VisiGarp: Graphical Representation of Qualitative Simulation Models

Anders Bouwer & Bert Bredeweg

Department of Social Science Informatics, University of Amsterdam
Roetersstraat 15, 1018 WB, Amsterdam, The Netherlands
E-mail: {anders, bert}@swi.psy.uva.nl

Abstract

Qualitative simulation models can play a useful role in computer-based learning environments, since they explicitly represent the domain knowledge required for causal reasoning about system structure and behaviour. However, the amount of information in a simulation is often large, which makes it difficult to transform the computer's reasoning trace to effective explanations. Besides textual explanations, graphical representations play an important role in the communication of knowledge. For example, state-transition diagrams, causal networks, structural hierarchies, tables and graphs all use different visual primitives to create a context for communication, and to denote relationships between the different kinds of entities. Our goal is to develop generic mechanisms for automatic visualization of qualitative simulation models, and to integrate these into a framework for generating multimedia (text and graphics) explanations. Currently, we have developed an interactive model inspection tool, VisiGarp, which automatically generates several kinds of diagrams from a qualitative simulation model; a mouse and menu interface allows flexible switching between the different views on the model, providing both overview and detail. Further work includes layout optimization, aggregation and abstraction mechanisms, and integration of graphics and text generation.

Introduction

Qualitative simulation, based on qualitative reasoning (QR) theory and techniques, employs models of the domain which encode the system studied explicitly, in terms of structure, parameters and causal dependencies between them. Because the knowledge in these models is represented in a generic way, different simulation scenarios can be run with the same model, by adjusting parameters or system elements which trigger different parts of the model (model fragments). Furthermore, model fragments for certain kinds of systems, situations, or processes can be reused in other, similar domains which reduces development costs for future applications.

Besides these advantages for the development process, qualitative simulation models offer important educational advantages. First, by using qualitative information rather than (or in addition to) numerical data, students are stimulated to think about the distinctions that matter. Second, the program's predictions about system behaviour are based on analysis of structural descriptions of the system, which are accessible to students. Therefore, students are empowered to learn to do the same – relating behaviour to structure. Finally, the compositional nature of the model library allows a generic form of didactic planning within domains, e.g., by shifting attention from a model fragment to a more specific subtype model fragment, or to a set of model fragments which are conditional for it. This paper will focus on the first two of these potential advantages, arguing that to fulfil this potential, the communication bottleneck has to be broken.

Qualitative Simulation Models

Qualitative simulation models are meant to represent the essential characteristics of a system's structure and its behaviour (*e.g.*, Weld & Kleer 1990). In our research, we use GARP as our qualitative simulation engine (Bredeweg 1992). The GARP framework contains the following building blocks. First, a representation of the (physical) entities as parts of the system studied, together with their structural relations. Second, a representation of time-varying properties in terms of quantities (the parameters of the system) and quantity spaces. A quantity space specifies the relevant possible values of a quantity in terms of alternating points and intervals (e.g., zero, plus, max, for a parameter which can have the value zero, max, or something in between). Third, a representation of all kinds of dependencies between parameters and parameter values, such as influences and proportional relationships.

Using these building blocks, initial situations (scenarios) can be specified as well as model fragments, encapsulating knowledge about specific processes, or situations. Scenarios usually consist of a structural description of the system and some initial values for certain parameters. Model fragments are rule-like, in terms of having conditions and consequences. The former specify the structural constraints and the specific quantity conditions that must hold for the model fragment to be applicable. The consequences specify the behavioural features that can be derived. An important part of this is the specification of the causal model underlying the behaviour.

Given a library of model fragments and a scenario, the simulation engine derives the system's behaviour in terms of processes becoming active or inactive, and consequently, values of parameters changing. When a parameter changes significantly (i.e., it reaches another value in its quantity space), or when it changes in relation to another parameter (i.e., after two quantities being equal, one becomes greater), a qualitative state transition occurs. For the resulting successor states, the whole process of applying model fragments and deriving changes continues, until no more successor states can be found.

Using simulation models in learning environments

How to use simulation models in computer-supported educational settings? This depends on the kinds of learning goals to be supported. If the goal is to learn how to operate (*e.g.*, Hollan et al. 1987), or design a system (*e.g.*, see Forbus et al. 1999), the simulation should be faithful to reality in all ways relevant to the task. These educational goals are outside the scope of this research, however. The learning goals we do address relate to understanding a system's behaviour, in terms of making predictions, and explaining events. A computer-based learning environment should support students in inspecting the model components, and simulation results. To guide their explorations, students will need to state hypotheses, check their predictions, and construct causal explanations for events taking place in the simulation (*e.g.*, see Joolingen & Jong 1991). Given a certain simulation scenario in its initial state, an exercise could be to predict possible changes of several parameters, and in what order they might occur. In order to test the hypothesis, the student can run a complete simulation—an envisionment. By looking at the state transitions, and the order in which they occur in the behaviour graph, the student can test the accuracy of his/her predictions. A common mistake is to consider too few possibilities, when in fact underdetermination leads to multiple branches– hence, unforeseen results.

Qualitative simulation supports the construction of explanations by allowing access to the causal and structural relationships underlying changes in the simulation. Contrary to events in the real world, events happening in the simulation can always be traced back to a starting state, providing at least a kernel for explanation.

The idea is that by interacting with the simulation, possibly working through a set of assignments, students will eventually understand the important characteristics of the system's behaviour in reality. When the behaviour of a simple partial system is understood successfully, one can progress to more complex models and scenarios, in terms of number of components, the amount of detail, or the level of generality (*e.g.*, White & Frederiksen 1990).

Ontology and principles of visualization

How to communicate the contents of qualitative simulation models to a user? The output generated by the GARP simulation engine consists of a large amount of complex propositional statements in Prolog code format, which is hard to read, search, and oversee, except for experienced knowledge engineers. However, the knowledge contained in the qualitative simulation models is highly structured. This structure can be exploited to generate appropriate visualizations, surpassing the possibilities of linear text. Graphical representations, such as block diagrams, trees and graphs can facilitate search, recognition and inference processes (Larkin & Simon 1995), which helps relating and comparing different information elements. These visualisations make structural aspects of knowledge explicit, which can facilitate internalisation of complex concepts.

Our visualisation ontology consists of circular, rectangular and oval shapes of variable sizes for different kinds of entities, and lines, arrows, inclusion, ordering, and indentation for different kinds of relations. In contrast to other simulation-based learning environments (*e.g.*, Hollan et al. 1987; Forbus et al. 1999), our visualization approach is generic and does not use domain-specific pictures or symbols. Instead, abstract shapes are used with text labels to denote specific entities and relation types.

These visual elements are combined in different ways to form views: particular visualizations supporting specific reasoning tasks. The design of these visualizations is based on the following principles:

- Information related to a specific entity is displayed within the visual element representing that entity, using the container-metaphor as much as possible.
- Information related to multiple entities is displayed as a separate visual element, connecting the entities involved, as much as possible.
- 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.
- Information which is only meaningful in a context should be displayed within that context, highlighted if necessary.
- If an information element is shown in a context in which further detail is not necessary, a simple textual label will suffice.
- Time-ordered information is displayed with time progressing along the x-axis as much as possible.

The benefits of our approach are that it does not require domain-specific knowledge, and the representations are more flexible, which is useful in diagrammatic reasoning. Also, the fact that they are more abstract can help when students are learning to model, generalizing insights, or need to transfer knowledge from one domain to another (*e.g.*, see Falkenhainer et al., 1989). It is important to note, however, that some domains (e.g., electronics) already have standardized visual languages with domain-specific symbols; our more abstract representations are not intended to replace these specialized visualizations.

VisiGarp: Visualization of Qualitative Models

Based on the ontology and principles discussed in the previous section, we have designed a tool, VisiGarp, which supports investigation of the simulation model and results.

View	**Information displayed and characteristics of the visualization**
Entities and attributes	Block diagram showing the elementary structure of the system in terms of entities and attribute 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
State-transition diagram	Shows progress and allows control of simulation in terms of states, possible terminations and completed transitions. 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 type of transition, the conditions leading to it, its results, and the status of the transition (open, terminated, ordered, or closed)
Parameter relations	Shows parameters, grouped within their entities, and the causal and mathematical relationships between different parameters, as a higraph
Parameter values	Shows parameter values and their derivatives for a specific state in tabular format
Parameter value history	Displays a graph for the values of a selected parameter 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 parameter relations view, with the contents of the particular model fragment highlighted in colour (blue for conditions, red 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 previous one, but this one shows which model fragments are conditional for which other model fragments

Table 1. Description of all views available in VisiGarp.

VisiGarp offers a number of views, each of which focuses on certain kinds of information, while using others to form the context, or to provide links to more detailed information. For each view, we have designed a mapping from the ontology of qualitative simulation to the ontology of visualisation primitives, to facilitate specific reasoning tasks (identifying, searching, counting, associating, and sequencing) and interaction types (reading, selecting, dragging, resizing, etc). Multiple views can be opened simultaneously, allowing users to navigate between global overviews and more detailed descriptions and to switch between different types of reasoning. An overview of all views currently available in VisiGarp is shown in table 1. To highlight the important characteristics of the different visualizations, some views will be discussed in more detail.

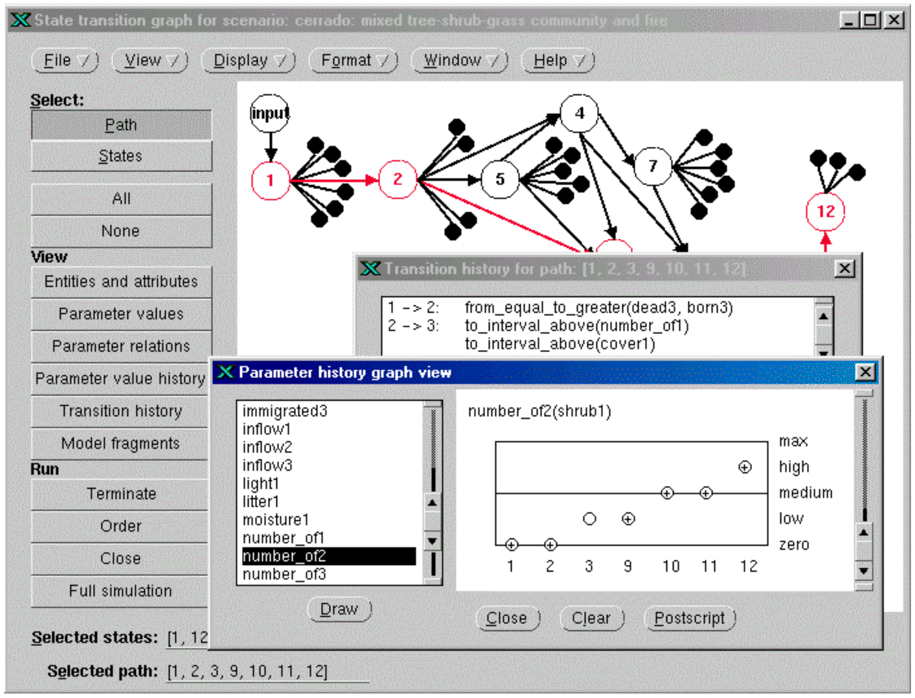
The state-transition graph, or behaviour graph, shows an overview of the progress of the simulation in terms of states and state transitions (see figure 1). This can be compared to the kinds of diagrams generated by the QSIM software (Kuipers, 1994). An interesting difference is that in their approach 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 a 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. And although domain experts and researchers (target users of QSIM) may be interested more in the possible outcomes of the simulation, for our target users (undergraduate students) it is important to be able to refer to specific states, because VisiGarp offers a lot of conceptual information associated with each state.

Alongside the diagram in figure 1, buttons are provided to control the simulation: a state can be selected to pursue the simulation in one direction only, or, alternatively, all branches can be pursued until no further progress is possible (the Full Simulation button). Terminations, which have not (yet) resulted in a state transition, are shown as tiny circles, connected to their originating state. They can also be hidden, to simplify the display.

States, terminations and transitions can be investigated in more detail by selecting them and clicking one of the other view buttons. When two or more states are selected in Path-mode, a path through these states (if one exists) is automatically selected. This is especially useful for the transition history and the parameter value history views, since they show behavioural aspects of multiple transitions/states in a single screen.

The transition history button pops up a small screen with a short textual description of all terminations/transitions in the selected path, or connected to the selected states (partly visible in figure 1). This gives a brief overview of all events that triggered a state transition. A more detailed description of individual terminations is also available by selecting one in the popup window. In the parameter value history view (the foremost window in figure 1), parameters can be selected from a list – when selected, the values of the parameter will be plotted over time (the sequence of states selected in the behaviour graph).



```
1 -> 2:   from_equal_to_greater(dead3, born3)
2 -> 3:   to_interval_above(number_of1)
          to_interval_above(cover1)
```

Selected states: [1, 12

Selected path: [1, 2, 3, 9, 10, 11, 12]

Figure 1: State-transition graph for the Cerrado simulation. For the selected path between state 1 and 12, the transition history is opened, as well as the value history for parameter number_of(shrub), which shows an increase from zero to high.

Like a traditional x-y graph that plots a dependent variable against time, a graph format is used with the quantity space of the parameter value on the y-axis, and the state sequence on the x-axis. Point values are plotted on horizontal lines indicating specific value points, whereas interval values are plotted between two lines, corresponding to the values bordering the interval. Note that connecting these points to form a line-graph would suggest too much, because the exact slope of ascent/descent within intervals is unknown in our qualitative simulation framework (nevertheless, straight dotted lines are used in the QSIM approach – but with the comment that 'dots connecting symbols have no significance' – Kuipers, 1994, p. 23).

Representing mathematical and causal relations

To find out the causes for changes which are represented in, *e.g.*, the transition history, or parameter value history described earlier, VisiGarp offers a view on the causal and mathematical model - the parameter relations view. Figure 2 shows a screenshot of the causal model for state 1 in a simulation of the Brazilian Cerrado vegetation with three populations (grass, shrubs and trees), and fires (Salles & Bredeweg 1997). In the diagram, all parameters are shown, together with the dependencies between parameters, between parameter values, and parameter derivatives. Parameters belonging to the same entity are grouped together within the block representing that entity, creating a graph of nodes and subnodes, resembling a higraph (Harel, 1995), but with edges occurring only between nodes of the same type. This facilitates recognition of dependencies within subsystems, and dependencies crossing subsystem borders. Toggle-buttons are supplied alongside the diagram to show or hide specific types of information. This way, also the value, the quantity space and the derivative of parameters can be shown. If turned on, the quantity space of a parameter is displayed within the parameter node in vertical orientation, with the current value highlighted and an arrow beside it, indicating in- or decrease. Also, it is possible to show or hide the entities and attribute relations to clarify the system structure.

To show what kinds of decisions have been made in the design of this visualization, consider the following alternative representation, developed earlier by members of our group (Arnbak et al., 2000), in figure 3. When comparing figure 2 and 3, several differences can be noted:

Parameters inside entities vs. connected to entity names

An argument for the representation in figure 3 is that the layout of the parameters is not restricted to a rectangular entity block, as in figure 2, which can be exploited to reduce the number of crossing lines if there are many parameter relationships between different entities. However, displaying the entity name alongside each parameter node results in repetition which is avoided in the representation of figure 2. Furthermore, in figure 2, it is easier to see which parameters belong to the same entity, as they are grouped together within the entity block. This representation also facilitates determining at which points in a causal chain the connection is made between parameters of different entities, as these parameter relations literally cross entity boundaries.

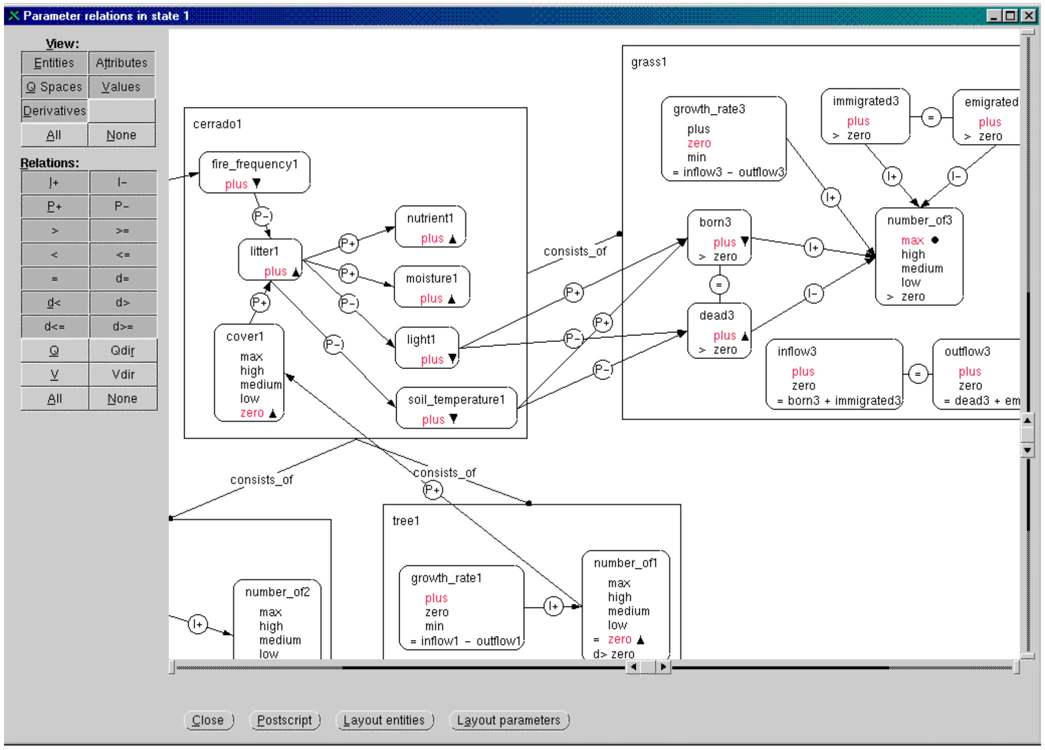


Figure 2: The parameter relations view for state 1 of the cerrado simulation, showing parameters within the entity they belong to, and relationships as labelled arrows. The I-relations denote influences, P-relations denote proportional relationships. Together, they specify the causal model in the domain. Also shown are the quantity spaces of the parameters, with the current value highlighted, and their derivative displayed alongside.

Mathematical relations represented inside parameters or as connections between parameters

Both figures use a labeled arrow between the two parameters involved for binary relationships like greater, and smaller than, but they differ with respect to their representation of formulae like outflow = dead + emigrated. In figure 3, the labeled arrow representation is expanded to incorporate such formulae, by splitting the ternary relationship into representations of the equal-sign part (outflow =), and the operator part (dead + emigrated), and connecting the two with an extra line. 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 this representation was judged as quite hard to read without experience, the design of figure 2 includes the original formula simply as text inside the node for the parameter it defines. It was judged more important to support a correct and natural reading of the formula than to have greater graphical uniformity and greater visibility of the existence of a mathematical relation connecting all parameters involved.

Representing quantity space and value of a parameter

In figure 2, the quantity space is shown within each parameter node, with the current value for this state highlighted in red. This way, it is made clear that a value belongs to a particular domain (i.e., quantity space), and that this quantity space is specific for the parameter in question. Because the display of such detailed information may distract attention from the parameters and their relations, it is important to note that they can be hidden from the display by depressing the toggle-button for quantity spaces, which makes the parameter nodes collapse to a format closer to the one in figure 3.

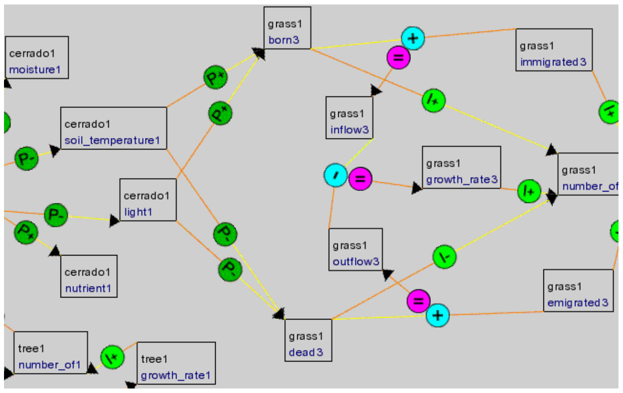
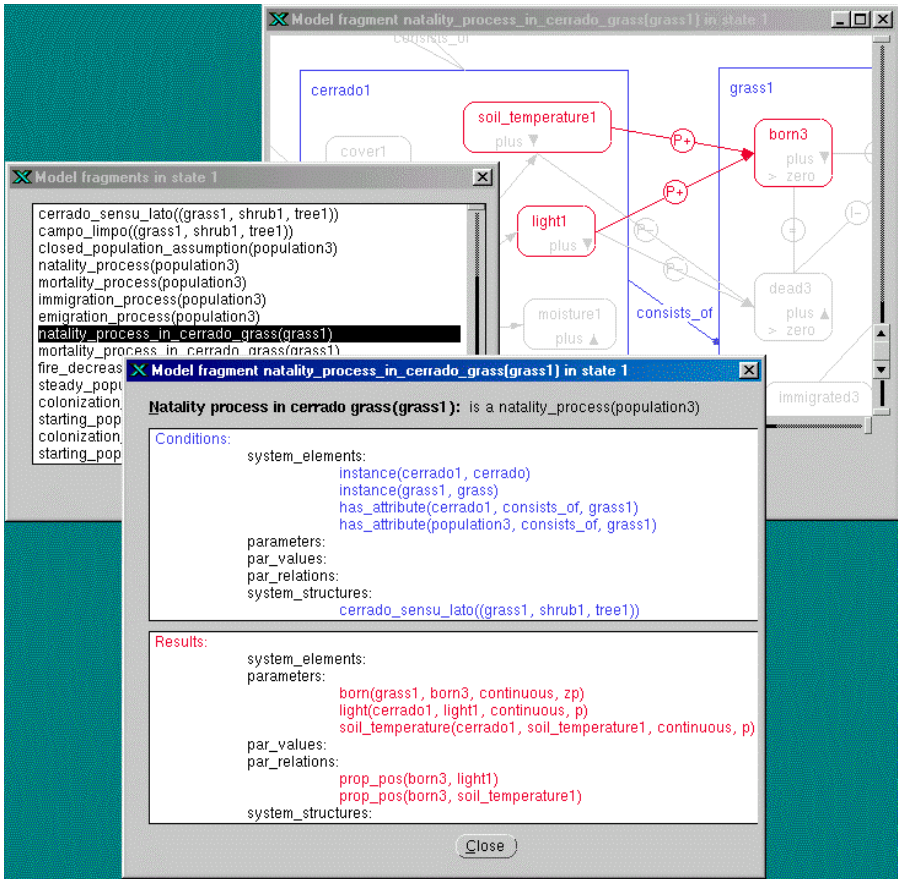


Figure 3. Fragment of a screenshot from GarpApplet: causal and mathematical relationships between parameters in the Brazilian cerrado vegetation domain. Alternative representation of the same information as displayed in figure 2.


Model fragment natality_process_in_cerrado_grass(grass1) in state 1
cerrado1
soil_temperature1
cover1
grass1
born3
light1
moisture1
consists_of
dead3
immigrated3
Model fragments in state 1
Close

```
cerrado_sensu_lato((grass1, shrub1, tree1))
campo_limpo((grass1, shrub1, tree1))
closed_population_assumption(population3)
natality_process(population3)
mortality_process(population3)
immigration_process(population3)
emigration_process(population3)
natality_process_in_cerrado_grass(grass1)
mortality_process_in_cerrado_grass(grass1)
fire_decreas
steady_popu
colonization
starting_pop
colonization
starting_pop
```

```
Model fragment natality_process_in_cerrado_grass(grass1) in state 1

Natality process in cerrado grass(grass1): is a natality_process(population3)

Conditions:
            system_elements:
                        instance(cerrado1, cerrado)
                        instance(grass1, grass)
                        has_attribute(cerrado1, consists_of, grass1)
                        has_attribute(population3, consists_of, grass1)
            parameters:
            par_values:
            par_relations:
            system_structures:
                        cerrado_sensu_lato((grass1, shrub1, tree1))

Results:
            system_elements:
            parameters:
                        born(grass1, born3, continuous, zp)
                        light(cerrado1, light1, continuous, p)
                        soil_temperature(cerrado1, soil_temperature1, continuous, p)
            par_values:
            par_relations:
                        prop_pos(born3, light1)
                        prop_pos(born3, soil_temperature1)
            system_structures:
```

Figure 4: The list of model fragments that are active in state 1. The contents of the selected model fragment *natality process* for the grass population is shown in detail, both in text and graphics format.

Use of colour

Figure 3 makes use of colour to encode several kinds of information.[1] As discussed above, the two parts of a labeled connection line/arrow are coloured differently to indicate the intended direction of reading; this is not necessary in figure 2, since the mathematical relations are represented differently there. More importantly, different types of relations are coloured differently in figure 3, e.g. shades of green for causal relationships, light blue for addition and subtraction, and purple for (in)equality. This helps the distinction of the different ontological groups of relationships, but the choices made are not the only ones possible. For example, to emphasize positive versus negative relationships, one might use one colour for positive proportionalities and influences, and another for negative proportionalities and influences. By contrast, the use of colour is kept to a minimum in figure 2, but more flexible and customized colouring and highlighting schemes may be included in future versions of VisiGarp.

[1] Colours are invisible in this black/white version; we hope some of the differences in greyscale can be seen.

Visualizing the contents of model fragments

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 4). 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 graphic format. The text format clearly distinguishes the conditions - which must be true for the model fragment to be applicable (shown at the top, in blue) - from the results - the knowledge introduced by the model fragment (shown at the bottom, in red). Also, the different knowledge categories are shown, with the actual content indented.

The grapical format is based on the causal model view (shown in figure 2), with colour used to highlight the contents of the specific model fragment (again, blue for conditions, and red for the results). 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 textual format, however, is more compact and gives a concise overview of what types of knowledge are introduced.

It is also possible to view how the different model fragments are structurally related to each other in the model library. Two views (not shown in this paper) are supplied for this purpose: the is-a hierarchy of model fragments, which shows the hierarchical subtype-relationships between model fragments, and the applies-to hierarchy, that shows which model fragments are conditional for which other model fragments. In both views, a model fragment can be selected for further inspection.

Evaluation of VisiGarp

VisiGarp has been evaluated by twenty-five third-year undergraduate students at our department (Social Science Informatics) at the University of Amsterdam. Participation in the study was mandatory, as a part of a course on evaluation of software. All but one of them had no experience with qualitative simulation models, nor with the domain of cerrado ecology, but most of them were reasonably computer-literate. The participants were given a short (30 min.) introduction to qualitative simulation and VisiGarp, explaining its research goals, its interface, and the domain model about the Brazilian cerrado vegetation.

Then, the participants had to work out ten small exercises on a paper handout sheet, using the VisiGarp interface, and a paper user guide. They spent about thirtyfive minutes on average to complete these tasks. Afterwards, they filled in two questionnaires, of which one was aimed at usability and potential usefulness; the other consisted of VisiGarp screenshots and test questions, intended to test whether the graphical representations could be understood after such a short period of experience with the system. The results indicated that people could use VisiGarp to run a simulation, investigate paths of transitions, find out the amount and direction of change in parameters, and determine causal paths leading to such changes. However, there were also points of criticism. In the current implementation, the automatic layout mechanism is not optimal, causing chunks of graphical elements to be scattered over the screen too widely, sometimes partly out of sight. Hence, diagram layout was rated 2.16 on average (SD = 1.25), on the answer scale of 1 (negative) to 5 (positive). Consequently, the participants had to spend a considerable amount of time scrolling and dragging to manipulate the layout of the visualizations. The participants considered it relatively easy to modify the layout to fit their needs, however (average score: 3.72, SD = 1.24). Not surprisingly, given the short time available and their unfamiliarity with the domain, 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. 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). A hypothesis to be tested is whether translation of the English terms (both domain-specific and generic QR concepts) into Dutch (most participants' native language) could help to overcome the two last points. The results from the knowledge-test questionnaire indicated that most easy questions were answered correctly, but the trickier questions were answered wrongly fairly often (in total, 75% of all answers was correct). In retrospect, some of the questions were probably not formulated clearly enough to rule out alternative interpretations, or did not specify clearly enough where to look for the answer. Overall, the results and comments indicated that VisiGarp needs more work on the layout and navigation mechanisms, but also that it 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.

Current status and further work

VisiGarp has been implemented in SWI-Prolog/XPCE (Wielemaker & Anjewierden, 1992). All figures in the VisiGarp screenshots were automatically generated, with some manual layout adjustments, using a large simulation model developed for GARP by Salles and Bredeweg (1997). This mechanism works for GARP models in any domain, e.g., a piston system with a heat-flow, a balance system with leaking fluid containers (Koning et al., 2000), and the ecology of Brazilian cerrado populations illustrated in this paper. Other work in our group concentrates on a JAVA implementation of some of these, and other visualization ideas (Arnbak et al., 2000). The results of the evaluation indicated that the layout of the diagrams is often suboptimal, especially when the models get larger. More intelligent layout algorithms, or alternative browsing techniques (e.g., automatic zooming or hyperbolic browsing) may alleviate parts of these problems. But even more leverage may be acquired by selecting the most interesting information in a simulation model. For instance, certain parameters sometimes deserve more attention than others, because they highlight the main difference between different states. Automating this selection process may benefit from research on aggregation and abstraction mechanisms (*e.g.,* Koning et al., 2000; Mallory et al., 1996).

A second goal of future work is to combine graphic representations with textual explanations, which can describe (e.g, see Pilkington & Grierson, 1996), contrast (*e.g.,* see Ferguson & Forbus, 1998), or add information to the figures. This involves thinking about explanatory dialogue (e.g., Moore, 1995), and educational settings in which the student is no longer only inspecting simulations, but also asked to form hypotheses, solve exercises and given feedback on his/her performance.

A third direction for further research is to integrate model inspection tools such as VisiGarp with model building tools, such as MoBuM (Bessa Machado 2000).

This allows students to build their own models, which can then be used to run simulations. Integration of the two systems will lead to interactive model building environments which empower students to articulate their thoughts, experiment with the results generated by their own model, and reflect on the outcomes.

Conclusions

This paper has described the design and implementation of VisiGarp, a tool for inspection of qualitative simulation models. Based on the output of the GARP simulation engine, which takes a library of model fragments and scenarios as input, VisiGarp generates diagrammatic representations of the knowledge and information involved. This includes views on structural aspects of the model, parameter values changing over time, and causal and mathematical relationships. Different visualization options have been illustrated, with a discussion of their particular advantages and disadvantages and the underlying motivations. Preliminary evaluation has shown that third-year undergraduate students can use the VisiGarp diagrams to complete exercises asking for simple qualitative predictions and causal explanations about domains unfamiliar to them. Suggestions for improvements are directed especially at enhancing the automatic layout mechanism, because students had to spend too much time on scrolling and manipulating the layout. Further work will also address automatic selection of the most interesting information to visualize, and integration of diagrams and textual explanation.

Acknowledgements

We would like to thank Peggy Tjaris for her contribution to the evaluation study, and to prof. Bob Wielinga, prof. Joost Breuker and the anonymous reviewers for their valuable comments on earlier drafts of this paper.

References

Arnbak, N., Dyk, T. van, Groen, R., Werf, R. van der. (2000), The GarpApplet. Internal report, Dept. of Social Science Informatics, University of Amsterdam, The Netherlands.

Bessa Machado, V. (2000). MoBuM: A Model Building Environment. Technical report, Dept. of Social Science Informatics, University of Amsterdam, The Netherlands.

Bredeweg, B. (1992). Expertise in qualitative prediction of behaviour. Ph.D. thesis, University of Amsterdam, The Netherlands.

Falkenhainer, B., Forbus, K. and Gentner, D. (1989). The Structure Mapping Engine: Algorithm and Examples. *Artificial Intelligence*, 41, pp. 1-63.

Ferguson, R. W., & Forbus, K. D. (1998). Telling juxtapositions: Using repetition and alignable difference in diagram understanding. In K. Holyoak, D. Gentner, & B. Kokinov (eds.), *Advances in Analogy Research*. Sofia: New Bulgarian University, pp. 109-117.

Forbus, K.D., & Whalley, P.B., & Everett, J.O., Ureel L., & Brokowski, M., & Baher, J., Sven E. Kuehne, S.E. (1999). CyclePad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*, 114, pp. 297–347

Harel, D. (1995). On Visual Formalisms. In J. Glasgow, N. H. Naranayan & B. Chandrasekaran (eds.), *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. AAAI/MIT Press, Cambridge, MA/London, pp. 235-271.

Hollan, J.D., Hutchins, E.L., & Weitzman, L. (1987). STEAMER: An interactive inspectable, simulation-based training systems. In G.Kearsley (ed.). *Artificial intelligence and instruction: applications and methods*. Addison-Wesley, Reading (Mass), pp. 113-134.

Joolingen, W.R., & Jong T., de (1991). Supporting hypothesis formation by learners exploring an interactive computer simulation. *Instructional Science*, 20, pp. 389-404.

Koning, K. de, Bredeweg, B., Breuker, J., and Wielinga, B. (2000), Model-based reasoning about learner behaviour. *Artificial Intelligence*, 117: pp. 173-229.

Kuipers, B. (1994). *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Cambridge, Massachusetts / London, England: MIT Press.

Larkin, J. H. & Simon, H. A. (1995). Why a diagram is (sometimes) worth ten thousand words. In J. Glasgow, N. H. Naranayan & B. Chandrasekaran (eds.), *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. AAAI/MIT Press, Cambridge, MA/London, pp. 69-109

Mallory, R. S., & Porter, B. W., & Kuipers, B. J. (1996). Comprehending complex behavior graphs through abstraction. In Iwasaki, Y., and Farquhar, A., eds., Proceedings of the Tenth International Workshop on Qualitative Reasoning, 137–146. Menlo Park, CA, USA: AAAI Press.

Moore, J.D. (1995). *Participating in Explanatory Dialogues. Interpreting and Responding to Questions in Context*. Cambridge, Massachusetts / London, England: MIT Press.

Pilkington, R. M., & Grierson, A. (1996). Generating explanations in a simulation-based learning environment. *International Journal of Human-Computer Studies*, 45, pp. 527-551.

Salles, P., & Bredeweg, B. (1997). Building Qualitative Models in Ecology. Proceedings of the International workshop on Qualitative Reasoning, QR'97. Istituto di Analisi Numerica C.N.R. Pavia, Italy, L. Ironi (ed.). pp. 155-164.

Weld, D., & Kleer, J. de (eds.) (1990). *Readings in qualitative reasoning about physical systems*. Palo Alto, CA: Morgan Kaufmann Publishers.

White, B.Y., & Frederiksen, J.R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, 42: pp. 99–157

Wielemaker, J. & Anjewierden, A. (1992). Programming in PCE/Prolog. Dept. of Social Science Informatics, University of Amsterdam, The Netherlands.