|< < 5 > >| VH Next Topic

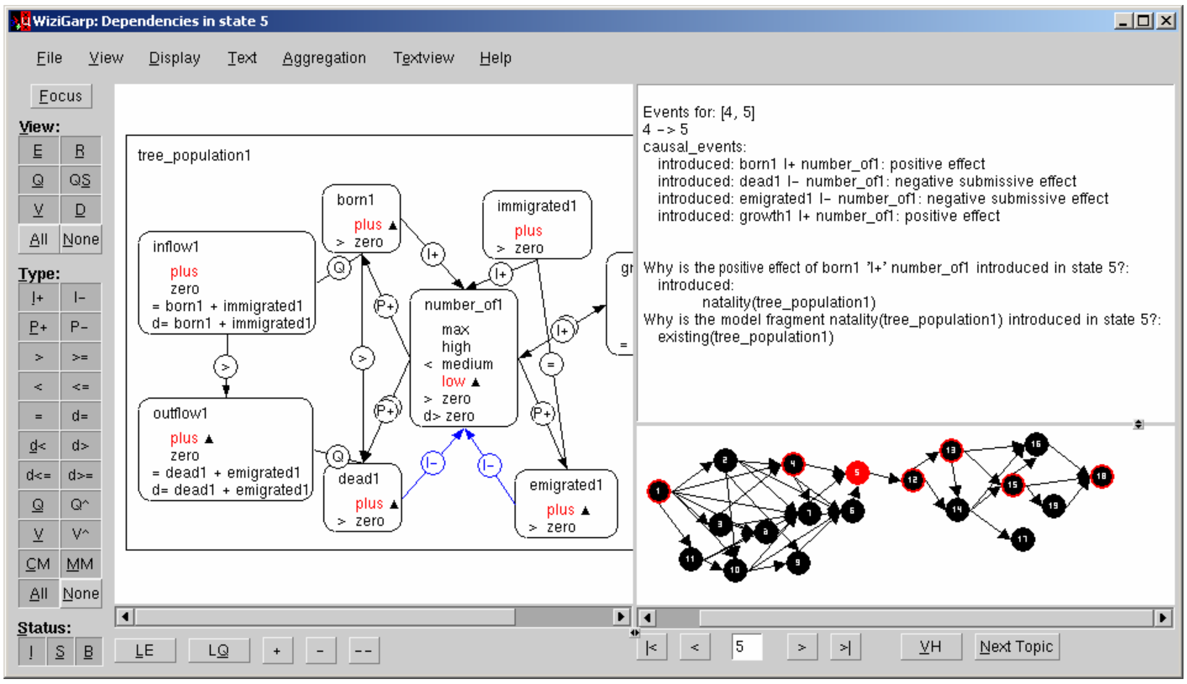
Events for: [4, 5]
4 -> 5
causal_events:
introduced: born1 I+ number_of1: positive effect
introduced: dead1 I- number_of1: negative submissive effect
introduced: emigrated1 I- number_of1: negative submissive effect
introduced: growth1 I+ number_of1: positive effect

Figure 5.27
The dependencies within the tree population in state 5.

Next, the learner closes the model fragment views and continues along the path to state 5 (see figure 5.27). In the transition 4$\rightarrow$5, four influences are introduced, of which two are submissive (the I- from dead1 and emigrated1) with respect to the three I+ dependencies (the I+ from born1 and growth1, and the already present immigrated1). The (in)equalities introduced give some insight into why some dependencies are submissive.

Now that the population exists (i.e., number_of1 $>$ zero), inflow1 is defined as born1 + immigrated1, and outflow1 as dead1 + emigrated1. Inflow is no longer equal to, but has become larger than outflow. Because in this simulation, immigrated1 is assumed to be equal to emigrated1, the values of inflow1 and outflow1 are in fact determined by born1 and dead1, as modelled by the quantity-correspondences (Q) in the diagram.

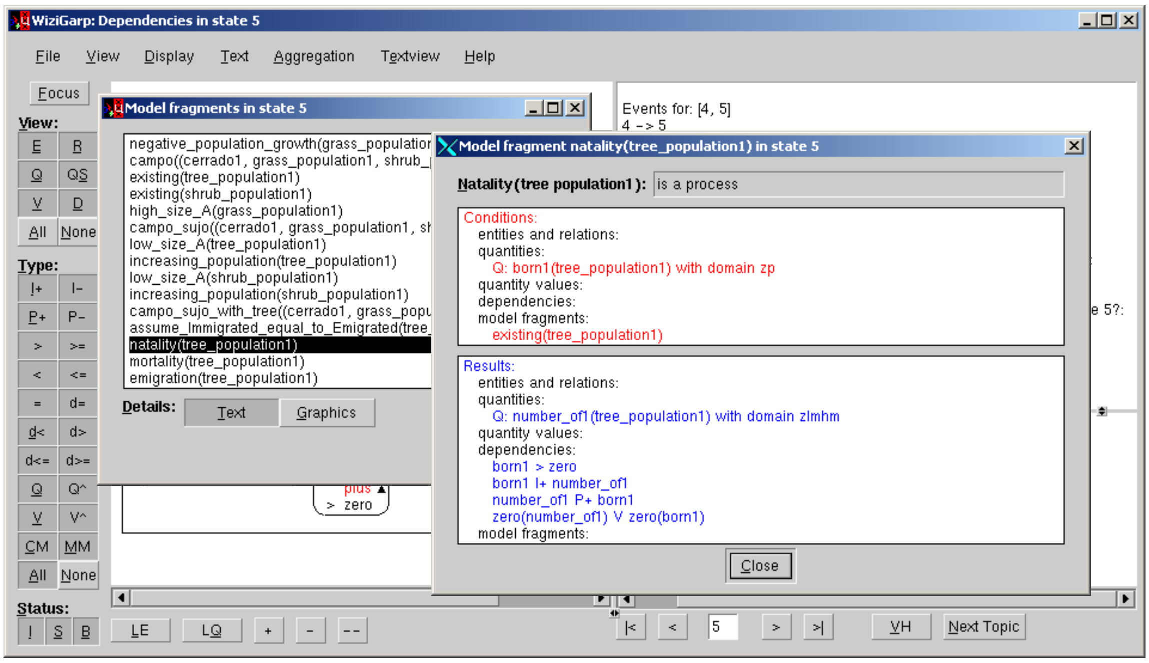
In summary, along the path 1$\rightarrow$4$\rightarrow$5, the tree population comes into existence and its size starts to increase due to colonization (immigrated1), and consequently, the birth, death and emigration processes also become active. Because born1 $>$ dead1, number_of1 keeps increasing.


WiziGarp: Dependencies in state 5
File View Display Text Aggregation Textview Help
Focus
View:
E B Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
LE LQ + - --
tree_population1
inflow1 plus zero = born1 + immigrated1 d= born1 + immigrated1
outflow1 plus zero = dead1 + emigrated1 d= dead1 + emigrated1
born1 plus > zero
immigrated1 plus > zero
number_of1 max high < medium low > zero d> zero
dead1 plus > zero
emigrated1 plus > zero
|< < 5 > >| VH Next Topic

```
Events for: [4, 5]
4 -> 5
causal_events:
   introduced: born1 I+ number_of1: positive effect
   introduced: dead1 I- number_of1: negative submissive effect
   introduced: emigrated1 I- number_of1: negative submissive effect
   introduced: growth1 I+ number_of1: positive effect


Why is the positive effect of born1 'I+' number_of1 introduced in state 5?:
   introduced:
            natality(tree_population1)
Why is the model fragment natality(tree_population1) introduced in state 5?:
   existing(tree_population1)
```

Figure 5.28
The learner asks two queries about the tree population in state 5.

The learner wants to know where the influence from born1 comes from, and asks a query about it. Again, this is done by clicking on the event that specifies the introduction of this effect, and selecting a question from the popup menu that arises. When WiziGarp answers that the reason is the introduction of the natality model fragment (which models the birth of new plants), the learner asks a follow-up query: why this model fragment is introduced. The answer consists of the model fragment that triggered it: existing(tree_population1), as shown in figure 5.28.


WiziGarp: Dependencies in state 5
File
View
Display
Text
Aggregation
Textview
Help
Focus
View:
Type:
Status:
Model fragments in state 5
Details:
Text
Graphics
Model fragment natality(tree_population1) in state 5
Close
LE
LQ
VH
Next Topic

Events for: [4, 5]
4 -> 5

negative_population_growth(grass_populatior
campo((cerrado1, grass_population1, shrub_|
existing(tree_population1)
existing(shrub_population1)
high_size_A(grass_population1)
campo_sujo((cerrado1, grass_population1, sh
low_size_A(tree_population1)
increasing_population(tree_population1)
low_size_A(shrub_population1)
increasing_population(shrub_population1)
campo_sujo_with_tree((cerrado1, grass_popu
assume_Immigrated_equal_to_Emigrated(tree_
natality(tree_population1)
mortality(tree_population1)
emigration(tree_population1)

Natality(tree population1): is a process

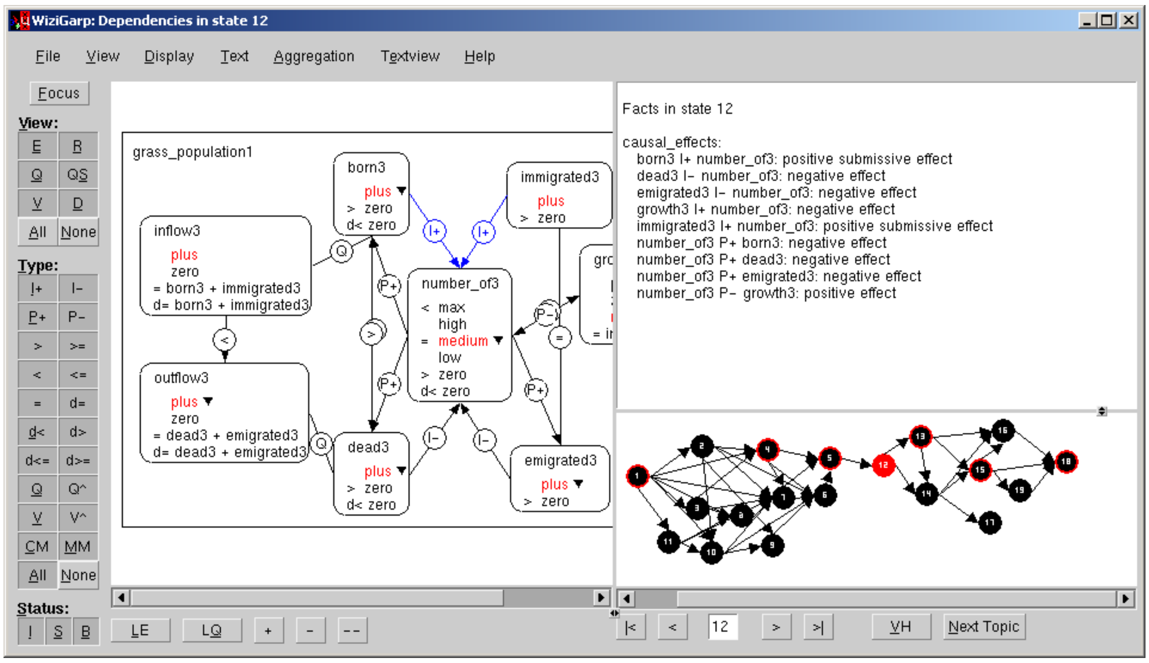
Conditions:
entities and relations:
quantities:
Q: born1(tree_population1) with domain zp
quantity values:
dependencies:
model fragments:
existing(tree_population1)

Results:
entities and relations:
quantities:
Q: number_of1(tree_population1) with domain zlmhm
quantity values:
dependencies:
born1 > zero
born1 I+ number_of1
number_of1 P+ born1
zero(number_of1) V zero(born1)
model fragments:

Figure 5.29
The contents of the model fragment natality for the tree population.

The learner opens the natality model fragment via the model fragment list view to discover that this introduces several dependencies involving born1 and number_of1, when the tree population is existing and includes a quantity born1 (see figure 5.29).

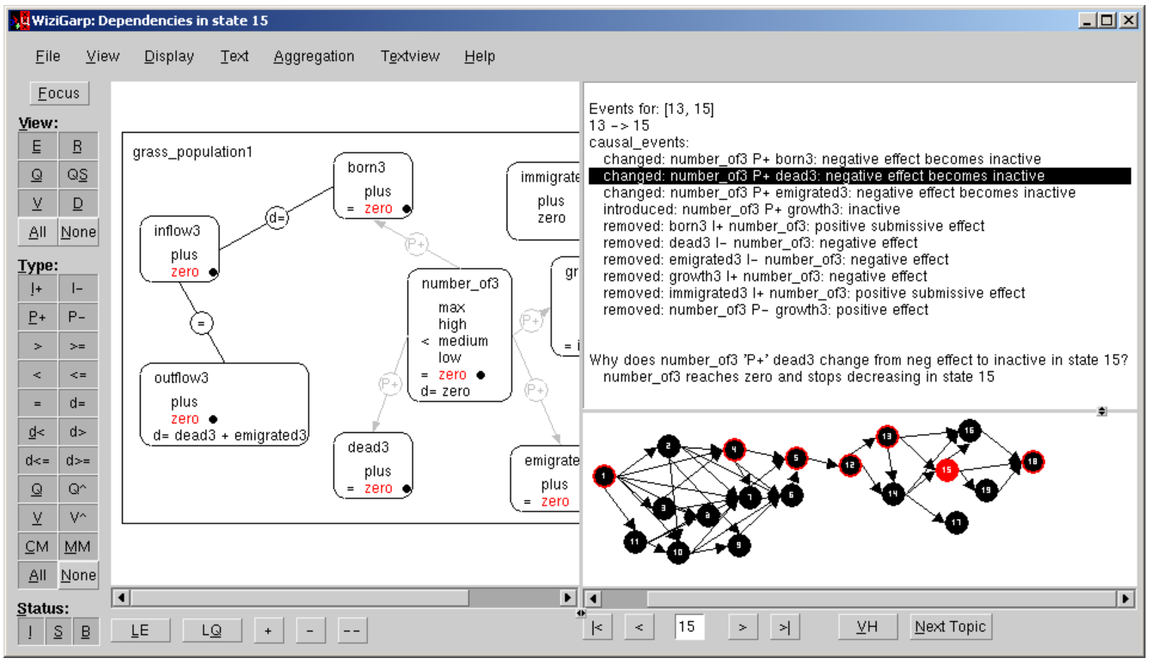
The dependencies say that born1 must be greater than zero; born1 has a positive influence (I+) on number_of1; number_of1 propagates this effect back to born1 (P+); and when number_of1 is zero, then born1 is also zero, due to the value correspondence (V).


WiziGarp: Dependencies in state 12
File View Display Text Aggregation Textview Help
Focus
View:
E R Q QS V D All None
Type:
I+ I– P+ P– > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
LE LQ + - --
grass_population1
born3 plus > zero d< zero
immigrated3 plus > zero
inflow3 plus zero = born3 + immigrated3 d= born3 + immigrated3
number_of3 < max high = medium low > zero d< zero
outflow3 plus zero = dead3 + emigrated3 d= dead3 + emigrated3
dead3 plus > zero d< zero
emigrated3 plus > zero
|< < 12 > >| VH Next Topic

```
Facts in state 12

causal_effects:
   born3 I+ number_of3: positive submissive effect
   dead3 I– number_of3: negative effect
   emigrated3 I– number_of3: negative effect
   growth3 I+ number_of3: negative effect
   immigrated3 I+ number_of3: positive submissive effect
   number_of3 P+ born3: negative effect
   number_of3 P+ dead3: negative effect
   number_of3 P+ emigrated3: negative effect
   number_of3 P– growth3: positive effect
```

Figure 5.30
The dependencies for the grass population in state 12.

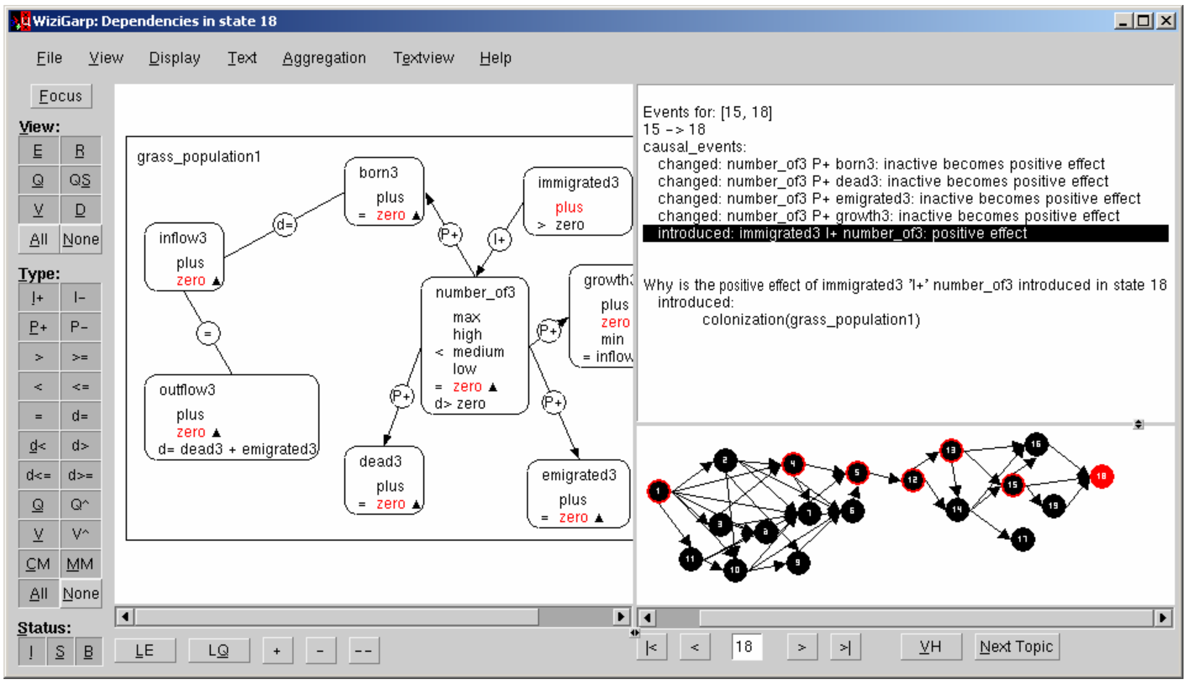
Having reached state 5, the learner changes the focus to the grass population, and continues along the path to investigate this from state 12 onwards. The main screen for this new focus is depicted in figure 5.30. The situation involves the same processes as for the tree population, but the (in)equalities differ, resulting in a different status of the causal effects. The effects of dead3 and emigrated3 are dominant over the submissive effects of born3 and immigrated3.



```
Events for: [13, 15]
13 -> 15
causal_events:
   changed: number_of3 P+ born3: negative effect becomes inactive
   changed: number_of3 P+ dead3: negative effect becomes inactive
   changed: number_of3 P+ emigrated3: negative effect becomes inactive
   introduced: number_of3 P+ growth3: inactive
   removed: born3 I+ number_of3: positive submissive effect
   removed: dead3 I− number_of3: negative effect
   removed: emigrated3 I− number_of3: negative effect
   removed: growth3 I+ number_of3: negative effect
   removed: immigrated3 I+ number_of3: positive submissive effect
   removed: number_of3 P− growth3: positive effect


Why does number_of3 'P+' dead3 change from neg effect to inactive in state 15?
   number_of3 reaches zero and stops decreasing in state 15
```

Figure 5.31
The dependencies for the grass population in state 15, with a user query.

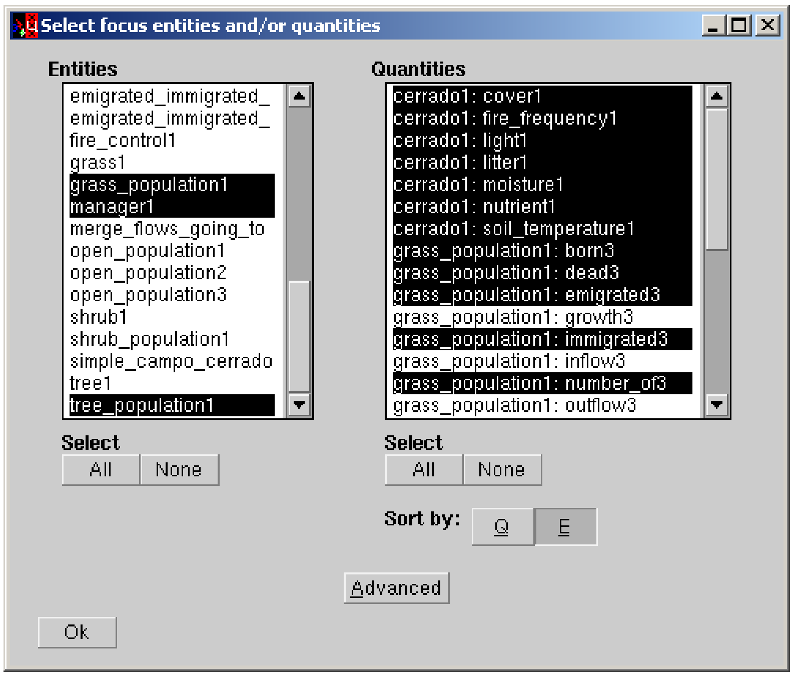
When the learner proceeds to state 13, WiziGarp shows that nothing happens in this transition in terms of causal effects (not shown). In the next step, 13→15, however, all effects become inactive or are removed altogether. When the learner asks a query about one of these events, WiziGarp answers that the change of number_of3 is responsible (see figure 5.31). Because number_of3 reaches zero and stops decreasing, the proportionalities (such as the one in the query) become inactive, and the process influences are removed. The grass population is no longer *existing*, in terms of model fragments. This would be visible if the learner had chosen the model fragment events instead of the causal (effects) events.


WiziGarp: Dependencies in state 18
File
View
Display
Text
Aggregation
Textview
Help
Focus
View:
E R Q QS V D All None
Type:
I+ I- P+ P- > >= < <= = d= d< d> d<= d>= Q Q^ V V^ CM MM All None
Status:
I S B
LE LQ + - --
grass_population1
born3 plus = zero
immigrated3 plus > zero
inflow3 plus zero
number_of3 max high < medium low = zero d> zero
growth3 plus zero min = inflow
outflow3 plus zero d= dead3 + emigrated3
dead3 plus = zero
emigrated3 plus = zero
|< < 18 > >| VH Next Topic

```
Events for: [15, 18]
15 -> 18
causal_events:
   changed: number_of3 P+ born3: inactive becomes positive effect
   changed: number_of3 P+ dead3: inactive becomes positive effect
   changed: number_of3 P+ emigrated3: inactive becomes positive effect
   changed: number_of3 P+ growth3: inactive becomes positive effect
   introduced: immigrated3 I+ number_of3: positive effect

Why is the positive effect of immigrated3 'I+' number_of3 introduced in state 18
   introduced:
          colonization(grass_population1)
```

Figure 5.32
The dependencies for the grass population in state 18, with another user query.

The learner continues to final state 18. In this transition the positive effect of immigrated3 is introduced, which causes number_of3 to start increasing again from zero (see figure 5.32). When the learner asks why this happens, WiziGarp answers that this is due to the colonization process, as was the case for the tree population earlier.


Select focus entities and/or quantities

Entities
emigrated_immigrated_
emigrated_immigrated_
fire_control1
grass1
grass_population1
manager1
merge_flows_going_to
open_population1
open_population2
open_population3
shrub1
shrub_population1
simple_campo_cerrado
tree1
tree_population1

Quantities
cerrado1: cover1
cerrado1: fire_frequency1
cerrado1: light1
cerrado1: litter1
cerrado1: moisture1
cerrado1: nutrient1
cerrado1: soil_temperature1
grass_population1: born3
grass_population1: dead3
grass_population1: emigrated3
grass_population1: growth3
grass_population1: immigrated3
grass_population1: inflow3
grass_population1: number_of3
grass_population1: outflow3

Figure 5.33
The learner selects a large set of focus quantities.

Now, after investigating two populations separately, the learner becomes interested in whether they affect each other in any way. To pursue this interest, the learner selects a new focus, by first selecting all quantities, and then deselecting some which do not seem very interesting: the quantities for the shrub population, and the growth, inflow and outflow quantities (see figure 5.33). Because this large set does not fit into the main screen diagram, the learner opens the dependencies view from the view menu. This view does not have a text view or state-graph integrated in it, and therefore allows more space for the dependencies themselves.

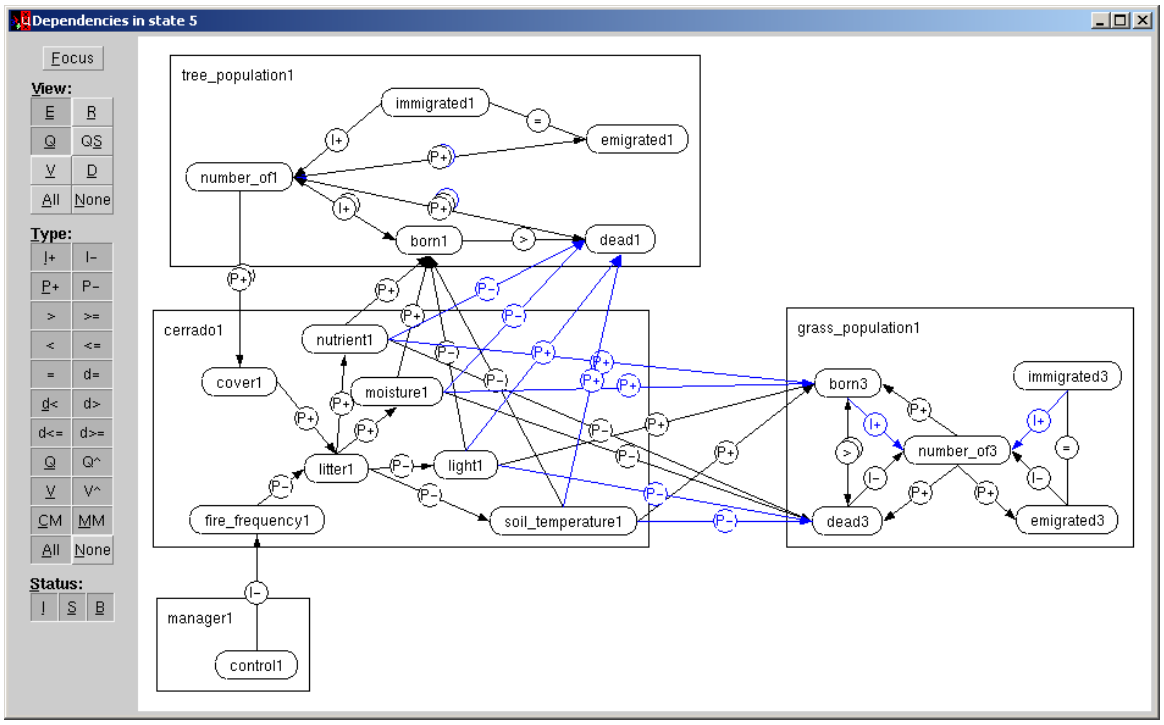
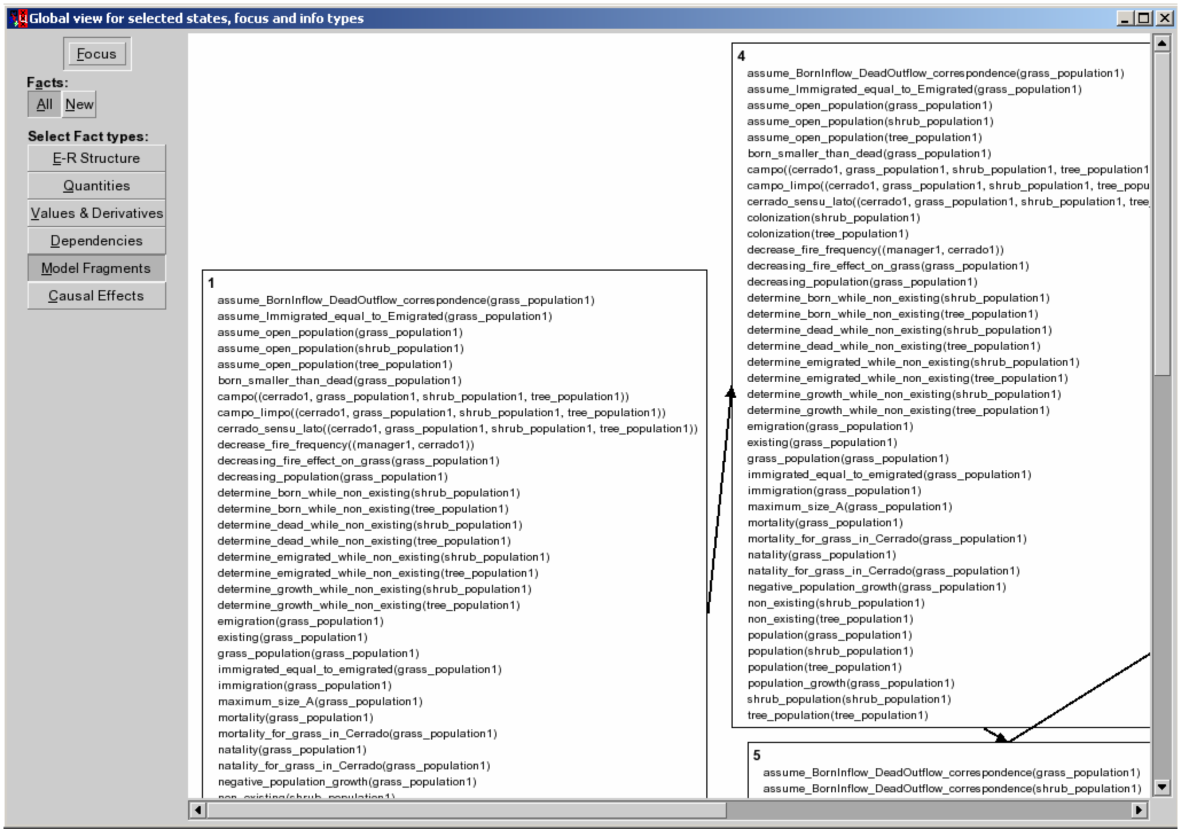


Figure 5.34
The dependencies for the selected set of quantities.

After modification of the display settings to hide the quantity spaces (to save space) and some adjustment of the layout, the result looks as displayed in figure 5.34. This diagram shows the learner that there are interaction effects between the two populations and the Cerrado, and that the Cerrado is also influenced by the manager.

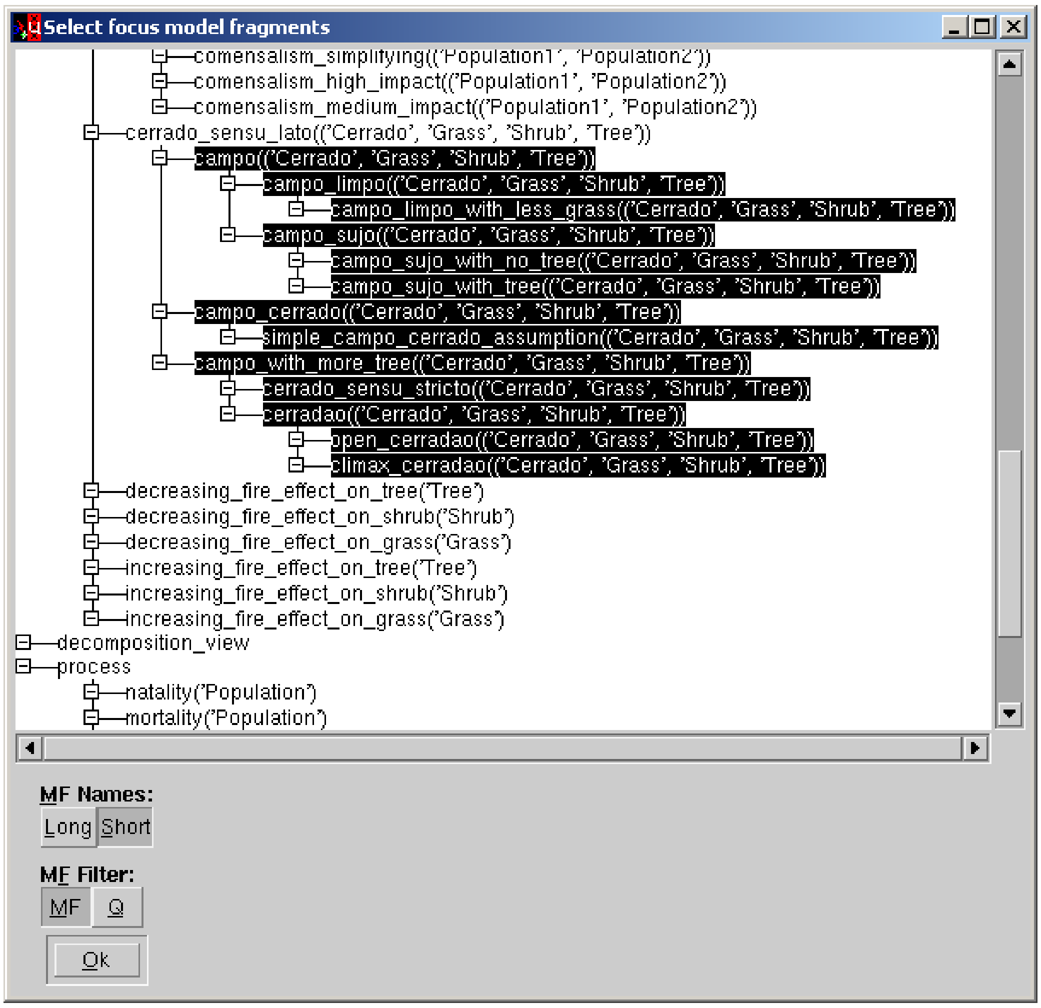
The tree population affects the Cerrado's cover, and the Cerrado quantities nutrient1, moisture1, light1 and soil_temperature1 affect the amount of births and deaths in the tree population and the grass population (in fact, also the shrub population, but this is not shown in the figure).



```
1
assume_BornInflow_DeadOutflow_correspondence(grass_population1)
assume_Immigrated_equal_to_Emigrated(grass_population1)
assume_open_population(grass_population1)
assume_open_population(shrub_population1)
assume_open_population(tree_population1)
born_smaller_than_dead(grass_population1)
campo((cerrado1, grass_population1, shrub_population1, tree_population1))
campo_limpo((cerrado1, grass_population1, shrub_population1, tree_population1))
cerrado_sensu_lato((cerrado1, grass_population1, shrub_population1, tree_population1))
decrease_fire_frequency((manager1, cerrado1))
decreasing_fire_effect_on_grass(grass_population1)
decreasing_population(grass_population1)
determine_born_while_non_existing(shrub_population1)
determine_born_while_non_existing(tree_population1)
determine_dead_while_non_existing(shrub_population1)
determine_dead_while_non_existing(tree_population1)
determine_emigrated_while_non_existing(shrub_population1)
determine_emigrated_while_non_existing(tree_population1)
determine_growth_while_non_existing(shrub_population1)
determine_growth_while_non_existing(tree_population1)
emigration(grass_population1)
existing(grass_population1)
grass_population(grass_population1)
immigrated_equal_to_emigrated(grass_population1)
immigration(grass_population1)
maximum_size_A(grass_population1)
mortality(grass_population1)
mortality_for_grass_in_Cerrado(grass_population1)
natality(grass_population1)
natality_for_grass_in_Cerrado(grass_population1)
negative_population_growth(grass_population1)

4
assume_BornInflow_DeadOutflow_correspondence(grass_population1)
assume_Immigrated_equal_to_Emigrated(grass_population1)
assume_open_population(grass_population1)
assume_open_population(shrub_population1)
assume_open_population(tree_population1)
born_smaller_than_dead(grass_population1)
campo((cerrado1, grass_population1, shrub_population1, tree_population1
campo_limpo((cerrado1, grass_population1, shrub_population1, tree_popu
cerrado_sensu_lato((cerrado1, grass_population1, shrub_population1, tree
colonization(shrub_population1)
colonization(tree_population1)
decrease_fire_frequency((manager1, cerrado1))
decreasing_fire_effect_on_grass(grass_population1)
decreasing_population(grass_population1)
determine_born_while_non_existing(shrub_population1)
determine_born_while_non_existing(tree_population1)
determine_dead_while_non_existing(shrub_population1)
determine_dead_while_non_existing(tree_population1)
determine_emigrated_while_non_existing(shrub_population1)
determine_emigrated_while_non_existing(tree_population1)
determine_growth_while_non_existing(shrub_population1)
determine_growth_while_non_existing(tree_population1)
emigration(grass_population1)
existing(grass_population1)
grass_population(grass_population1)
immigrated_equal_to_emigrated(grass_population1)
immigration(grass_population1)
maximum_size_A(grass_population1)
mortality(grass_population1)
mortality_for_grass_in_Cerrado(grass_population1)
natality(grass_population1)
natality_for_grass_in_Cerrado(grass_population1)
negative_population_growth(grass_population1)
non_existing(shrub_population1)
non_existing(tree_population1)
population(grass_population1)
population(shrub_population1)
population(tree_population1)
population_growth(grass_population1)
shrub_population(shrub_population1)
tree_population(tree_population1)

5
assume_BornInflow_DeadOutflow_correspondence(grass_population1)
assume_BornInflow_DeadOutflow_correspondence(shrub_population1)
```

Figure 5.35
The global view with all model fragments, containing too much information.

To see if the simulation contains any more interesting things to be investigated, the learner opens the global view again. To get an overview of the model fragments, the learner selects all model fragments in the advanced focus settings. The result is shown in figure 5.35, which contains too much information to comprehend. Besides the model fragments for the processes which apply to each of the three plant populations, there are several other types of model fragments, including various assumptions, and descriptive model fragments which characterize the different populations, or the Cerrado as a whole.


Select focus model fragments

```
          ⊟—comensalism_simplifying((’Population1’, ’Population2’))
          ⊟—comensalism_high_impact((’Population1’, ’Population2’))
          ⊟—comensalism_medium_impact((’Population1’, ’Population2’))
      ⊟—cerrado_sensu_lato((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
          ⊟—campo((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
              ⊟—campo_limpo((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
                  ⊟—campo_limpo_with_less_grass((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
              ⊟—campo_sujo((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
                  ⊟—campo_sujo_with_no_tree((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
                  ⊟—campo_sujo_with_tree((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
          ⊟—campo_cerrado((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
              ⊟—simple_campo_cerrado_assumption((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
          ⊟—campo_with_more_tree((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
              ⊟—cerrado_sensu_stricto((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
              ⊟—cerradao((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
                  ⊟—open_cerradao((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
                  ⊟—climax_cerradao((’Cerrado’, ’Grass’, ’Shrub’, ’Tree’))
      ⊟—decreasing_fire_effect_on_tree(’Tree’)
      ⊟—decreasing_fire_effect_on_shrub(’Shrub’)
      ⊟—decreasing_fire_effect_on_grass(’Grass’)
      ⊟—increasing_fire_effect_on_tree(’Tree’)
      ⊟—increasing_fire_effect_on_shrub(’Shrub’)
      ⊟—increasing_fire_effect_on_grass(’Grass’)
  ⊟—decomposition_view
  ⊟—process
      ⊟—natality(’Population’)
      ⊟—mortality(’Population’)
```

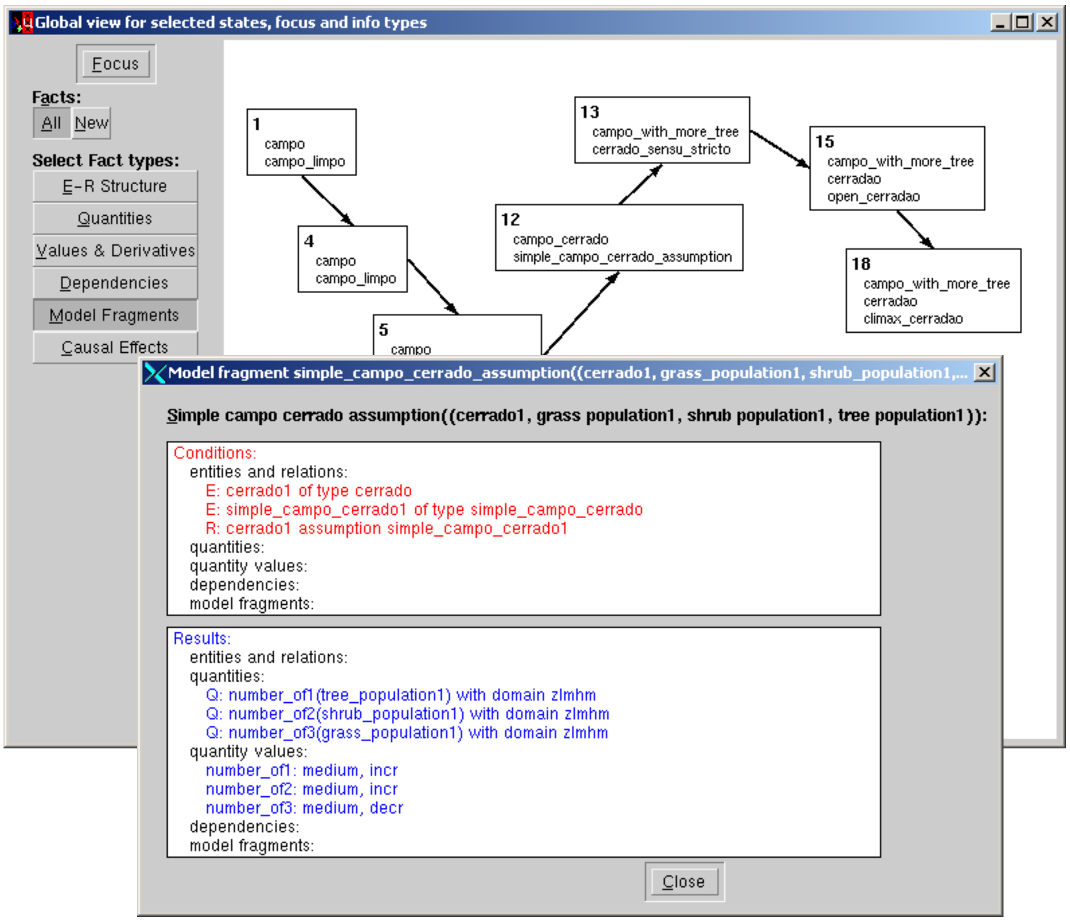
MF Names: Long Short

MF Filter: MF Q

Ok

Figure 5.36
The learner selects the Cerrado vegetation type model fragments.

To reduce the number of model fragments to an acceptable level, the learner selects particular types of model fragment: the subtypes of cerrado_sensu_lato, the specific name for the vegetation type characteristics of the Cerrado. This selection is done in the advanced focus settings, as shown in figure 5.36.


Global view for selected states, focus and info types
Focus
Facts:
All
New
Select Fact types:
E-R Structure
Quantities
Values & Derivatives
Dependencies
Model Fragments
Causal Effects
1 campo campo_limpo
4 campo campo_limpo
5 campo
12 campo_cerrado simple_campo_cerrado_assumption
13 campo_with_more_tree cerrado_sensu_stricto
15 campo_with_more_tree cerradao open_cerradao
18 campo_with_more_tree cerradao climax_cerradao
Model fragment simple_campo_cerrado_assumption((cerrado1, grass_population1, shrub_population1,...
Close

Simple campo cerrado assumption((cerrado1, grass population1, shrub population1, tree population1)):

```
Conditions:
   entities and relations:
      E: cerrado1 of type cerrado
      E: simple_campo_cerrado1 of type simple_campo_cerrado
      R: cerrado1 assumption simple_campo_cerrado1
   quantities:
   quantity values:
   dependencies:
   model fragments:
```

```
Results:
   entities and relations:
   quantities:
      Q: number_of1(tree_population1) with domain zlmhm
      Q: number_of2(shrub_population1) with domain zlmhm
      Q: number_of3(grass_population1) with domain zlmhm
   quantity values:
      number_of1: medium, incr
      number_of2: medium, incr
      number_of3: medium, decr
   dependencies:
   model fragments:
```

Figure 5.37
The global view with the Cerrado vegetation type model fragments.

The resulting global view is shown in figure 5.37. This figure exemplifies the Cerrado Succession Hypothesis, which suggests that an empty grassland (campo limpo) can transform over time into a vegetation without grass and a maximum number of trees (climax cerrado). The model fragment *simple_campo_cerrado_assumption* (which is active in state 12) implements the simplifying assumption that a typical state in the development of the Cerrado is the state in which all three population sizes have the value medium. This assumption is the reason why all paths in the state-transition graph include state 12. The conditions of the model fragment contain three statements, of which the latter two are about the assumption simple_campo_cerrado. These two are examples of model ingredients which were deliberately hidden in scenario 1.

5.4 Discussion

The didactic plan representation allows a teacher to plan a series of topics which are treated in a specified sequence of didactic actions, as exemplified by scenario 1. Together with the possibility for learners to explore the simulation on their own, as exemplified by scenario 2, this makes WiziGarp a powerful tool for learning and teaching.

In order to demonstrate the range of WiziGarp's functionality in the form of two distinctly different scenarios, some of the didactic means were used in one scenario but not in the other. For example, scenario 1 only presented facts, contrastive explanations between states, and exercise questions for the learner to answer. Scenario 2 on the other hand included facts, events (on the local level), and queries from the learner answered by WiziGarp's causal explanations. Of course, in other interaction sessions, there can be a mix of elements found in these two scenarios. At any moment during scenario 1, the learner could have taken over the initiative by doing something other than clicking the Next Topic button. In scenario 2, the learner could have clicked this button to see if WiziGarp would have something interesting to tell. By modifying the intro and outro texts in the didactic plan, the learner could be stimulated to open other views at certain points during the interaction, before going to the next topic in the didactic plan.

5.4.1 Evaluation of WiziGarp

Unlike VisiGarp, WiziGarp has not yet been evaluated by many users, as it currently has the status of a prototype. Nevertheless, two example interaction scenarios with WiziGarp have been presented in this chapter to demonstrate that WiziGarp solves some of the problems of VisiGarp and related existing work. The question generation module QUAGS, which is part of WiziGarp, has been evaluated separately by domain experts, as discussed in section 4.4.8.5, yielding positive results. But besides the generation of questions, WiziGarp has a number of other new functionalities, such as the didactic plan and focussing mechanisms, the causal explanations, the contrastive explanations, the global view, and the navigation options in the main screen. To evaluate the usefulness of all of these in addition to, or in combination with the functionality already incorporated in VisiGarp is no easy task. The different didactic means need to be evaluated separately, and in various combinations. Combinations are necessary if support is desired from WiziGarp in realistic educational settings. Interesting experimental setups may include a comparison of different didactic approaches, *i.e.*, student-initiated exploration vs. system-initiated tutoring, for the same set of topics. Also, the effectiveness of diagrams combined with text (in WiziGarp) might be compared to the diagrams (or texts) alone (in VisiGarp, or a restricted version of WiziGarp).

5.4.2 Didactic Planning

As demonstrated by scenario 1, WiziGarp can be used with a didactic plan to support a teaching style which combines expository explanations with exercise questions. Currently, the didactic plans are written by hand, which has to be done before a session starts. While being clear and powerful, this approach is also time-consuming and inflexible. Further work will therefore address the automatic generation of didactic plans. Considerations regarding the design of didactic plans, such as described in section 5.2, will have to be tested and generalized into generic rules for generating a didactic plan for a random simulation. A didactic plan could also define a longer learning route, progressing along multiple simulations, as envisioned by Salles and Bredeweg [2001b]. An even more demanding goal to pursue is not only to automatically generate a plan beforehand, but also to adapt it to a particular learner's situation during the interaction. For example, if a learner can not answer a particular exercise question presented by WiziGarp, or keeps focusing on the same topic, WiziGarp might do something to recognize and address the learner's need. To this end, the current work must be integrated with mechanisms for diagnosing learner behaviour, as described by de Koning et al. [2000].

Currently, WiziGarp has no way to interpret answers from a learner to the tutorial questions posed by WiziGarp. When such functionality would be added this would allow the creation of a learner model, which represents what the learner might or might not know. Heuristic rules for determining this have been presented in the literature, including 'if the user (in)correctly answers a question, assume they (don't) know the concept', and 'if all the sub-concepts taught so far are known, assume the concept is probably known' [Cawsey, 1993]. Related work on explanation planning by Suthers [1992] distinguishes several tasks in the different phases of planning, such as thinking about what the user might know, propose relevant explanatory models, choosing between different models, identifying the need for supporting explanations, and ordering the elements of an explanation in an optimal way. These tasks are implemented using appropriate mechanisms, such as spreading activation, top-down goal refinement, plan critics, and others, which match characteristics of the specific tasks. The explanation planning mechanism proposed by Moore [1995] uses a record of all the plans leading to an explanation as an explicit representation of the discourse planning history. This way, alternative explanations can be generated for a communicative goal in case a user is unsatisfied with an explanation.

5.4.3 Collaborative Learning

So far, the tools described in this thesis have been designed for and evaluated in contexts with single users. An interesting avenue for further research is to consider how tools such as VisiGarp and WiziGarp could be used by (and adapted for) multiple learners learning collaboratively. The extensive use of external representations might be useful in such settings, because they are supposed to support communication. According to Suthers [2003], there is an interesting relationship between the dialogue between learners and the external representations that they encounter or create, because the type of representation may limit, or support particular reasoning and communication. A large body of research focuses on the role of argumentation in scientific investigation. For example, Belvedere [Suthers et al., 1997] has a tool for creating argumentation diagrams and includes coaching feedback to help learners identify weak spots in their argumentation. The idea of learning by arguing [Pilkington et al., 1992] stresses the importance of dialectical argumentation. Here, the learner is not receiving material passively from an expert (system), but rather arguing with other learners, real or artificial [Aïmeur et al., 1997]. By attacking the other participant's arguments and defending their own, learners learn more about both viewpoints. The chances of conflicts arising can be maximized by preselecting participants who disagree on particular issues [Quignard and Baker, 1999], or by assigning roles, and incorporating dialogue games in the interface [Bouwer, 1999]. In the latter work, a diagram is created automatically from the participants' dialogue, which shows that the relationship between representation and dialogue may run both ways.

5.5 Conclusion

This chapter has demonstrated the explanatory functionalities of WiziGarp in the Cerrado ecology domain in the form of two simulated scenarios. The first scenario, in which WiziGarp leads the interaction based on a didactic plan, shows that it is possible to predefine a route through the material in a complex simulation, aimed at a learner without much experience in ecology or QR. The technique of aggregating alternative orderings, defined in chapter 3, helps to reduce the complexity of the state-transition graph to digestible proportions. The focusing mechanism helps to direct the attention towards a specific topic, as specified in the didactic plan. The scenario only utilized the main screen of WiziGarp, which demonstrates that it is not necessary to switch between multiple screens to handle a variety of topics, from the structure of the system, via the behaviour of the main quantities, and causal models involving basic processes and interactions between multiple entities. To summarize, based on a didactic plan, WiziGarp can present a learner the essence of what happens to a complex system such as the Cerrado, and why, without overwhelming the learner with details which are often present in qualitative simulations. The didactic plan supports handling of the material step by step from simple topics to more complex topics, while combining expository text with exercise questions.

The second scenario, which is aimed at a learner with more expertise, shows that it is also possible for a user to take control over the interaction. WiziGarp offers support in various ways, by determining the most interesting quantities, offering controls for navigation through the simulation, and various options for manipulating the contents of the diagrams. The different views in WiziGarp show different aspects of the simulation, such as the graphical representation of facts within a state, the model fragment views which link the actual behaviour to the generic domain knowledge, the textual representation of events and graphical value histories along a path, and the global view, which presents an overview of any number of paths. The scenario demonstrated that WiziGarp's aggregation methods are not destructive, but allow easy switching between the abstracted and the original state-transition graph. This scenario also showed the use of context-sensitive queries about events of interest, to which the answer is automatically generated by determining the relevant causal links. To summarize, the functionality of WiziGarp allows an experienced user to quickly get an overview of what is contained in the simulation, and to go into specific topics in detail. Because WiziGarp allows switching from system-controlled to user-controlled interaction at any moment, it is a flexible tool which may be used to support learning about dynamic phenomena in different educational settings.

6

Conclusion

"A life without cause is a life without effect."

Barbarella, Queen of the Galaxy (1968), US Movie directed by Roger Vadim, based on a comic strip by French writer Jean-Claude Forest

6.1 Summary of the Main Results

The main research goal addressed in this thesis was to investigate how explanations of dynamic phenomena can be generated using qualitative simulations, and how these can be utilized in a domain-independent interactive learning environment. The first approach to address these issues was to use visualization as the principal means to communicate qualitative models and simulation results. This has resulted in the design and implementation of a tool, called VisiGarp. This tool extends the current state of the art by generating interactive diagrams of multi-state simulations, incorporating all of the model primitives available in the GARP framework. Evaluation of VisiGarp indicated that students were able to use it to learn about population dynamics in ecology, a domain unfamiliar to them. However, when large models were used, the amount of information often became too large to display on the screen adequately. To resolve this problem, the potential of aggregation principles to reduce the complexity of qualitative simulations has been explored. A set of aggregation techniques has been implemented which select, chunk, generalize, or group information elements, so that the result contains less states and transitions, as well as less information within them. This was done by focusing on events of various kinds, at different levels of abstraction. Finally, a prototype interactive learning environment has been designed and implemented, called WiziGarp, which incorporates the aggregation mechanisms, and expands the communicative functions of VisiGarp. Textual descriptions, causal explanations, contrastive explanations, queries, and exercise questions were added to complement the visualizations. To plan combinations of these didactic means, a method has been presented for specifying didactic plans about qualitative simulations. This allows WiziGarp to be used with different didactic styles, from student-initiated exploration to system-initiated tutoring.

The contributions of this thesis concern different areas of research related to qualitative simulation, *i.e.*, visualization, aggregation, and interactive explanation, each of which is discussed in detail in the sections below.

6.2 Visualization of Qualitative Simulations

Research on information visualization, diagrammatic reasoning, and external representations points out that diagrams have an added value for learning, as they provide an analogical mapping from knowledge structure to visual structure, and provide explicit handles which facilitate reasoning and communication (*e.g.*, see Kulpa [1994], Cox and Brna [1995], and van Someren et al. [1998]). The potential of graphical representations for the communication of qualitative simulations has been investigated in chapter 2. In order to allow the automatic generation of diagrammatic representations based on qualitative simulations, a mapping has been devised from GARP modelling primitives, such as states, entities, quantities and dependencies, to specific visual primitives, such as circles, boxes, ellipses, and arrows. Principles have been formulated which specify how the visual primitives should be combined to offer different views of system behaviour. These principles are meant to support the interpretation and navigation of information elements, while making efficient use of space in meaningful ways. Principles which aim to support interpretation and minimize the effort of learning include showing information in context where necessary, and using abbreviations or icons only for labels which are repeated often. Principles which support search and navigation include using encapsulation or connections for different types of relationships, avoiding needless repetition of visual elements, and assigning richer information elements more space. These principles have been incorporated in the design and implementation of VisiGarp, the tool which provides users with interactive access to the information in a qualitative simulation and the underlying model library in the form of diagrams. The views include the state-transition diagram (for simulation control and navigation), the entities and relations view (showing the structure of the system), the dependencies view (showing causal and mathematical relationships), the quantity value history (showing how quantities change qualitatively), the model fragment views (for an overview and detailed views of relevant model fragments) and hierarchical views (to show is-a or applies-to relationships between entity types or model fragment types).

After formative pilot studies with VisiGarp, which led to the improvement of usability issues, an evaluation study has been carried out with thirty first-year undergraduate students. The setup included paper-based pre- and post-tests concerning domain-specific knowledge about the Cerrado, and the ability to read the diagrammatic representations. The treatment consisted of interaction with VisiGarp about two scenarios of increasing complexity, with accompanying exercise questions to be answered. A questionnaire was used afterwards to gauge the participants' attitudes about VisiGarp. The results confirmed that undergraduate students without qualitative reasoning experience can learn to use VisiGarp to complete non-trivial exercises about a domain unfamiliar to them, within ninety minutes. The students can read the different kinds of visualizations fairly accurately already from the start, and their ability to correctly answer questions about ecology improves significantly during the course of the experiment. The students rated VisiGarp quite positively, overall. The improvements in the layout and navigation mechanisms led to adequate diagrams for the first scenario, but the second, more complex, scenario still gave rise to problems related to the large number of things to be displayed at once.

VisiGarp is currently in use by researchers in various domains to test and communicate simulation results of their models [Nuttle et al., 2004, Neumann and Bredeweg, 2004, Tullos et al., 2004], and by several teachers of academic courses which incorporate qualitative modelling in ecology, physics and chemistry [Salles and Bredeweg, 2003, Alvarez-Bravo et al., 2004, Salles et al., 2004]. This proves that there is a need for a tool like VisiGarp to communicate knowledge about system behaviour, and that VisiGarp can be used by people from various backgrounds in different academic settings. However, when the complexity of the simulation increases, this affects the usability of the tool. Domain experts and teachers may know how to deal with such complexity, but learners can be overwhelmed by the amount of information present in large simulations. For this reason, it was important to consider the issues of aggregation and interactive explanation.

6.3 Aggregation of Qualitative Simulations

In order to address the question of how to reduce the complexity of qualitative simulations, a conceptual framework was presented which includes five levels of abstraction and four aggregation methods. This builds on, and extends related work on aggregation of qualitative simulations [Mallory and Porter, 2000, de Koning et al., 2000]. The five levels of abstraction are characterized as follows: (1) the system state level, which describes the state of the system as facts holding at a particular moment or during an interval in time; (2) the local event level (corresponding largely to Mallory's local level), at which successive states are compared and meaningful differences are represented as local events; (3) the path segment level, which denotes a necessary sequence of events taking place along a sequence of states without branches; (4) the path level, which denotes a possible sequence of events, that may contain branches and/or cycles; and (5) the global level, at which all alternative possibilities for the behaviour of the system are considered. The aggregation methods are selection, chunking, generalization, and grouping, which are characterized by the way they leave out, combine, or refer to information elements. A set of aggregation techniques has been implemented which use these methods to decrease the amount of information within states and transitions as well as the number of states and transitions to be considered. These techniques apply at one or more of the abstraction levels.

To reduce the amount of information within states, a technique was presented for classification of the status of causal effects, which allows selection of only the effective dependencies, while hiding inactive, submissive, and balanced ones. On the local level, techniques for the recognition of events (based on selection, chunking and grouping) further reduce the amount of information by focusing on *what changes*, rather than *what is*. Events are recognized in terms of values and derivatives, causal and mathematical dependencies, entities and structural relations, and model fragments. At the path segment level, techniques are given for the recognition of path segments and chunking of value events. At the path level, state-transition graph characteristics are recognized as well as value events which represent the maximum and minimum reached in a particular path. At the global level, the number of states and transitions is reduced, thereby strongly reducing the number of alternative behaviours. This is done based on two conceptual principles: preferring linear descriptions of what happens, and focusing on begin and end of event sequences. Algorithms are presented for *transitive reduction*, and *aggregation of sequence*, both of which abstract from temporal information, using the method of selection and generalization, respectively.

Taken together, the set of techniques presented provides a powerful and flexible way of compressing the information in a GARP simulation, supporting different views on the behaviour of the simulated system. For the example simulation about the Cerrado Succession Hypothesis (as well as several other GARP simulations), applying the techniques results in a significant reduction in the number of the states and behaviours, as well as in the amount of information within each state. This reduces the perceived complexity, while retaining and distilling the conceptually important information to be communicated.

6.4 Interactive Explanation

In order to support the communication of knowledge, it is important to consider how to present the relevant information in a dialogue interaction. The existing research on explanation generation predominantly addresses explanations of static knowledge contained in databases or knowledge bases, rather than explanations of behaviour, as contained in qualitative simulations. Therefore, an analysis has been provided (in chapter 4) of the didactic goals which should be supported in order to learn about system behaviour in conceptual terms. Working out these didactic goals in detail has led to a comprehensive set of questions which can be addressed using the types of information found in qualitative simulations. The answers to these questions, *i.e.*, the explanations, vary in content and form, depending on the type of question.

Some of the questions can be answered by *visual explanations*, *i.e.*, the visualizations discussed in chapter 2. To answer the remaining questions, the range of communicative functionalities had to be extended, exploiting the results of the aggregation mechanisms described in chapter 3. For this reason, the VisiGarp tool has been redesigned, resulting in the WiziGarp prototype learning environment. In WiziGarp, the visualizations have been complemented by other didactic means: textual descriptions of facts and events, queries, causal explanations, contrastive explanations, and tutoring questions. The *textual descriptions* support the recognition and interpretation of the important aspects of a system's behaviour. The user can ask *queries* about facts and events by selecting them from an automatically generated list of relevant queries, thereby avoiding the need for natural language interpretation of user input. For queries which ask for a *causal explanation* (*e.g.*, why did event E happen?), an explanation is constructed which presents the concurrent or preceding facts or events which have caused the event in question. By recursively asking queries about its causes, an event can be explained in full, tracing back to the initial situation, or assumptions made during the course of a simulation. The *contrastive explanations* list the differences between two selected states in terms of facts. This is useful to compare two situations which need not be in the same sequence. The *tutoring questions* are questions which the system can pose to the student to practise his/her problem-solving skills. These questions are automatically generated given a selected topic, perspective, and certain other question characteristics. Having a rich set of didactic means requires planning of when and how to use them. The original design of VisiGarp, as described in chapter 2, was based on an approach of free exploration, and did not include any didactic planning in itself. Because students need support and guidance in order to have an effective learning experience, the evaluation studies of VisiGarp included exercise questions and instructions on which scenarios or screens to open next. These instructions were written by hand, which takes up a large amount of time. For this reason, WiziGarp has been equipped with a didactic planning module, which can read a didactic plan that is much more compact and easy to write than the detailed instructions. The existing literature on didactic planning was not specific enough for the purposes of this work, so a new structure has been defined for didactic plans in the context of qualitative simulations.

A didactic plan consists of a list of topics, and a list of didactic actions, which make use of the topics. A topic consists of three parts: system scope, time frame, and type of information. The choice on these three dimensions allows small and large topics to be defined, accommodating various difficulty levels. A didactic action contains a specific view in WiziGarp, a topic to be displayed in it, and a specification of the kind of didactic means to be used (*i.e.*, fact descriptions, event descriptions, contrastive explanations, or tutoring questions). Before the information is presented (in graphical or textual format) it is filtered so that only the information elements which relate to the specified topic are shown. If the topic definition is specific enough, this creates a focus which solves the problem of having too much information to display.

The approach taken allows didactic plans to be specified which comply with existing ideas on didactic planning, such as progressing from simple to more complex material [Goldstein, 1982], dealing with structure before moving to behaviour [White and Frederiksen, 1990, Salles and Bredeweg, 2001a], and leaving more advanced users more freedom to explore larger topics than novices [Winkels and Breuker, 1993]. At one end of the spectrum, the user is in full control to open views, as in VisiGarp, select topics, and ask for explanations at will, whereas on the other end of the spectrum, the user is guided by a strict sequence of specified topics, and shown questions and textual explanations. The interaction scenarios with WiziGarp presented in chapter 5 are two examples which demonstrate how this works. The way that didactic plans are formulated is flexible, so that didactic approaches can be defined which combine, or alternate styles anywhere within the spectrum from user-initiated exploration to system-directed tutoring.

Currently, the didactic plans are written by hand, but a set of domain-independent heuristics has been presented which can be used to generate didactic plans automatically.

6.5 Future Work

Besides extensions of the didactic means already mentioned in chapter 2 and 4, future work will address the following issues: evaluation of the tools in educational settings, addressing collaborative learning, building interactive models, and incorporating student modelling and diagnosis to improve the didactic planning. Each of these issues is discussed in more detail in the following subsections.

Evaluation in Educational Settings. VisiGarp has been evaluated in various contexts, using students, teachers and domain experts. The results have confirmed its value in terms of usability and usefulness, especially for graduate students and domain experts when they have used it for extended periods of time. To test the educational value in other contexts, more rigorous evaluation is necessary, using larger numbers of students in real educational settings. As mentioned earlier, steps are taken to employ VisiGarp in courses on physics and chemistry [Alvarez-Bravo et al., 2004, Salles et al., 2004]. WiziGarp has a richer set of functionalities, which allow even more interesting experimental setups. These could include comparing different didactic approaches, *i.e.*, student-initiated exploration vs. system-initiated tutoring, for the same set of topics. Also, the effectiveness of diagrams combined with text (in WiziGarp) might be compared to the diagrams (or texts) alone (in VisiGarp, or a restricted version of WiziGarp).

Collaborative Learning. It would be interesting to investigate whether and how the design of VisiGarp and WiziGarp influences how students behave when working together. Do the diagrams function as external representations for communication between the students as well as between the system and the student? Will students incorporate the GARP modelling primitives and their visualizations in their own communication? Perhaps, argumentation can be employed as a didactic strategy in relation to qualitative simulation domains. When different simulation models produce conflicting results, this may be a source for computer-supported argumentation [Pilkington et al., 1992, Aïmeur et al., 1997, Quignard and Baker, 1999, Bouwer, 1999].

Building Interactive Models. There is a growing amount of educational software which lets students create diagrams as a conceptual model of a particular scientific phenomenon. For example, concept mapping methods allow users to place concepts and links on a canvas to create a knowledge model, capturing their understanding [Coffey et al., 2003]. Cavalli-Sforza [1998] suggests that creating a representation yourself has more effect than receiving one. This complies with the view that learning is an active process of constructing knowledge [Bruner, 1961]. Forbus argues that building computer models should become a major hobby, not only because it deepens our understanding, but also because it is fun to do, just like people enjoy building physical models of planes, houses, etc. [Forbus, 1996b].

Contemplating the above, it seems fruitful to incorporate modelling in an interactive learning environment to support qualitative understanding of system behaviour. Several model-building tools exist [Soloway et al., 1997, Reichherzer et al., 1998, Leelawong et al., 2001, Forbus et al., 2001], but HOMER [Jellema, 2000, Bessa Machado and Bredeweg, 2002, 2003] and MOBUM [Bessa Machado, 2004] are especially good candidates, since they are also based on the GARP framework for qualitative simulation. Integration of WiziGarp's functionality into such model-building tools seems promising, as the software would then support the cycle of building, testing and inspecting models.

Student Modelling, Diagnosis, and Didactic Planning. An intelligent didactic planner incorporates knowledge about the student, as well as knowledge about the domain. The work by Salles and Bredeweg [2001b] proposes model progression techniques for the GARP simulation framework, to inform a didactic planner about which topic to pursue next. The work by de Koning et al. [2000] on the STARlight system deals with modelling and diagnosis of students' reasoning about system behaviour, also using the GARP framework. By asking prediction questions to a student about a simulation, the STARlight system can find out which reasoning step(s) the student has missed when he deviates from the ideal reasoning trace. However, STARlight did not include a graphical interface which allowed domain-independent investigation of simulations. It seems profitable to integrate model progression and diagnosis with the functionalities of WiziGarp, as they largely complement each other. Currently, the STARlight system incorporates only questions about what is true in one state, and what will happen in the next. To exploit the full potential of combining both approaches, the diagnosis techniques should be extended to include reasoning about events across multiple states. This would lead to more accurate modelling and diagnosis of students' reasoning behaviour. If this is incorporated into the didactic planning, it would also lead to a tailored sequence of relevant explanations.

6.6 Concluding Remarks

To arrive at full-fledged intelligent learning environments which exploit qualitative simulation to the fullest extent, it will be necessary to integrate the different functionalities mentioned above, with the kinds of tools described in this thesis. When the whole process of scientific investigation is addressed, including collaborative modelling, simulation, support for argumentation, diagnosis and coaching feedback, the result may be called a *science learning space* [Koedinger et al., 1999]. More specifically, combining model-building with aggregation, visualization and explanation of simulation results will lead to *interactive model-building environments* [Bouwer et al., 2002]. Learning technology which exploits the interaction with conceptual models, such as the work described in this thesis, will contribute to more effective and exciting ways of learning and teaching about dynamic phenomena.

A

VisiGarp Treatment Exercise Questions

This appendix lists the exercise questions used in the treatment during the evaluation of VisiGarp 1.0, as described in section 2.5.3. These questions are meant to introduce novice students to the VisiGarp tool and the domain of population ecology, especially in the Brazilian Cerrado, described in section 2.5.1. Two simulation scenarios are used. Scenario 1 is about a single plant population. Scenario 2 is about the Cerrado, which consists of three plant populations. The treatment contains instructions on how to use VisiGarp, as well as multiple choice questions which require detailed study of the material to be answered.

The original version of these questions was in Dutch [Tjaris, 2002], aimed at the Dutch participants in the evaluation study. For convenience, they are translated here into English.

Scenario 1

1. What is your name?

2. What is the participant number given to you by the experimenter?

VisiGarp:

- In the **File** menu, choose the option Open Saved Simulation and then select the document Scenario1.svst
- The layout of the screen can be adjusted by dragging objects with the mouse
- In the *State transition graph*, select state **3**
- In the **View** menu, choose the option *Entities and relations*

3. Which statement is true?

a. open_population1 consists_of population1

b. population1 consist_of biological_species1

c. open_population1 is an assumption about biological_species1

VisiGarp:

- Close the *Entities and relations* view
- In the **View** menu, choose the option *Dependencies*

! Make sure that state **3** is still selected

4. Which of the terms below is **not** a quantity (characteristic) of the entity **population1**?

a. biological_species1

b. born1

c. number_of1

VisiGarp:

- In the **View** menu in the *Dependencies in state 3* screen, click on the option *Q Spaces*
- The layout of the diagram can be adjusted by dragging objects with the mouse

5. Which values can the number_of the population adopt?

a. zero, plus

b. zero, normal, max

c. zero, plus, normal, max

6. Which values can the number_of births (born) adopt?

a. zero, normal, max

b. zero, plus

c. min, zero, plus

VisiGarp:

- In the **View** menu in the *Dependencies in state 3* screen, click on the option *Values* and the option *Derivatives*

7. Which value does the number_of the population have? And does this number increase or decrease in this state?

a. number_of has the value zero and is increasing

b. number_of has the value normal and is decreasing

c. number_of has the value normal and is increasing

8. Which value does born of the population have? Does born increase or decrease in this state?

a. born has the value plus and is increasing

b. born has the value plus en is decreasing

c. born has the value zero and is decreasing

VisiGarp:

- Close the *Dependencies* view
- In the *State transition graph*, select a path between state **8** and state **6**
- A path can be selected by selecting two or more states while keeping down the Shift-button and consequently pressing the *Path* button in the **Select** menu
- In the **View** menu, Press the *Quantity value history button*
- In the *Quantity value history* view, select **number_of1(population1)**
- Then, press the *Draw* button in the **Graph** menu

9. Study the selected path. Which statement about this path is true?

a. the number_of the population changes from state 8 to 1 from zero to max, and from state 2 to 6 in the reverse direction, i.e., from max to zero

b. the number_of the population changes from state 8 to 1 from max to zero, and from state 2 to 6 in the reverse direction, i.e., from zero to max

c. the number_of the population changes in the selected path from max to zero

VisiGarp:

- Close the *Quantity value history* view
- In the **Select** menu, press the *None* button to deselect the path
- Select state **1** and **2**, and press the *Transition history* button, in the **View** menu

10. Why is there no increase of the number_of the population yet in state **1**, and why is there in state **2**?

a. There is less emigration in state 2

b. Born and dead are in balance in state 1, but not in state 2

c. There is no colonization yet in state 1, but there is in state 2

VisiGarp:

- Close the *Transitions for selected states* view

- In the **Select** menu, press the States button
- Select state **3** and open the *Dependencies* view

11. Why does the growth of the population increase?

a. born is equal to dead AND immigration is greater than emigration

b. born is equal to dead AND immigration is smaller than emigration

c. born is equal to dead AND immigration is equal to emigration

VisiGarp:

- Close the *Dependencies* view in state **3**
- Select state **4** and open the *Dependencies* view

12. There is no change in the growth of the population, because

a. born and dead are equal and immigration and emigration are equal

b. born and dead are equal and immigration is greater than emigration

c. born is greater than dead and immigration is smaller than emigration

VisiGarp:

- Close the *Dependencies* view in state **4**

13. In a moment we will start with the next simulation, which describes another situation. But first, we would like to ask you about your opinion of the difficulty level of the questions about the first simulation.

1	2	3	4	5
easy	moderately easy	neutral	moderately difficult	difficult

Scenario 2

VisiGarp:

- In the **File** menu, choose the option *Open Saved Simulation* and then select the document: Scenario2.svst
- In the *State transition graph*, select state **5**, and in the **View** menu, press the *Entities and relations* button

14. Which statement about the Cerrado vegetation is correct?

a. the vegetation consists of 2 populations, that is, populations of grass and shrubs

b. the vegetation consists of 3 populations, that is, populations of grass, shrubs, and trees

c. the vegetation consists of 2 populations, that is, populations of shrubs and trees

15. What is the task of the Cerrado manager within the presented simulation model?

a. to control fire, especially increasing the frequency of fires

b. to merge different flows going to zero

c. to control fire, especially decreasing the frequency of fires

VisiGarp:

- Close the *Entities and relations* view

! Make sure that state **5** is still selected

- In the **View** menu, press the *Dependencies* button
- Tip: In the **Dependencies** menu in the *Dependencies* view for state 5, select the option *None* to get a clear picture

16. Which quantities are relevant for describing a population (for example, the tree population)?

a. number_of, dead, born, and fire_frequency

b. number_of, dead, born, and growth

c. number_of, inflow, outflow, growth, and cover

VisiGarp:

- In the **View** menu in the *Dependencies* view for state 5, turn on the option *Q Spaces*

17. Which values can the number_of a population (for example the grass population) adopt?

a. zero, plus

b. zero, low, medium high and max

c. min, zero and plus

VisiGarp:

- In the **View** menu in the *Dependencies* view for state 5, turn on the option *Derivatives* and the option *Values*

18. What is the value of cover of the Cerrado vegetation? And does cover increase or decrease in this state?

a. cover has the value plus and is decreasing

b. cover has the value low and is increasing

c. cover has the value plus and is increasing

19. What is the value of number_of the tree population? And does number_of increase or decrease in this state?

a. number_of has the value plus and is decreasing

b. number_of has the value low and is increasing

c. number_of has the value high and is decreasing

VisiGarp:

- Close the *Depencies* view for state **5**
- Select a path from state **1** to state **16**
- A path can be selected by selecting two or more states while keeping the Shift-key pressed, and consequently press the *Path* button in the **Select** menu.
- In the **View** menu, press the *Quantity value history* button
- in the *Quantity value history* view, select number_of1(tree_population1), number_of2(shrub_population1), number_of3(grass_population1) while keeping the Shift-key pressed
- Next, press the *Draw* button in the **Graph** menu

20. Study the selected path. Which statement about this path is correct?

a. the number_of the grass and shrub population changes from zero to max and from zero to high, respectively, while the number_of the tree population decreases from max to zero.

b. the number_of the tree- and shrub population changes from zero to max and from zero to high, respectively, while the number_of the grass population decreases from max to zero.

c. the number_of the grass and tree population changes from zero to max and from zero to high, respectively, while the number_of the shrub population decreases from max to zero.

VisiGarp:

- Close the *Quantity value history view*
- In the **Select** menu, press the button *None* to deselect the path, and then press the button *States*
- Select all states in the simulation by pressing the button *All*
- In the **View** menu, press the *Quantity value history* button
- In the *Quantity value history* view, select number_of1(tree_population1), number_of2(shrub_population1), number_of3(grass_population1) while keeping the Shift-key pressed
- Then, press the *Draw* button in the **Graph** menu

21. Study how the values of number_of for the tree, shrub, and grass population compare over all states. Which statement best characterizes these changes over the different states?

a. the tree population decreases, mostly from max to zero or low, while the grass and shrub population mostly increase from zero to max or high.

b. the shrub population decreases, mostly from max to zero, while the grass and tree population increase, mostly from zero to max or high.

c. the grass population decreases, mostly from max to zero, while the shrub and tree population increase, mostly from zero to max or high.

VisiGarp:

- Close the *Quantity value history* view
- In the **Select** menu, press the *None* button to deselect the states
- Then, select state **5**
- In the **View** menu, press the *Dependencies* button
- In the **View** menu of the *Dependencies* view in state **5**, turn on the options *Q Spaces, Values, and Derivatives*

22. What kind of dependency is there between fire frequency and litter in the Cerrado vegetation?

a. When fire frequency decreases, litter also decreases

b. When fire frequency decreases, litter increases

c. There is no direct dependency between these two quantities

23. What kind of dependency is there between litter and soil_temperature in the Cerrado vegetation?

a. When litter increases, soil_temperature also increases

b. When litter increases, soil_temperature decreases

c. There is no direct dependency between these two quantities

24. What kind of dependency is there between cover and litter in the Cerrado vegetation?

a. When cover increases, litter also increases

b. When cover increases, litter decreases

c. There is no direct dependency between these two quantities

25. What kind of dependency is there in the Cerrado vegetation between litter and the factors soil_temperature, light, moisture and nutrient?

a. Changes in litter (*e.g.*, increase) propagate to the same changes in light and soil_temperature (they also increase), and changes in the opposite direction for moisture and nutrient (they decrease)

b. Changes in litter (*e.g.*, increase) propagate to the same changes in moisture and nutrient (they also increase), and changes in the opposite direction for light and soil_temperature (they decrease)

c. Changes in litter (bijv. increase) propagate to the same changes in light and moisture (they also increase), and changes in the opposite direction for soil_temperature and nutrient (they decrease)

26. What can be deduced about the growth of the tree population?

a. the tree population increases, because immigration is equal to emigration, while born is greater than dead

b. the tree population increases, because immigration is greater than emigration, while born is equal to dead

c. the tree population increases, because immigration is greater than emigration, AND born is greater than dead.

27. What can be deduced about the growth of the shrub population?

a. the shrub population increases, because immigration is equal to emigration, while born is greater than dead

b. the shrub population increases, because immigration is greater than emigration, while born is equal to dead

c. the shrub population increases, because immigration is greater than emigration, AND born is greater than dead.

28. The factors of soil_temperature, light, moisture and nutrient have an effect on born and dead of the populations in the Cerrado vegetation. Which of the following statements about the tree population is correct?

a. When litter increases, the composition of soil_temperature, light, moisture and nutrient changes in such a way that born of the tree population (growth of new trees) becomes equal to dead (dying of existing trees)

b. When litter increases, the composition of soil_temperature, light, moisture and nutrient changes in such a way that born of the tree population (growth of new trees) becomes greater than dead (dying of existing trees)

c. When litter increases, the composition of soil_temperature, light, moisture and nutrient changes in such a way that born of the tree population (growth of new trees) becomes smaller than dead (dying of existing trees).

29. The factors of soil_temperature, light, moisture and nutrient have an effect on born and dead of the populations in the Cerrado vegetation. Which of the following statements about the grass population is correct?

a. When litter increases, the composition of soil_temperature, light, moisture and nutrient changes in such a way that the grass population born (growth of new grass) becomes equal to dead (dying of existing grass)

b. When litter increases, the composition of soil_temperature, light, moisture and nutrient changes in such a way that the grass population born (growth of new grass) becomes greater than dead (dying of existing grass)

c. When litter increases, the composition of soil_temperature, light, moisture and nutrient changes in such a way that the grass population born (growth of new grass) becomes smaller than dead (dying of existing grass).

30. In this simulation a manager regulates the fire frequency in the Cerrado vegetation. Which of the following statements is correct? (Note: Try to deduce the answer from the *Dependencies* shown in the *Dependencies* view)

a. When the manager succeeds in decreasing the fire frequency, the grass and shrub population will increase, while the tree population will decrease

b. When the manager succeeds in decreasing the fire frequency, the grass and tree population will increase, while the shrub population will decrease

c. When the manager succeeds in decreasing the fire frequency, the tree and shrub population will increase, while the grass population will decrease

VisiGarp:

- Close the *Dependencies in state 5* view

31. How would you rate the difficulty level of the questions about the second simulation?

1	2	3	4	5
easy	moderately easy	neutral	moderately difficult	difficult

List of Figures

List of Tables

Bibliography

Achinstein, P. Can there be a model of explanation? In D.-H. Ruben, editor, *Explanation*, chapter V, pages 136–159. Oxford University Press, New York, 1993.

Acker, L. H., J. C. Lester, A. F. Souther, and B. W. Porter. Generating coherent explanations to answer students' questions. In H. Burns, J. W. Parlett, and C. L. Redfield, editors, *Intelligent Tutoring Systems: Evolutions in Design*, chapter 7, pages 151–176. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.

Aïmeur, E., H. Dufort, D. Leibu, and C. Frasson. Some justifications for the learning by disturbing strategy. In B. du Boulay and R. Mizoguchi, editors, *Artificial Intelligence in Education: Knowledge and Media in Learning Systems. Proceedings of AI-ED 97 World Conference on Artificial Intelligence in Education*, pages 119–126, Amsterdam/Tokyo, 1997. IOS.

Ainsworth, S. The functions of multiple representations. *Computers and Education*, 33(2-3):131–152, 1999.

Alvarez-Bravo, J. V., J. J.Alvarez-Sanchez, and F. J. Gonzalez-Cabrera. Learning physical concepts using a qualitative approach: a teaching proposal. In J. de Kleer and K. D. Forbus, editors, *Proceedings of QR 2004, the 18th International Workshop on Qualitative Reasoning, August 2-4, 2004*, Evanston, Illinois, 2004. Northwestern University.

Anderson, J. R., C. F. Boyle, A. T. Corbett, and M. W. Lewis. Cognitive modelling and intelligent tutoring. *Artificial Intelligence*, 42:7–49, 1990.

Arens, Y., E. H. Hovy, and M. Vossers. On the knowledge underlying multimedia presentations. In M. T. Maybury, editor, *Intelligent Multimedia Interfaces*, pages 280–306. AAAI Press/MIT Press, Menlo Park (CA), Cambridge (MA), London (England), 1993.

Arnbak, N., T. van Dijk, R. Groen, and R. Van der Werf. Visualisatie van simulatiemodellen. MiniProject Report, Social Science Informatics (SWI), Universiteit van Amsterdam. In Dutch, 2000.

Battista, G. D., P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: algorithms for the visualization of graphs*. Prentice Hall, Upper Saddle River, NJ, 1999.

van Berkum, J. J. A. and T. de Jong. Instructional environments for simulations. *Journal of Education & Computing*, 6(3/4):305–358, 1991.

Bessa Machado, V. *Supporting the Construction of Qualitative Knowledge Models*. PhD thesis, University of Amsterdam, 2004.

Bessa Machado, V. and B. Bredeweg. Investigating the model building process with HOMER. In B. Bredeweg, editor, *Proceedings of the International workshop on Model-based Systems and Qualitative Reasoning for Intelligent Tutoring Systems*, pages 1–13, San Sebastian, Spain, 2002.

Bessa Machado, V. and B. Bredeweg. Building qualitative models with HOMER: A study in usability and support. In P. Salles and B. Bredeweg, editors, *Proceedings of the 17th International workshop on Qualitative Reasoning, QR'03*, pages 39–46, Brasilia, Brazil, 2003.

Bloom, B. *Taxonomy of educational objectives: The classification of educational goals*. Longmans, Green, New York / Toronto, 1956.

Bouwer, A. ArgueTrack: Computer support for argumentation in education. MSc by Research Thesis. Department of Artificial Intelligence, University of Edinburgh, 1999.

Bouwer, A., V. Bessa Machado, and B. Bredeweg. Interactive model building environments. In P. Brna, M. Baker, K. Stenning, and A. Tiberghien, editors, *The Role of Communication in Learning to Model*, chapter 6, pages 155–182. Lawrence Erlbaum Associates, London, 2002.

Bredeweg, B. *Expertise in Qualitative Prediction of Behaviour*. PhD thesis, University of Amsterdam, Amsterdam, 1992.

Bredeweg, B. and K. D. Forbus. Qualitative modeling in education. *AI Magazine*, 24(4):35–46, 2003.

Bredeweg, B. and P. Struss. Current topics in qualitative reasoning (editorial introduction). *AI Magazine*, 24(4):13–16, 2003.

Brown, J., R. R. Burton, and F. Zdybel. A model-driven question-answering system for mixed-initative computer-assisted instruction. *IEEE Transactions on Systems, Man and Cybernetics*, 3:248–257, 1973.

Brown, J. S., R. R. Burton, and J. de Kleer. Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman and J. S. Brown, editors, *Intelligent Tutoring Systems*, New York, 1982. Academic Press.

Bruner, J. S. The act of discovery. *Harvard Educational Review*, 31:21–31, 1961.

Cavalli-Sforza, V. *Constructed vs. received graphical representations for learning about scientific controversy: Implications for learning and coaching*. PhD thesis, Intelligent Systems Program, University of Pittsburgh, 1998.

Cawsey, A. J. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues*. MIT Press, Cambridge, Massachusetts / London, England, 1993.

Cheng, P. C.-H., R. K. Lowe, and M. Scaife. Cognitive science approaches to understanding diagrammatic representations. *Artificial Intelligence Review*, 15:79–94, 2001.

Coffey, J. W., M. J. Carnot, P. J. Feltovich, J. Feltovich, R. R. Hoffman, A. J. Cañas, and J. D. Novak. A summary of literature pertaining to the use of concept mapping techniques and technologies for education and performance support. Technical report submitted to the chief of naval education and training, Pensacola, FL, 2003. Retrieved 8 july, 2004, from www.ihmc.us/users/acanas/Publications/ConceptMapLitReview/IHMC20Review

Collins, A. and A. L. Stevens. A cognitive theory of inquiry teaching. In P. Goodyear, editor, *Teaching Knowledge and Intelligent Tutoring*, chapter 10. Ablex, Norwood, New Jersey, 1991.

Corbett, A. T. and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 5(4):253–278, 1995.

Cox, R. and P. Brna. Supporting the use of external representations in problem solving: The need for flexible learning environments. *Journal of Artificial Intelligence in Education*, 6(2/3):239–302, 1995.

Davis, R. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

Dubbeldam, B. A qualitative model of the greenhouse effect. Master's thesis, Social Science Informatics (SWI), Universiteit van Amsterdam, 2003.

Eisenack, K. and G. Petschel-Held. Graph theoretical analysis of qualitative models in sustainability science. In N. Agell and J. A. Ortega, editors, *QR 2002, Proceedings of the 16th International Workshop on Qualitative Reasoning*, pages 53–60, Sevilla, Spain, 2002. Edicion Digital.

Engelhardt, Y. *The Language of Graphics: A framework for the analysis of syntax and meaning in maps, charts and diagrams*. PhD thesis, University of Amsterdam, 2002.

Falkenhainer, B., K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

Falkenhainer, B. C. and K. D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

Forbus, K. D. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

Forbus, K. D. Qualitative physics: Past, present and future. In H. E. Shrobe, editor, *Exploring Artificial Intelligence*, pages 239–296. Morgan Kaufmann, San Mateo, California, 1988.

Forbus, K. D. The qualitative process engine. In D. S. Weld and J. H. de Kleer, editors, *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann, San Mateo, California, 1990.

Forbus, K. D. Self-explanatory simulators for middle-school science education: A progress report. In A. Farquhar and Y. Iwasaki, editors, *Proceedings of the International Workshop on Qualitative Reasoning*, pages 52–56, Stanford, California, 1996a.

Forbus, K. D. Why computer modeling should become a popular hobby. *D-Lib magazine*, 1996b. Retrieved 19 Jan, 2005, from www.dlib.org/dlib/october96/10forbus.html.

Forbus, K. D. Using qualitative physics to create articulate educational software. *IEEE Expert*, 12(3): 32–41, 1997.

Forbus, K. D., K. Carney, R. Harris, and B. Sherin. A qualitative modeling environment for middle-school students: a progress report. In G. Biswas, editor, *Proceedings of QR 2001, 15th International Workshop on Qualitative Reasoning, St. Mary's University, San Antonio, Texas, 17-18 May 2001*, pages 65–72, Stoughton, WI, 2001. The Printing House.

Forbus, K. D., L. C. Ureel II, K. Carney, and B. Sherin. Qualitative modeling for middle-school students. In J. de Kleer and K. D. Forbus, editors, *Proceedings of QR 2004, 18th International Workshop on Qualitative Reasoning, Evanston, USA, August 2–4, 2004*, pages 81–88, 2004.

Forbus, K. D., P. B. Whalley, J. O. Everett, L. Ureel, M. Brokowski, J. Baher, and S. E. Kuehne. Cyclepad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*, 114(1/2):297–347, 1999.

Foss, C. L. Detecting lost users: Empirical studies on browsing hypertext. Technical report, Computer-Based Learning Unit, University of Leeds, 1989.

Fox, B. *The human tutorial dialogue project*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1993.

van Fraassen, B. C. The pragmatics of explanation. In D.-H. Ruben, editor, *Explanation*, chapter XI, pages 275–309. Oxford University Press, New York, 1993.

Frantz, F. A taxonomy of model abstraction techniques. Technical report, Air Force Research Laboratory, Rome, New York, 1996.

Gautier, P. O. and T. R. Gruber. Generating explanations of device behavior using compositional modeling and causal ordering. In *Proceedings of QR '93, International Workshop on Qualitative Reasoning*, pages 89–97, 1993.

Gentner, D. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2), 1983.

Glasgow, J., N. H. Narayanan, and B. Chandrasekaran, editors. *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. AAAI Press / The MIT Press, Menlo Park, California, 1995.

Goddijn, F. Automatische vraaggeneratie bij kwalitatieve simulaties. Master's thesis, Artificial Intelligence, Universiteit van Amsterdam, 2002. (In Dutch).

Goddijn, F., A. Bouwer, and B. Bredeweg. Automatically generating tutoring questions for qualitative simulations. In P. Salles and B. Bredeweg, editors, *Proceedings of the 17th International workshop on Qualitative Reasoning, QR'03*, pages 87–94, Brasilia, Brazil, 2003.

Goldstein, I. P. The genetic graph: a representation for the evolution of procedural knowledge. In D. Sleeman and J. S. Brown, editors, *Intelligent Tutoring Systems*, pages 51–77. Academic Press, New York, 1982.

Graesser, A. C., K. Wiemer-Hastings, P. Wiemer-Hastings, R. Kreuz, and the Tutoring Research Group. Autotutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*, 1: 35–51, 1999.

Grandbastien, M. Teaching expertise is at the core of its research. *International Journal of Artificial Intelligence in Education*, 10:335–349, 1999. Based on an invited talk presented at the ITS 98 conference.

Groen, R. Supporting model building. Master's thesis, Social Science Informatics (SWI), Universiteit van Amsterdam, 2003.

Harel, D. State charts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

Harel, D. On visual formalisms. In J. Glasgow, N. H. Narayanan, and B. Chandrasekaran, editors, *Diagrammatic Reasoning*, pages 235–271. The MIT Press, Cambridge, Massachusetts, 1995.

Hempel, C. G. Explanation in science and history. In D.-H. Ruben, editor, *Explanation*, chapter 1, pages 17–41. Oxford University Press, New York, 1993. Originally published in R. G. Colodny (ed.), Frontiers of Science and Philosophy, Allen & Unwin and University of Pittsburgh Press, 1962, 7-33.

Hollan, J., E. Hutchins, and L. Weitzman. Steamer: An interactive, inspectable, simulation-based training system. In G. Kearsley, editor, *Artificial intelligence and instruction: applications and methods*, pages 113–134. Addison-Wesley, Reading (Mass), 1987.

Hollan, J. D., E. L. Hutchins, and L. Weizenbaum. STEAMER: an interactive inspectable simulation-based training system. *AI Magazine*, 5:15–27, 2, 1984.

van der Hulst, A. *Cognitive Tools: Two exercises in non-directive support for exploratory learning*. PhD thesis, University of Amsterdam, Amsterdam, 1996.

Iwasaki, Y. and H. A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986.

Jellema, J. Ontwerpen voor ondersteuning: De rol van taakkennis bij ondersteuningsontwerp. Master's thesis, Universiteit van Amsterdam. Sociaal-Wetenschappelijke Informatica, 2000. (In Dutch).

Johnson-Laird, P. *Mental Models*. Harvard University Press, Cambridge, MA, 1983.

van Joolingen, W. R. Qmaps: Qualitative reasoning for simulation learning environments. *International Journal of Artificial Inteligence and Education*, 6(1):67–89, 1995.

van Joolingen, W. R. Cognitive tools to support discovery learning. In B. P. Goettl, H. M. Halff, C. L. Redfield, and V. J. Shute, editors, *Proceedings of ITS '98, the 4th International Conference on Intelligent Tutoring Systems*, page 5, San Antonio, Texas, USA, August 1998. Invited Talk.

van Joolingen, W. R. Cognitive tools for discovery learning. *International Journal of Artificial Inteligence and Education*, 10:385–397, 1999.

Kamps, J. Alleviation of irrelevant knowledge: support knowledge for constructing qualitative models. MSc Thesis. Department of Computer Science, Vrije Universiteit Amsterdam, 1993.

Kim, J. Noncausal connection. *Nous*, 8:41–52, 1974.

Kim, J. Explanatory realism, causal realism, and explanatory exclusion. In D.-H. Ruben, editor, *Explanation*, chapter IX, pages 228–246. Oxford University Press, New York, 1993.

de Kleer, J. and J. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

de Kleer, J. H. and J. S. Brown. Assumptions and ambiguities in mechanistic mental models. In D. Gentner and A. L. Stevens, editors, *Mental Models*, pages 155–190. Lawrence Erlbaum, Hillsdale, 1983.

de Kleer, J. H. and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

Koedinger, K. R., D. D. Suthers, and K. D. Forbus. Component-based construction of a science learning space. *International Journal of Artificial Intelligence in Education*, 10:292–313, 1999.

Koeman, J. An articulate model of the water cycle. Master's thesis, Social Science Informatics (SWI), Universiteit van Amsterdam, 2003.

de Koning, K. *Model-Based Reasoning About Learner Behaviour*. IOS Press, Amsterdam, 1997.

de Koning, K., B. Bredeweg, J. Breuker, and B. Wielinga. Model-based reasoning about learner behaviour. *Artificial Intelligence*, 117:173–229, 2000.

Kuipers, B. Commonsense reasoning about causality: Deriving behaviour from structure. *Artificial Intelligence*, 24:169–203, 1984.

Kuipers, B. Qualitative simulation. *Artificial Intelligence*, 29:289–388, 1986.

Kuipers, B. J. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, Massachusetts, 1994.

Kulpa, Z. Diagrammatic representation and reasoning. *Machine GRAPHICS & VISION*, 3(1/2):77–103, 1994.

Lajoie, S. P. and M. Vivet, editors. *Artificial Intelligence in Education: Open Learning Environments. Proceedings of AI-ED 99 World Conference on Artificial Intelligence in Education*, Amsterdam/Berlin/Oxford/Tokyo/Washington, DC, 1999. IOS.

Lakoff, G. and M. Johnson. *Metaphors we live by*. Chicago: University of Chicago, 1980.

Laplace, P. *A philosophical essay on probabilities*. Dover, New York, 1951. Translated by F. W. Truscott and F. L. Emory. Original work published 1814.

Larkin, J. and H. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11:65–99, 1987.

Leelawong, K., Y. Wang, G. Biswas, N. Vye, J. Bransford, and D. Schwartz. Qualitative reasoning techniques to support learning by teaching. In G. Biswas, editor, *Proceedings of QR 2001, 15th International Workshop on Qualitative Reasoning, St. Mary's University, San Antonio, Texas, 17-18 May 2001*, pages 65–72, Stoughton, WI, 2001. The Printing House.

Lehmann, J. *Causation in Artificial Intelligence and Law: a modelling approach*. PhD thesis, Universiteit van Amsterdam, 2003.

Levelt, W. J. M. Het lineariseringsprobleem van de spreker. *Interdisciplinair Tijdschrift voor Taal- & Tekstwetenschap*, 2(1):1–15, 1982. (In Dutch). Edited translation of The speaker's linearization problem, Philosophical Transactions of the Royal Society London, B 295, 305-315, 1981.

Lewis, D. Causal explanation. In D.-H. Ruben, editor, *Explanation*, chapter VII, pages 182–206. Oxford University Press, New York, 1993.

Lipton, P. Contrastive explanation. In D.-H. Ruben, editor, *Explanation*, chapter VIII, pages 207–227. Oxford University Press, New York, 1993.

Luckin, R. and L. Hammerton. Getting to know me: Helping learners understand their own learning needs through metacognitive scaffolding. In S. A. Cerri, G. Gouarderes, and F. Paraguacu, editors, *Proceedings of ITS 2002, the 6th International Conference on Intelligent Tutoring Systems*, pages 759–771, Biarritz, France and San Sebastian, Spain, 2002. Springer.

Mallory, R. S. *Tools for Explaining Complex Qualitative Simulations*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, 1998.

Mallory, R. S. and B. W. Porter. Explanation structures for complex qualitative behavior graphs. In J. J. Flores, editor, *Proceedings of QR 2000, 14th International Workshop on Qualitative Reasoning, Morelia, Mexico, 5-7 June 2000*, pages 97–103, Morelia, Mexico, 2000. Universidad Michoacana de San Nicolas de Hidalgo.

McKeown, K. R. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27: 1–41, 1985.

Metzler, D. P. and C. J. Martincic. QUE: explanation through exploration. *Expert Systems with Applications*, 15:253–263, 1998.

Mittal, V., S. Roth, J. Moore, J. Mattis, and G. Carenini. Generating explanatory captions for information graphics. In *Proceedings International Joint Conference on Artificial Intelligence, Montreal, Canada, August 1995*, pages 1276–1283, 1995.

Moore, J. D. *Participating in Explanatory Dialogues. Interpreting and Responding to Questions in Context*. MIT Press, Cambridge, Massachusetts / London, England, 1995.

Moore, J. D. Making computer tutors more like humans. *Journal of Artificial Intelligence in Education*, 7(2):181–214, 1996.

Mustapha, S. M. F. D. S., P. Jen-Sen, and S. M. Zain. Application qualitative process theory to qualitative simulation and analysis of inorganic chemical reaction. In N. Agell and J. A. Ortega, editors, *QR 2002, Proceedings of the 16th International Workshop on Qualitative Reasoning*, pages 177–184, Sevilla, Spain, 2002. Edicion Digital.

Neumann, M. and B. Bredeweg. A qualitative model of the nutrient spiraling in lotic ecosystems to support decision makers for river management. In J. de Kleer and K. D. Forbus, editors, *Proceedings of QR 2004, the 18th International Workshop on Qualitative Reasoning, August 2-4, 2004*, Evanston, Illinois, 2004. Northwestern University.

Norman, D. A. *Things that make us Smart: Defending Human Attributes in the Age of the Machine*. Perseus Books, Cambridge, Massachusetts, 1993.

Nuttle, T., B. Bredeweg, and P. Salles. Qualitative reasoning about food webs: Exploring alternative representations. In J. de Kleer and K. D. Forbus, editors, *Proceedings of QR 2004, the 18th International Workshop on Qualitative Reasoning, August 2-4, 2004*, Evanston, Illinois, 2004. Northwestern University.

Pilkington, R. M. Question answering for intelligent on-line help: The process of intelligent responding. *Cognitive Science*, 16(4):455–491, 1992.

Pilkington, R. M., J. R. Hartley, D. Hintze, and D. J. Moore. Learning to argue and arguing to learn: An interface for computer-based dialogue games. *Journal of Artificial Intelligence in Education*, 3 (3):275–295, 1992.

Quignard, M. and M. Baker. Favouring modellable computer-mediated argumentative dialogue in collaborative problem-solving situations. In S. P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education: Open Learning Environments. Proceedings of AI-ED 99 World Conference*, pages 129–136, Amsterdam/Berlin/Oxford/Tokyo/Washington, DC, 1999. IOS.

Reichherzer, T. R., A. J. Cañas, K. M. Ford, and P. J. Hayes. The Giant: A classroom collaborator. In C. Frasson, G. Gouardères, L. W. Johnson, J. Lester, and J. Rickel, editors, *Workshop Notes of the ITS '98 Workshop on Pedagogical Agents*, San Antonio, Texas, August 1998.

Richards, C. J. The fundamental design variables of diagramming. In M. Anderson, B. Meyer, and P. Olivier, editors, *Diagrammatic Representation and Reasoning*, chapter 5, pages 85–102. Springer, London, 2002.

Rickel, J. and B. W. Porter. Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence*, 93:201–260, 1997.

Roth, S. F., M. C. Chuah, S. Kerpedjiev, J. A. Kolojejchick, and P. Lucas. Towards an information visualization workspace: Combining multiple means of expression. *Human-Computer Interaction Journal*, 12(1/2):131–185, 1997.

Ruben, D.-H., editor. *Explanation*. Oxford University Press, New York, 1993.

Rühl, D. Genereren van uitleg bij model-gebaseerde diagnose. Master's thesis, Universiteit van Amsterdam. Sociaal-Wetenschappelijke Informatica, 1997. (In Dutch).

Sachenbacher, M. and P. Struss. AQUA: A framework for automated qualitative abstraction. In G. Biswas, editor, *Proceedings of QR 2001, 15th International Workshop on Qualitative Reasoning, St. Mary's University, San Antonio, Texas, 17-18 May 2001*, pages 5–12, Stoughton, WI, 2001. The Printing House.

Salles, P. *Qualitative Models in Ecology and their Use in Learning Environments*. PhD thesis, University of Edinburgh, 1997.

Salles, P. and B. Bredeweg. Building qualitative models in ecology. In *Proceedings of the International workshop on Qualitative Reasoning, QR'97*, pages 155–164, Italy, June 1997. Istituto di Analisi Numerica C.N.R. Pavia.

Salles, P. and B. Bredeweg. Constructing progressive learning routes through qualitative simulation models in ecology. In G. Biswas, editor, *Proceedings of the 15th International workshop on Qualitative Reasoning, QR'01*, pages 82–89, San Antonio, Texas, May 17-19 2001a. IOS-Press.

Salles, P. and B. Bredeweg. Constructing progressive learning routes through qualitative simulation models in ecology. In J. Moore, G. L. Redfield, and J. Johnson, editors, *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, pages 595–597, Ohmsha, Japan, Osaka, 2001b. IOS-Press.

Salles, P. and B. Bredeweg. A case study of collaborative modelling: building qualitative models in ecology. In U. Hoppe, F. Verdejo, and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*, pages 245–252, Osaka, Japan, 2003. IOS-Press/Ohmsha.

Salles, P., B. Bredeweg, S. Araujo, and W. Neto. Qualitative models of interactions between two populations. *AI Communications*, 16(Issue 4,24):291–308, 2003.

Salles, P., B. Bredeweg, and R. Winkels. Deriving explanations from qualitative models. In B. du Boulay and R. Mizoguchi, editors, *Artificial Intelligence in Education: Knowledge and Media in Learning Systems. Proceedings of AI-ED 97 World Conference on Artificial Intelligence in Education*, pages 474–481, Amsterdam/Tokyo, 1997. IOS.

Salles, P., R. Gauche, and P. Virmond. A qualitative model of the daniell cell for chemical education. In J. de Kleer and K. D. Forbus, editors, *Proceedings of QR 2004, the 18th International Workshop on Qualitative Reasoning, August 2-4, 2004*, Evanston, Illinois, 2004. Northwestern University.

Salmon, W. C. Scientific explanation and the causal structure of the world. In D.-H. Ruben, editor, *Explanation*, chapter III, pages 78–112. Oxford University Press, New York, 1993.

Sinclair, J. M. and R. M. Coulthard. *Towards an Analysis of Discourse: The English used by teachers and pupils*. Oxford University Press, London, 1975.

Sleeman, D. and J. S. Brown, editors. *Intelligent Tutoring Systems*. Academic Press, London, 1982.

Soloway, E., A. Z. Pryor, J. S. Krajcik, S. Jackson, S. J. Stratford, M. Wisnudel, and J. T. Klein. Scienceware's model-it: Technology to support authentic science inquiry. *Technological Horizons on Education*, 25(3):54–56, 1997. Retrieved 19 Jan, 2005, from http://docushare.soe.umich.edu/docushare/dsweb/View/Collection-15.

van Someren, M. W., P. Reimann, H. P. Boshuizen, and T. de Jong, editors. *Learning with Multiple Representations*. Pergamon, Amsterdam, 1998.

Suthers, D. and A. Weiner. Groupware for developing critical discussion skills. In *Computer Supported Cooperative Learning (CSCL '95)*, Bloomington, Indiana, October 1995. Retrieved 19 Jan, 2005, from http://lilt.ics.hawaii.edu/lilt/papers/1995/suthers-weiner-cscl95.pdf.

Suthers, D. D. Answering student queries: Functionality and mechanisms. In C. Frasson, G. Gauthier, and G. I. McCalla, editors, *Proceedings of ITS '92, Second International Conference on Intelligent Tutoring Systems*, volume 608, pages 191–198, Montréal, Canada, 1992. Springer-Verlag.

Suthers, D. D. Representational guidance for collaborative learning. artificial intelligence in education. In H. U. Hoppe, F. Verdejo, and J. Kay, editors, *Proceedings of AI-ED 2003, the 11th International Conference on Artificial Intelligence in Education*, pages 3–10, Amsterdam, 2003. IOS Press. Keynote address.

Suthers, D. D., E. E. Toth, and A. Weiner. An integrated approach to implementing collaborative inquiry in the classroom. In *Computer Supported Cooperative Learning (CSCL '97)*, Toronto, December 1997. Retrieved 19 Jan, 2005, from www.oise.utoronto.ca/cscl/papers/suthers.pdf.

Tjaris, P. Kwalitatieve simulaties als middel tot kennisoverdracht. Master's thesis, Social Science Informatics (SWI), Universiteit van Amsterdam, 2002. In Dutch.

Tufte, E. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 1983.

Tufte, E. *Envisioning Information*. Graphics Press, Cheshire, CT, 1990.

Tufte, E. *Visual Explanations*. Graphics Press, Cheshire, CT, 1997.

Tullos, D. D., M. Neumann, and J. J. Alvarez-Sanchez. Development of a qualitative model for investigating benthic community response to anthropogenic activities. In J. de Kleer and K. D. Forbus, editors, *Proceedings of QR 2004, the 18th International Workshop on Qualitative Reasoning, August 2-4, 2004*, Evanston, Illinois, 2004. Northwestern University.

Tversky, B. Cognitive origins of graphic conventions. In F. T. Marchese, editor, *Understanding images*, pages 29–53. Springer-Verlag, New York, 1995.

Veermans, K. *Intelligent support for discovery learning: Using opportunistic learner modeling and heuristics to support simulation based discovery learning*. PhD thesis, University of Twente, 2002.

Weld, D. S. Reasoning about model accuracy. *Journal of Artificial Intelligence*, 56:255–300, 1992.

Weld, D. S. and J. H. de Kleer. *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, 1990.

Wenger, E. *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann, San Francisco, California, 1987.

van der Werf, R. The use of qualitative modelling and simulation tools in high school education: an engineering study. Master's thesis, Social Science Informatics (SWI), Universiteit van Amsterdam, 2003.

White, B. Y. and J. R. Frederiksen. Causal model progressions as a foundation for learning environments. *Artificial Intelligence*, 42:99–157, 1990.

Winkels, R. *Explorations in Intelligent Tutoring and Help*. PhD thesis, University of Amsterdam, Amsterdam, 1992.

Winkels, R. and B. Bredeweg. Qualitative models in interactive learning environments. *Interactive Learning Environment*, 5:1–134, 1998. Editorial special issue.

Winkels, R. and J. Breuker. Automatic generation of optimal learning routes. In P. Brna, S. Ohlsson, and H. Pain, editors, *Proceedings of AI-ED 93, the World Conference on Artificial Intelligence in Education*, pages 330–337, Edinburgh, 1993. AACE.

Woolf, B. and T. Murray. A framework for representing tutorial discourse. In J. P. McDermott, editor, *Proceedings of IJCAI-87, the 10th International Joint Conference on Artificial Intelligence*, pages 189–192, Milan, Italy, 1987. Morgan Kaufmann.

Summary

This thesis is about the generation of interactive explanations of dynamic phenomena in a domain-independent manner. Qualitative simulations are considered as the basis for generating such kinds of explanations, to be used in interactive learning environments. A qualitative simulation represents the behaviour of a particular system in terms which support conceptual understanding, such as qualitative values and derivatives, and causal dependencies. In this thesis, the GARP framework for qualitative simulation is used. Model primitives in GARP include entities, relations, quantities, values, derivatives, and causal and mathematical dependencies. These are combined in the specification of the initial situation and the specification of model fragments, which represent which information is relevant for particular situations and processes, and under which circumstances. Given a library of model fragments and a description of the initial situation, the qualitative simulation engine generates predictions in the form of a state-transition graph, where each state represents the state of the system at a particular moment or interval in time. The output of GARP, which largely consists of statements in a predicate logic format, is not easy to understand for novices, however. Simulations may incorporate many states, and/or a great amount of information within a state, which makes it difficult to recognize the most important pieces of information. This creates problems when trying to communicate the results of qualitative simulations and their underlying models to users, especially novices. In this thesis, a solution for these problems is presented, both in theoretical terms, and in the form of implemented tools and mechanisms. The thesis involves three research themes which address different parts of the solution: *visualization* of qualitative simulations, *aggregation* of qualitative simulations, and *interactive explanation*.

Visualization of Qualitative Simulations

External representations, such as graphs and other diagrams, are regarded as useful tools in supporting learning about the structure and behaviour of systems. Due to the use of space in meaningful ways, external representations provide a more direct mapping to the structure of the represented knowledge than textual representations, which are essentially linear in nature. External representations may support the user in acquiring an overview of the information, searching for particular pieces of information, and relating multiple pieces of information.

In this thesis, a graphical language is defined for visualizing qualitative simulations, based on a mapping from model primitives to visual primitives, and principles for combining the visual primitives to useful diagrams. Furthermore, a tool has been designed and implemented which uses this graphical language to generate such diagrams automatically, based on a qualitative simulation in the GARP format. This tool, named VisiGarp, incorporates several kinds of diagrams, including the entities and relations view, the dependencies view, the quantity value history view, the model fragment view, several is-a hierarchies, and the state-transition diagram. VisiGarp has been evaluated by students and teachers, using a setup which included exercises to guide students through material about Brazilian Cerrado ecology, the example domain used in this thesis. The results suggest that students (without expertise about ecology or qualitative reasoning) are able to use VisiGarp to find the answers

to exercise questions, and that they learn something about the domain during the course of the experiment. VisiGarp is currently in use by researchers, teachers and students in universities in Amsterdam (The Netherlands), Brasilia (Brazil), Giessen and Jena (Germany), and Segovia (Spain). The VisiGarp tool is completely domain-independent, and has been utilized in domains such as Brazilian Cerrado ecology, river stream ecology, biology, physics, and electrochemistry.

Aggregation of Qualitative Simulations

When the domain model includes more than a few elements, the resulting simulation can grow quite large, in terms of the number of states and transitions, as well as the amount of facts per state. In such cases, it becomes impractical to investigate the simulation completely, and the need occurs to aggregate the information in the simulation.

In this thesis, a suite of aggregation techniques is introduced which reduces the complexity of qualitative simulations while retaining the most important information elements. The aggregation techniques are based on four generic methods of aggregation (selection, chunking, generalization, and grouping), which are applied to five different levels of aggregation (system state level, local level, path segment level, path level, and global level). On the system state level, which represents the system in a particular state, causal dependencies are classified so that the effective ones can be selected, while ignoring the inactive submissive, and balanced ones. On the local level, adjacent states are compared to interpret differences between facts as *events* in different categories (value and derivative events, (in)equality events, correspondence events, causal effects events, structural events, and model fragments events). On the path segment level, chunks of states are recognized which represent *necessary* sequences of behaviour. On the path level, cycles are recognized and additional chunking of events is performed where possible, representing *possible* sequences of behaviour. On the global level, alternative behaviours are considered to perform *transitive reduction*, *aggregation of alternative orderings*, and *aggregation of sequence* on the state-transition graph. Applying a combination of these techniques on the example simulation representing possible developments of the Cerrado results in a reduction of 19 states, 47 transitions, 869 paths, and 4800 facts in the original simulation to 6–11 states, 6–14 transitions, 2–9 paths, and 480 events. This reduction greatly facilitates communication of the relevant information to users who want to learn about the behaviour of the simulated system. The work presented here extends existing approaches to aggregating qualitative simulations by taking into account a richer set of event categories, and a conceptually grounded set of aggregation levels and techniques, without the need of specifying user interests beforehand.

Interactive Explanation

In order to generate effective explanations, it is necessary to consider the didactic goals they should address, the didactic means that should be used in communication with a user, and the didactic style in which they are incorporated.

In this thesis, a classification is presented of the didactic goals that should be addressed in a qualitative simulation-based learning environment. This includes the terminology of qualitative simulation, the generic domain knowledge (domain vocabulary, entity is-a hierarchy, and model fragments), and the simulated behaviour (recognizing the behaviour, as well as predicting and explaining it). The didactic means needed to communicate these didactic goals include the aforementioned diagrammatic representations employed by VisiGarp, but also textual means, such as textual descriptions of facts and events, queries, causal explanations, contrastive explanations, and exercise questions. Didactic styles incorporated in an interactive learning environment can vary along the dimension of who has the initiative–the system or the student. In a system-initiated tutoring style, the system presents material following a didactic plan, asks tutoring questions and provides explanations. In a student-initiated exploration style, the student is in control, deciding on which topics to investigate, and posing queries to the system. Both styles require a way of specifying topics of interest. In this thesis, a topic is defined as the combination of a particular subsystem, a particular time-frame, and particular information types of interest.

All these considerations have led to the development of a prototype interactive learning environment, which is intended to communicate the contents of qualitative models and simulation results to users, in particular students. This prototype, called WiziGarp, extends the functionalities of VisiGarp by utilizing the aggregation techniques to simplify qualitative simulations, and by incorporating diagrams as well as the textual means of communication described above. Furthermore, WiziGarp includes ways to focus on a particular topics, which can be used for didactic planning in a system-initiated style, or free exploration in a student-initiated style of interaction. WiziGarp allows any combination of the two didactic styles, but in this thesis, two interaction scenarios are presented to exemplify both ends of the didactic spectrum. This functionality makes WiziGarp a very powerful and flexible tool for supporting learning and teaching with qualitative simulations.

Future Work

Future work will address further evaluation of VisiGarp and WiziGarp in educational settings, interaction between multiple users who are learning collaboratively, integration with tools for model building, and incorporating modelling and diagnosis of student behaviour to improve the didactic planning.

Samenvatting

Dit proefschrift betreft de generatie van interactieve uitleg over dynamische verschijnselen op een domeinonafhankelijke manier. Kwalitatieve simulaties worden beschouwd als de basis voor het genereren van zulke uitleg voor gebruik in interactieve leeromgevingen. Een kwalitatieve simulatie representeert het gedrag van een bepaald systeem in termen die conceptueel begrip ondersteunen, zoals kwalitatieve waarden en afgeleiden, en causale afhankelijkheden. In dit proefschrift wordt gebruik gemaakt van het GARP-raamwerk voor kwalitatief simuleren. Modelprimitieven in GARP omvatten entiteiten, relaties, kwantiteiten, waarden, afgeleiden en causale en wiskundige afhankelijkheden. Deze worden gecombineerd in de beschrijving van de beginsituatie en de beschrijving van modelfragmenten. Modelfragmenten specificeren welke informatie relevant is voor bepaalde situaties en processen, en onder welke omstandigheden. Gegeven een bibliotheek van modelfragmenten en een beschrijving van de beginsituatie genereert de kwalitatieve simulator voorspellingen in de vorm van een toestandsgraaf waarbij elke knoop de toestand van het systeem beschrijft op een bepaald moment of interval in de tijd. De output van GARP, die grotendeels bestaat uit statements in een predikatenlogica-formaat, is echter niet gemakkelijk te begrijpen voor onervaren gebruikers. Simulaties kunnen vele toestanden bevatten en/of een grote hoeveelheid informatie binnen een toestand, wat het moeilijk maakt om de belangrijkste informatie-elementen te herkennen. Dit levert problemen op zodra men probeert de resultaten van kwalitatieve simulaties te communiceren aan gebruikers, in het bijzonder onervaren gebruikers. In dit proefschrift wordt een oplossing voor deze problemen gepresenteerd, zowel in theoretische termen als in de vorm van geïmplementeerde hulpmiddelen en mechanismen. Het proefschrift omvat drie onderzoeksthema's die leiden tot verschillende delen van de oplossing: *visualisatie* van kwalitatieve simulaties, *aggregatie* van kwalitatieve simulaties en *interactieve uitleg*.

Visualisatie van Kwalitatieve Simulaties

Externe representaties, zoals grafen en andere diagrammen, worden beschouwd als nuttige hulpmiddelen ter ondersteuning van het leren over de structuur en het gedrag van systemen. Door op een betekenisvolle manier gebruik te maken van ruimte leveren externe representaties een meer directe mapping van de structuur van de gepresenteerde kennis dan textuele representaties, die in essentie lineair zijn. Externe representaties kunnen de gebruiker helpen bij het verkrijgen van een overzicht van de informatie, het zoeken naar bepaalde informatie-elementen, en het verbinden van meerdere informatie-elementen.

In dit proefschrift wordt een grafische taal gedefinieerd voor het visualiseren van kwalitatieve simulaties gebaseerd op een mapping van modelprimitieven naar visuele primitieven en principes om de visuele primitieven te combineren tot nuttige diagrammen. Bovendien is een software-tool ontworpen en geïmplementeerd die deze grafische taal gebruikt om zulke diagrammen automatisch te genereren op basis van een kwalitatieve simulatie in het GARP-formaat. Deze tool, genaamd VisiGarp, genereert diverse soorten diagrammen, waaronder het entiteiten en relaties-diagram, het afhankelijkheden-diagram, het kwantiteit-waardehistorie-diagram, het modelfragment-diagram, verscheidene type-hiërarchieën, en het toestandsdiagram. VisiGarp is geëvalueerd door studenten en

docenten waarbij de methode oefeningen bevatte om studenten te leiden door materiaal over de ecologie van de Braziliaanse Cerrado, het voorbeelddomein in dit proefschrift. De resultaten wijzen erop dat studenten (zonder expertise over ecologie of kwalitatief redeneren) in staat zijn om met behulp van VisiGarp antwoorden te vinden op oefenvragen en dat zij iets leren over het domein gedurende het experiment. VisiGarp is inmiddels in gebruik genomen door onderzoekers, docenten en studenten binnen universiteiten in Amsterdam (Nederland), Brasilia (Brazilië), Giessen en Jena (Duitsland) en Segovia (Spanje). De VisiGarp-tool is geheel domeinonafhankelijk en is gebruikt in domeinen als Braziliaanse Cerrado-ecologie, ecologie van rivieren, biologie, natuurkunde en elektrochemie.

Aggregatie van Kwalitatieve Simulaties

Als het domeinmodel meer dan een paar elementen bevat kan de resulterende simulatie vrij groot worden, zowel in termen van het aantal toestanden en transities als het aantal feiten per toestand. In zulke gevallen wordt het lastig om de gehele simulatie te onderzoeken en ontstaat de behoefte om de informatie in de simulatie te aggregeren.

In dit proefschrift wordt een serie van aggregatietechnieken geïntroduceerd die de complexiteit van kwalitatieve simulaties reduceert terwijl de belangrijkste informatie-elementen behouden blijven. De aggregatietechnieken zijn gebaseerd op vier generieke methoden van aggregatie (selecteren, samenvoegen, generaliseren, en groeperen) die worden toegepast op vijf verschillende niveaus van aggregatie (systeemtoestand-niveau, lokaal niveau, padsegment-niveau, pad-niveau en globaal niveau). Op het systeemtoestand-niveau, dat het systeem in een bepaalde toestand weergeeft, worden causale afhankelijkheden geclassificeerd zodat de effectieve kunnen worden geselecteerd terwijl de inactieve, submissieve, en gebalanceerde buiten beschouwing worden gelaten. Op het lokale niveau worden opeenvolgende toestanden vergeleken om verschillen tussen feiten te interpreteren als *gebeurtenissen* in verschillende categorieën (waarden en afgeleiden, ongelijkheden, correspondenties, causale effecten, structuur en modelfragmenten). Op het padsegment-niveau worden opeenvolgingen van toestanden herkend die *noodzakelijke* gedragssequenties weergeven. Op het pad-niveau, dat *mogelijke* gedragssequenties weergeeft, worden cykels herkend en vindt waar mogelijk additionele samenvoeging van gebeurtenissen plaats. Op het globale niveau worden alternatieve gedragingen beschouwd om vervolgens *transitieve reductie*, *aggregatie van alternatieve ordeningen* en *aggregatie van sequentie* uit te voeren op het toestandsdiagram. Toepassing van een combinatie van deze technieken op de voorbeeldsimulatie, die mogelijke ontwikkelingen in de Cerrado weergeeft, resulteert in een reductie van 19 toestanden, 47 transities, 869 paden en 4800 feiten in de originele simulatie tot 6–11 toestanden, 6–14 transities, 2–9 paden en 480 gebeurtenissen. Deze reductie vergemakkelijkt de communicatie van de relevante informatie aan gebruikers die willen leren over het gedrag van het gesimuleerde systeem. Het onderzoek dat hier wordt gepresenteerd is een verbetering ten opzichte van bestaande benaderingen van aggregatie van kwalitatieve simulaties doordat een rijkere set van typen gebeurtenissen in ogenschouw wordt genomen, evenals een conceptueel onderbouwde serie van aggregatieniveaus en technieken, waarbij het niet meer noodzakelijk is dat de gebruiker vooraf interesses specificeert.

Interactieve Uitleg

Om effectieve uitleg te kunnen genereren is het noodzakelijk om te kijken naar de didactische doelen die bereikt moeten worden, de didactische middelen die ingezet moeten worden in de communicatie met een gebruiker, en de didactische stijl waarin deze worden ingebed.

In dit proefschrift wordt een classificatie gepresenteerd van de didactische doelen die verwezenlijkt moeten worden in een kwalitatieve simulatie-gebaseerde leeromgeving. Deze classificatie omvat de terminologie van kwalitatieve simulatie, de generieke domeinkennis (domeinvocabulaire, entiteiten-type-hiërarchie en modelfragmenten) en het gesimuleerde gedrag (zowel het herkennen van het gedrag, als het voorspellen en verklaren ervan). De didactische middelen benodigd voor de communicatie van deze didactische doelen omvatten de eerdergenoemde diagrammatische representaties uit VisiGarp, maar ook tekstuele middelen, zoals tekstuele beschrijvingen van feiten en gebeurtenissen, vragen, causale verklaringen, contrastieve uitleg en oefeningen. Didactische stijlen in een interactieve leeromgeving kunnen variëren op de dimensie gekenmerkt door wie het initiatief heeft–het systeem of de student. In een systeemgeïnitieerde *tutor*-stijl presenteert het systeem materiaal volgens een didactisch plan, stelt didactische vragen en geeft uitleg. In een student-geïnitieerde exploratie-stijl is de controle in handen van de student, die beslist welke onderwerpen worden behandeld en vragen stelt aan het systeem. Beide stijlen vereisen een manier om te specificeren wat belangrijke onderwerpen zijn. In dit proefschrift wordt een onderwerp gedefinieerd als de combinatie van een bepaald subsysteem, een bepaalde tijdsspanne en bepaalde gewenste informatietypen.

Al deze overwegingen hebben geleid tot de ontwikkeling van een prototype interactieve leeromgeving dat tot doel heeft de inhoud van kwalitatieve modellen en simulatieresultaten te communiceren aan gebruikers, in het bijzonder studenten. Dit prototype, genaamd WiziGarp, breidt de functionaliteit van VisiGarp uit door het gebruik van de aggregatietechnieken om kwalitatieve simulaties te vereenvoudigen en door naast diagrammen ook de hierboven beschreven tekstuele communicatiemiddelen te gebruiken. Bovendien kent WiziGarp manieren om te focussen op bepaalde onderwerpen, die gebruikt kunnen worden voor didactische planning in een systeem-geïnitieerde stijl, of vrije exploratie in een student-geïnitieerde stijl van interactie. WiziGarp maakt elke combinatie van deze twee didactische stijlen mogelijk, maar in dit proefschrift worden twee interactie-scenario's gepresenteerd om beide uiteinden van het didactische spectrum te illustreren. Deze functionaliteit maakt WiziGarp een zeer krachtig en flexibel hulpmiddel voor het ondersteunen van leren en onderwijzen met kwalitatieve simulaties.

Toekomstig Onderzoek

Toekomstig onderzoek zal zich richten op verdere evaluatie van VisiGarp en WiziGarp in onderwijssettings, interactie tussen meerdere gebruikers die gezamenlijk leren, integratie met hulpmiddelen voor modelbouw en de toepassing van het modelleren en diagnosticeren van het gedrag van studenten ter verbetering van de didactische planning.

SIKS Dissertatiereeks

1998-1	Johan van den Akker (CWI) DEGAS - An Active, Temporal Database of Autonomous Objects
1998-2	Floris Wiesman (UM) Information Retrieval by Graphically Browsing Meta-Information
1998-3	Ans Steuten (TUD) A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
1998-4	Dennis Breuker (UM) Memory versus Search in Games
1998-5	E.W.Oskamp (RUL) Computerondersteuning bij Straftoemeting
1999-1	Mark Sloof (VU) Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products
1999-2	Rob Potharst (EUR) Classification using decision trees and neural nets
1999-3	Don Beal (UM) The Nature of Minimax Search
1999-4	Jacques Penders (UM) The practical Art of Moving Physical Objects
1999-5	Aldo de Moor (KUB) Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
1999-6	Niek J.E. Wijngaards (VU) Re-design of compositional systems

1999-7	David Spelt (UT) Verification support for object database design
1999-8	Jacques H.J. Lenting (UM) Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.
2000-1	Frank Niessink (VU) Perspectives on Improving Software Maintenance
2000-2	Koen Holtman (TUE) Prototyping of CMS Storage Management
2000-3	Carolien M.T. Metselaar (UvA) Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.
2000-4	Geert de Haan (VU) ETAG, A Formal Model of Competence Knowledge for User Interface Design
2000-5	Ruud van der Pol (UM) Knowledge-based Query Formulation in Information Retrieval.
2000-6	Rogier van Eijk (UU) Programming Languages for Agent Communication
2000-7	Niels Peek (UU) Decision-theoretic Planning of Clinical Patient Management
2000-8	Veerle Coup (EUR) Sensitivity Analyis of Decision-Theoretic Networks
2000-9	Florian Waas (CWI) Principles of Probabilistic Query Optimization
2000-10	Niels Nes (CWI) Image Database Management System Design Considerations, Algorithms and Architecture
2000-11	Jonas Karlsson (CWI) Scalable Distributed Data Structures for Database Management

2001-1	Silja Renooij (UU) Qualitative Approaches to Quantifying Probabilistic Networks
2001-2	Koen Hindriks (UU) Agent Programming Languages: Programming with Mental Models
2001-3	Maarten van Someren (UvA) Learning as problem solving
2001-4	Evgueni Smirnov (UM) Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets
2001-5	Jacco van Ossenbruggen (VU) Processing Structured Hypermedia: A Matter of Style
2001-6	Martijn van Welie (VU) Task-based User Interface Design
2001-7	Bastiaan Schonhage (VU) Diva: Architectural Perspectives on Information Visualization
2001-8	Pascal van Eck (VU) A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
2001-9	Pieter Jan 't Hoen (RUL) Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
2001-10	Maarten Sierhuis (UvA) Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design
2001-11	Tom M. van Engers (VUA) Knowledge Management: The Role of Mental Models in Business Systems Design
2002-01	Nico Lassing (VU) Architecture-Level Modifiability Analysis

2002-02	Roelof van Zwol (UT) Modelling and searching web-based document collections
2002-03	Henk Ernst Blok (UT) Database Optimization Aspects for Information Retrieval
2002-04	Juan Roberto Castelo Valdueza (UU) The Discrete Acyclic Digraph Markov Model in Data Mining
2002-05	Radu Serban (VU) The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents
2002-06	Laurens Mommers (UL) Applied legal epistemology; Building a knowledge-based ontology of the legal domain
2002-07	Peter Boncz (CWI) Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
2002-08	Jaap Gordijn (VU) Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
2002-09	Willem-Jan van den Heuvel(KUB) Integrating Modern Business Applications with Objectified Legacy Systems
2002-10	Brian Sheppard (UM) Towards Perfect Play of Scrabble
2002-11	Wouter C.A. Wijngaards (VU) Agent Based Modelling of Dynamics: Biological and Organisational Applications
2002-12	Albrecht Schmidt (Uva) Processing XML in Database Systems
2002-13	Hongjing Wu (TUE) A Reference Architecture for Adaptive Hypermedia Applications

2002-14	Wieke de Vries (UU) Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems
2002-15	Rik Eshuis (UT) Semantics and Verification of UML Activity Diagrams for Workflow Modelling
2002-16	Pieter van Langen (VU) The Anatomy of Design: Foundations, Models and Applications
2002-17	Stefan Manegold (UvA) Understanding, Modeling, and Improving Main-Memory Database Performance
2003-01	Heiner Stuckenschmidt (VU) Ontology-Based Information Sharing in Weakly Structured Environments
2003-02	Jan Broersen (VU) Modal Action Logics for Reasoning About Reactive Systems
2003-03	Martijn Schuemie (TUD) Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
2003-04	Milan Petkovic (UT) Content-Based Video Retrieval Supported by Database Technology
2003-05	Jos Lehmann (UvA) Causation in Artificial Intelligence and Law - A modelling approach
2003-06	Boris van Schooten (UT) Development and specification of virtual environments
2003-07	Machiel Jansen (UvA) Formal Explorations of Knowledge Intensive Tasks
2003-08	Yongping Ran (UM) Repair Based Scheduling
2003-09	Rens Kortmann (UM) The resolution of visually guided behaviour

2003-10	Andreas Lincke (UvT) Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
2003-11	Simon Keizer (UT) Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
2003-12	Roeland Ordelman (UT) Dutch speech recognition in multimedia information retrieval
2003-13	Jeroen Donkers (UM) Nosce Hostem - Searching with Opponent Models
2003-14	Stijn Hoppenbrouwers (KUN) Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
2003-15	Mathijs de Weerdt (TUD) Plan Merging in Multi-Agent Systems
2003-16	Menzo Windhouwer (CWI) Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
2003-17	David Jansen (UT) Extensions of Statecharts with Probability, Time, and Stochastic Timing
2003-18	Levente Kocsis (UM) Learning Search Decisions
2004-01	Virginia Dignum (UU) A Model for Organizational Interaction: Based on Agents, Founded in Logic
2004-02	Lai Xu (UvT) Monitoring Multi-party Contracts for E-business
2004-03	Perry Groot (VU) A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
2004-04	Chris van Aart (UvA) Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek(UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UvA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18	Vania Bessa Machado (UvA) Supporting the Construction of Qualitative Knowledge Models
2004-19	Thijs Westerveld (UT) Using generative probabilistic models for multimedia retrieval
2004-20	Madelon Evers (Nyenrode) Learning from Design: facilitating multidisciplinary design teams
2005-01	Floor Verdenius (UvA) Methodological Aspects of Designing Induction-Based Applications
2005-02	Erik van der Werf (UM) AI techniques for the game of Go
2005-03	Franc Grootjen (RUN) A Pragmatic Approach to the Conceptualisation of Language
2005-04	Nirvana Meratnia (UT) Towards Database Support for Moving Object data
2005-05	Gabriel Infante-Lopez (UvA) Two-Level Probabilistic Grammars for Natural Language Parsing
2005-06	Pieter Spronck (UM) Adaptive Game AI
2005-07	Flavius Frasincar (TUE) Hypermedia Presentation Generation for Semantic Web Information Systems
2005-08	Richard Vdovjak (TUE) A Model-driven Approach for Building Distributed Ontology-based Web Applications
2005-09	Jeen Broekstra (VU) Storage, Querying and Inferencing for Semantic Web Languages
2005-10	Anders Bouwer (UvA) Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments