Project no.	**004074**
Project acronym:	**NATURNET-REDIME**
Project title:	**New Education and Decision Support Model for Active Behaviour in Sustainable Development Based on Innovative Web Services and Qualitative Reasoning**
Instrument:	**SPECIFIC TARGETED RESEARCH PROJECT**
Thematic Priority:	**SUSTDEV-2004-3.VIII.2.e**

D4.2.3 Analysis of Frequently Asked Questions and Improvements to the Garp3 Workbench[1]

Due date of deliverable: **31/08/2007**
Actual submission date: **05/09/2007**

Start date of project: **1st March 2005** Duration: **30 months**

Organisation name of lead contractor for this deliverable:
UvA

Revision: Final

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

[1] Authors: Anders Bouwer, Jochem Liem, Floris Linnebank and Bert Bredeweg

Abstract

This document describes the analysis of frequently asked questions from domain experts using the Garp3 workbench for building qualitative models of ecological case studies in the context of the NaturNet-Redime project, and presents improvements made to the workbench to further increase its usability and support users in the model-building process. A list of questions has been compiled, together with answers from modelling experts, and organized based on the topics they address. This list is made publicly available, so that it can be used as a resource for learning about qualitative modelling and the Garp3 workbench. The enhanced Garp3 workbench includes improvements to the reasoning engine and different aspects of the graphical user interface.

Document history

Version	Status	Date	Author
0.1	Draft document structure	04/07/2007	A. Bouwer
0.2	Added section 2.4 and section 8	14/08/2007	J. Liem
0.3	Added section 3 and 6.2	29/08/2007	A. Bouwer
0.4	Added section 5, 6, 7	30/08/2007	B. Bredeweg, A. Bouwer
0.5	Added section 4	31/08/2007	F. Linnebank
0.6	Improved Layout, added figures and references	31/08/2007	A. Bouwer
0.7	Final editing	03/09/2007	J. Liem
0.8	Improved references order	04/09/2007	J. Liem
0.9	Final editing	05/09/2007	F. Linnebank
0.9	Final editing	05/09/2007	F. Linnebank
1.0	Final editing	05/09/2007	B. Bredeweg

Contents

1. Introduction

This document describes our efforts to further support the process of building qualitative models. The results presented in this report are all based on close interaction with active users of the Garp3 software. In section 2, we describe the means we have used to manually support these users in modelling their case studies within the NaturNet-Redime project. This support was archived as much as possible, and used to work out improvements for the Garp3 workbench and the corresponding support. As a result, we have developed and integrated into the Garp3 software and website, a collection of solutions to complement and gradually replace the manual support provided by our modelling experts. The following improvements have been realised. Section 3 presents a systematically organized list of Frequently Asked Questions (FAQ) together with their answers. Section 4 describes technical improvements to the Garp3 reasoning engine. Section 5 presents support for multiple languages and the improved tooltip functionality. Section 6 describes improvements to the Simulate environment. Section 7 discusses improvements to the Sketch environment. Section 8 presents other improvements to the Garp3 software. Section 9 concludes this document.

2. Means of Support

2.1. Skype and VNC

In order to provide support on the qualitative modelling of case models within the NaturNet-Redime project, software was used to enable remote real-time communication between domain experts and Garp3 modelling experts. Skype was used to allow free telephone-like communication over the internet. VNC was used to share a common desktop on which the Garp3 software could be run. Both Skype and VNC were used concurrently to complement each other and provide a richer form of communication.

2.1.1. Internet Telephony using Skype

As telephone communication to other countries is very expensive, Skype is a good alternative to communicate. The Skype program can be used to make free calls over the internet to anyone else who also has Skype. It is available for free and works on most operating systems (Windows/Linux/MacOSX). Skype offers the possibility of making conference calls with up to 5 people. Two additional features are traditional chatting and file transfers.
To support our users with installation and use of the software, we put online a small set of guidelines at the QRM portal: http://hcs.science.uva.nl/QRM/community/skype.html.

2.1.2. VNC

Virtual Network Computing (VNC) is an open protocol which is used to remotely control a computer (the server) by using a simple program (the viewer/client). It allows users to see the desktop of a remote machine (and optionally control it with their local mouse and keyboard), as if the user is in front of that computer. A person who wants to share his desktop, starts a VNC Server, and specifies a password giving complete interactivity rights, and another for view-only rights. Other participants can connect to the VNC Server (using the password) via the internet to view (and interact with) the desktop on that computer. VNC implementations (servers and clients) exist for most platforms (most importantly Windows, Linux and MaxOSX). This allows viewers to connect to servers running a completely different operating system.
To support our users with the installation and use of VNC, we put a small set of guidelines online at the QRM portal: http://hcs.science.uva.nl/QRM/community/vnc.html.

2.2. FlashMeeting

From April 2007 onwards, the decision was made to switch from using Skype and VNC to using FlashMeeting instead. FlashMeeting does not allow desktop sharing as VNC does, but it does include audio-visual communication (provided that headsets and cameras are available for all participants), and furthermore, it allows interaction sessions to be archived for recollection in the future. There have been 11 FlashMeetings between UvA and HiFi, and 11 FlashMeetings between UvA and BOKU, between April and July 2007. Due to a temporary failure of the server on which the FlashMeetings are stored, two of the most recent meetings could not be recovered. For a selection of 10 of the most interesting meetings, relevant excerpts have been transcribed and summarized. This has been one of the main sources for extracting questions from users about Garp3 and the modelling process.

2.3. QRM Mailing List

In August 2005, we created the Qualitative Reasoning and Modelling (QRM) mailing list, to support knowledge dispersion within the community of Garp3 users. The topics of the mailing list revolve around conceptual and technical issues related to building qualitative simulation models in the Garp3 framework, as well as announcements for updates of the Garp3 software. Users can send questions or comments about the software, qualitative modelling and reasoning in general, or a specific issue in a particular model. The UvA team of modelling experts assumes a leading role in answering questions posed on the QRM list as much as possible. Besides the participants in WP6 of the Naturnet-Redime project, also people from outside are able to subscribe to this mailing list. Currently, there are 19 active subscribers to the QRM mailing list.

The questions and answers are archived on the QRM portal (at http://hcs.science.uva.nl/QRM/community/) so that subscribers can browse and search the archive to look for solutions to problems they may have. The QRM archive currently contains 129 messages.

2.4. Bug Report System and Bug Fixes

As part of the ongoing effort to support modellers in their work, and to further structure the development process of Garp3, the Mantis bug report system[2] was added to the QRM Portal in October 2006 (see Figure 1.1). This allows users of Garp3 to file bugs, and both users and developers to easily keep track of issues. Until now over 80 bugs were filed of which 70 are resolved. There are about 10 bugs still open which can either not be reproduced, are planned as future work, or are currently not important enough to spend resources on.

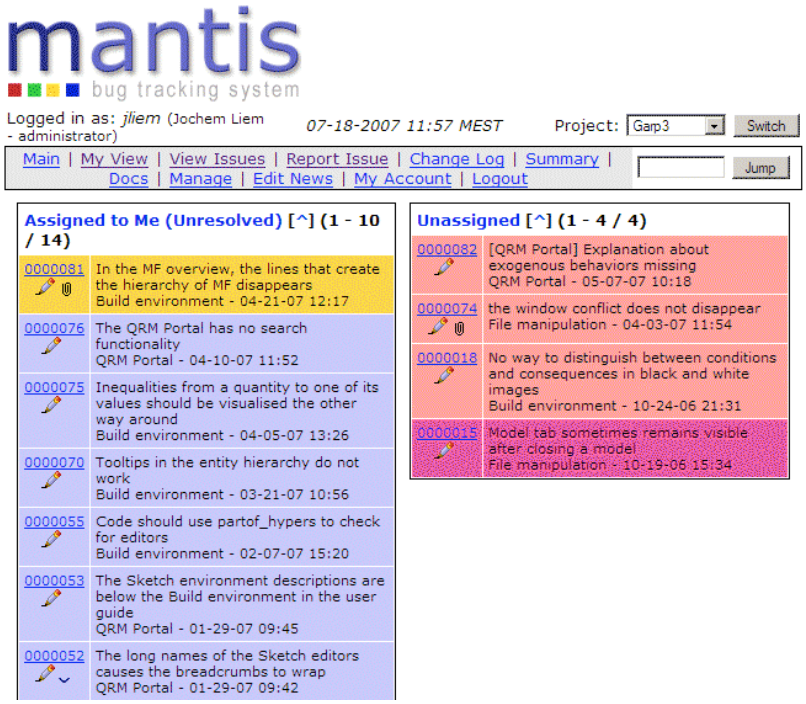

Figure 1.1: The Mantis Bug report system on the QRM Portal

[2] http://www.mantisbugtracker.com/

3. Frequently Asked Questions

During the model-building process, novice modellers are likely to encounter difficulties which can be remedied easily by expert modellers. Because some problems occur relatively frequently, it is worthwhile to record these problems, together with their potential solutions, in the form of a list of Frequently Asked Questions (FAQ). To establish such a list, we have analyzed the model building process of several users involved in case studies within the NaturNet-Redime project. The questions originate from interactions between users (domain experts) and GARP3 modelling experts that have been recorded and analyzed, as well as from email correspondence (see section 2.2 and 2.3). Where necessary, the questions have been rephrased to enhance clarity and to be more widely applicable. Furthermore, the answers to these revised questions have been assembled by Garp3 modelling experts. The result is a systematically organized list of questions and answers which are expected to be useful for future users working on similar issues. For this reason, the complete (and regularly updated) FAQ is also made available online at the Qualitative Reasoning and Modelling website at http://hcs.science.uva.nl/QRM/help/faq.

3.1. The List of Frequently Asked Questions

The list of Frequently Asked Questions (FAQ) is categorized by topic. The following topics have been identified: software; causal dependencies; correspondences; inequalities and values; model fragments and scenarios; system structure: entities, agents, configurations, etc.; simulation; modelling process; save and print. The complete list of questions is shown below, organized by topic.

Software

1. What is Garp3?
2. What happened to Garp, VisiGarp and Homer?
3. How and where should I save my models?
4. How do I prevent model corruption?
5. [Linux] The correct Garp3 windows do not become active in KDE. Why?

Causal Dependencies

1. What makes a causal model correct?
2. Should I use a proportionality or an influence?
3. How do I model a feedback loop?

Correspondences

1. When should I use a correspondence?
2. Should a correspondence be directed or undirected?

Inequalities & Values

1. What kind of quantity space should I use?
2. Why is it not possible to create inequalities between zero and another value?
3. When should I introduce an extra variable for a calculation?
4. Can you specify (in)equality statements between derivatives?
5. What is the purpose of model fragments with only conditional inequalities and values?
6. Should I add equality statements between zero point values of quantity spaces of different quantities?
7. If I define the equality relations A=B and B=C, should I also define A=C?

8. What is the difference between specifying that two values are equal and specifying that two quantities are equal?
9. I want my quantity to be bigger than zero, can I just use an inequality?
10. Why is setting values of quantities and derivatives bad practice?

Model Fragments & Scenarios

1. Can I put everything into one model fragment?
2. What is the difference between a condition and a consequence?
3. How does Garp3 reason with model fragments with conditions only?
4. Why is my model fragment considered "Incomplete"?
5. What is the difference between static, process and agent fragments?
6. When should I use an assumption?
7. Where should I put an inequality relation: in the scenario, or in a particular MF?
8. This process MF (or agent MF) should become active/inactive in some situations. How do I model that?
9. How can I enforce the machine to try all possibilities of relative ordering between two variables?
10. Why are there multiple elements (e.g., entities, or quantities) in this model fragment with the same name?

System Structure: Entities, Agents, Configurations, etc.

1. To which entity should I attach a quantity?
2. When should I use an agent as a modelling concept?

Simulation

1. What is the best way to create a good qualitative model and simulation?
2. I don't have any states. Why?
3. Why doesn't my model fragment fire?
4. My simulation contains only one state - why aren't there more?
5. The simulation does not generate the states that I would expect. Why not?
6. How to fix a model which does not generate all expected states?
7. The simulation contains too many states. What can I do about this?
8. How do I determine whether the behaviours in the simulation are correct?
9. Why does it take so long for my simulation to run?
10. How do I run a simulation step by step?
11. Should I run the full simulation or run it step by step?
12. How do I select a state with a particular number?
13. How can there be multiple identical states in the simulation?
14. Why are there two inequalities X > Y and Y < X in this state?
15. How to ensure that a proper end state is reached?
16. How can I inspect what happens in the transitions between states?
17. Why does a quantity have no derivative?
18. Why does a quantity have no magnitude?
19. I added a value in the scenario, but this value is lost in the simulation. Why?
20. I have two opposing influences on a quantity. When one is removed I get no successor state. Why?
21. An influence originating from a quantity with a non-zero value is not causing the target quantity to increase (or decrease). Why?
22. I get an "Out of local stack" error. What is going on?

Modelling Process

1. When should I start running a simulation?
2. How to model human actions?
3. Is it possible to model cyclic system behaviour?
4. How do I create a model that shows cyclical system behaviour?
5. How do I come up with good names for entities, quantities, model fragments, etc.?

Save and Print

1. How do I create images of models for publications?
2. How do I create images of the diagrams of models or simulation results for presentations?
3. How do I create an image of the Garp3 interface?

The answers to these questions are shown in the following subsections, each covering a particular topic.

3.1.1. Software

1. What is Garp3?

Garp3 is a workbench for building, simulating, and inspecting qualitative models.

2. What happened to Garp, VisiGarp and Homer?

All those tools have been integrated into Garp3

3. How and where should I save my models?

A good practice is to save your models in a separate directory outside your Garp3 directory. It is a bad idea to save your models in the Garp3 directory, since when you upgrade Garp3 you might accidentally delete your old Garp3 directory including your models. Furthermore it is good practice to save different versions of your model during development. This makes it easier to go back if a certain new representation proves not to work, or certain changes were counterproductive. It also makes the model corruption less disastrous. A common practice is to add the suffix vs0x to model files to indicate version numbers. A further suffix a, or b can be used to distinguish different competing improvements to the same model.

4. How do I prevent model corruption?

One good practice is to save models often and as different files, as when a model becomes corrupt it is possible to go back to a previous version. When collaborating with other people make sure that you both run the same version of Garp3, as opening and saving a model from a recent version in an older version of the software may cause model corruption.

5. [Linux] The correct Garp3 windows do not become active in KDE. Why?

By default KDE has window focus stealing prevention on. This has to be set to *None* so applications can make the correct windows active. To deactivate window focus stealing prevention go to 'Control Center', 'Desktop', 'Window Behaviour'. In the 'Advanced' tab

set 'Focus stealing prevention level' to 'None'. The correct Garp3 windows should now become active.

3.1.2. Causal Dependencies

1. What makes a causal model correct?

Causal chains often start with an influence (I), which may be followed by proportionalities (P) that propagate the effect. Chains of proportionalities (P) following each other occur quite often, but chains of multiple influences (I) following each other are likely to be incorrect. If you have both I and P dependencies affecting the same quantity, it is generally advised to remove one of the two.

2. Should I use a proportionality or an influence?

Influences should be used when the following relation has to be modelled: "If the magnitude of the source quantity has a non-zero value, the target quantity will change." When the relation "If the source quantity changes, the target quantity will change too." has to be modelled, proportionalities should be used. Influences use the magnitude of a quantity to derive the derivative of the target quantity, while proportionalities use the derivative of a quantity to derive the derivative of the target quantity.

3. How do I model a feedback loop?

This can be done for example by using an influence (I+ or I-) from X to Y for the direct effect, and a proportionality (P+ or P-) from Y to X for the feedback effect. A loop consisting of only proportionalities is incorrect (especially when it involves an uneven number of P- dependencies, it will not work). It can be useful to try running the model first without the feedback effect, and add it later.

3.1.3. Correspondences

1. When should I use a correspondence?

A correspondence can be used to ensure that values always occur together. Be aware that this may not be the case when there are multiple opposing influences on the quantities involved. In that case, a correspondence may be too strict.

2. Should a correspondence be directed or undirected?

Is there a causal connection with a particular direction? In that case, the correspondence should probably be directed (with the same direction as the causality). Or do the corresponding values always occur together? In that case, the correspondence should be undirected (= bidirectional).

3.1.4. Inequalities & Values

1. What kind of quantity space should I use?

The quantity space should contain just the right amount of distinctive values that are necessary to model the behaviour of that particular quantity in a qualitatively meaningful way. Think about whether zero should be included as a point value, and which other point values might be necessary (e.g., as thresholds for the specification of conditions for model fragments). Is it possible for the quantity to assume a negative value? Since the amount of possible states greatly depends on the size of quantity spaces, it often

helps to reduce the number of states by limiting the number of values in a quantity space.

2. Why is it not possible to create inequalities between zero and another value?

Garp3 prohibits the creation of inequalities between zero and other values, since it would only add superfluous knowledge. Every zero is by definition equal to each other zero. The only reason to create an inequality would be to indicate that a certain point value is greater (or smaller) than zero. A better way to indicate that a point value is greater (or smaller) than zero is to add the value zero in the quantity space.

3. When should I introduce an extra variable for a calculation?

Introducing an extra quantity for a calculation can be a good idea when this result has an intuitive meaning, and will be used at several places itself (e.g., in conditions of model fragments, or other calculations). Sometimes, the new quantity can be considered to be an aggregate of multiple other quantities (e.g., population growth as an aggregate of more basic processes birth, death, immigration and emigration). However, be aware that introduction of new quantities can lead to an increase in complexity.

4. Can you specify (in)equality statements between derivatives?

Yes you can, by selecting the derivative symbols of two quantities and adding an (in)equality statement.

5. What is the purpose of model fragments with only conditional inequalities and values?

If the magnitude or derivative of a quantity cannot be determined (via causal dependencies, plus or minus relations, equality relations or correspondences), normally the quantity would have unset values for the magnitude or derivative. However, the engine has a reasoning-assumption mechanism for conditional values and inequalities that cannot be proven to be true (cannot be derived, but are possibly true). For example, if a quantity has an unset magnitude, and a model fragment has a conditional magnitude value for that quantity, the engine tries to assume that the condition holds (i.e. the quantity has that particular value). As always, if an inconsistency can be derived (possibly due to the assumed value), the state is deleted. The same mechanism applies for inequalities. If a conditional inequality cannot be derived, it is assumed. Examples of model fragments with only conditional inequalities or values can be found in the full envisionment version (version 2) of the communicating vessels model.

6. Should I add equality statements between zero point values of quantity spaces of different quantities?

No, by definition zeros are universally equal throughout the model. This means that even zeros belonging to different quantity spaces can be considered as having an equality relation between them. Adding these equalities is superfluous.

7. If I define the equality relations A=B and B=C, should I also define A=C?

No, the qualitative simulator can infer this by itself, adding the statement would be superfluous. This inference is called transitive inequality reasoning.

8. What is the difference between specifying that two values are equal and specifying that two quantities are equal?

Specifying that two values are equal means that those values are quantitatively the same. For example two cups could have the same maximum height (e.g. 5cm). This does not mean that the quantities have the same value at the same time. Specifying that two quantities are equal means that they will have the same value at the same time during simulation.

9. I want my quantity to be bigger than zero, can I just use an inequality?

It is strange to specify that a quantity cannot have a specific value, as there should be a reason this value was defined in the quantity space. A good reason to constrain the possible values of a quantity is to minimize a simulation. In this case the assumption is made that the quantity has smaller than, equal to or greater than some specific value. The best way to model this is to create a new model fragment in which the entity/agent with the corresponding quantity is included along with an assumption, all as conditions. Furthermore, the inequality should be added as a consequence. Once the assumption is included in a scenario, the states in which the value is zero will be removed from the state graph. By removing the assumption from the scenario the states will reappear.

10. Why is setting values of quantities and derivatives bad practice?

Especially the values of derivatives should be inferred by the qualitative simulator using proportionalities and influences. By setting these you model that something should have a specific value in a specific situation, but not why it should have that value. When proportionalities and influences are used, the cause of change within a system is modelled, and not the result of some process. For this reason setting values of quantities should be reduced to a minimum, and setting values of derivatives should be avoided at all times.

3.1.5. Model Fragments & Scenarios

1. Can I put everything into one model fragment?

In principle, there is no boundary to the amount of information in a model fragment (MF). However, it is good modelling practice to use separate MFs for distinct concepts or ideas. This facilitates systematic model building, exploring alternative possibilities, debugging the model, and possible reuse of model fragments.

2. What is the difference between a condition and a consequence?

Model ingredients can be incorporated into model fragments as conditions or consequences. The conditions indicate the model ingredients a scenario has to contain in order to become active. The consequences in a model fragment represent the model ingredients that have to be added to a simulated scenario if the model fragment applies. A model fragment can be read as: 'When these conditions are met, these consequences have to be true.'

3. How does Garp3 reason with model fragments with conditions only?

If a condition is not inconsistent with the remaining information derived then Garp3 will assume the condition to be true.

4. Why is my model fragment considered "Incomplete"?

In an incomplete model fragment not every entity and agent is structurally related. There should always be a path of configurations leading from each entity or agent to each other entity or agent.

5. What is the difference between static, process and agent fragments?

Static fragments are used to model stable characteristics of a system, such as the structure of the system. They may also contain proportionalities, but they cannot contain influences or agents. Process fragments are used to model processes that induce change within the system. Agent fragments are used to model external effects on the system. Influences can be used in process and agent fragments. Agents can only be used in agent fragments.

6. When should I use an assumption?

Assumptions (e.g., inequalities between certain quantities) can be used to constrain the simulation in some way. It is common practice to define an assumption with a name that corresponds to the MF, and use this as a condition for the MF. To control external quantities, it may be nicer to use agents in model fragments.

7. Where should I put an inequality relation: in the scenario, or in a particular MF?

This depends on whether you know the inequality is true from the start (in this case, you can put it in the scenario), or is used as a trigger for a particular MF (in this case, put it in the conditions of the MF), or is true when a MF fires (in this case, put it in the consequences of the MF). Sometimes, however, it is not necessary to specify the inequality at all, if it can be derived automatically from the quantity values. If it is possible to let the machine determine it automatically, this is preferred, so that the number of inequality specifications in MFs and scenarios can be kept to a minimum.

8. This process MF (or agent MF) should become active/inactive in some situations. How do I model that?

It is common practice to model a process (or agent) with a process quantity (e.g., flow) whose value is calculated by the difference between two other quantities, and influence(s) originating from this process quantity. This way, the influence(s) become inactive when the difference is zero, i.e., when the balance is restored. Alternatively, it is possible to use specific activation conditions for process (or agent) MFs to fire, but then care should be taken that when the process becomes inactive, this does not lead to discontinuous changes.

9. How can I enforce the machine to try all possibilities of relative ordering between two variables?

This can be done by creating 3 subtype model fragments that say 'assume X smaller than Y', 'assume X equal to Y', and 'assume X greater than Y', with the appropriate (in)equality statements as conditions.

10. Why are there multiple elements (e.g., entities, or quantities) in this model fragment with the same name?

When an MF is reused (e.g., by creating a child MF, or by incorporating an MF as a condition), all of its elements are put into the new model fragment. When multiple MFs are reused in one new MF, this may lead to multiple elements (entities, quantities, etc.) with the same name. If two occurrences are meant to refer to the same thing, they should be connected by an identity link in the new MF.

3.1.6. System Structure: Entities, Agents, Configurations, etc.

1. To which entity should I attach a quantity?

When the model contains the right set of entities, it is often straightforward to determine which entity a quantity should belong to. When in doubt between different entities, first check whether these entities are really modelling different concepts, and whether any entities are missing. Think about the other quantities of each candidate entity: does the quantity fit well into one of these lists? In the case of a quantity related to a process involving multiple entities (e.g. a flow from one container E1 to another container E2), the quantity may be modelled as belonging to entity E1, entity E2, or to an entity modelling something related to the process itself (e.g., a pipe entity E3).

2. When should I use an agent as a modelling concept?

Agents are used to model processes that affect the system of interest, but are external to it. For example, if the system of interest is a river, a polluting factory might be modelled as an agent that adds pollution to the river, while not being part of the river itself. Agents are labels that are applied within scenarios and model fragments. If a particular label appears in a scenario, model fragments that contain that agent label as a condition are applied. If the label does not appear in the scenario, these model fragments are ignored.

3.1.7. Simulation

1. What is the best way to create a good qualitative model and simulation?

It is a good idea to first think about the system you want to model and your models goals, before actually creating the model. A structured approach to go through this process is by following the Framework for conceptual QR description of case studies. At the implementation stage it is crucial not to try and implement the entire model at once. Instead it is better to create and test individual model fragments one-by-one as much as possible. Also focus on model fragments that model single processes. Describe (explicate) the behaviour you expect each model fragment will generate. For each of these processes create at least one scenario to test the model fragments. Improve the single model fragments until the behaviour is correct and move on to the next model fragment. Then try to combine different model fragments in the simulation and improve them to generate the correct behaviour. Work towards the end result in a stepwise manner, ensuring the correctness of each individual step. Do not immediately start creating dozens of model fragments, as it will make the debugging of models extremely difficult.

2. I don't have any states. Why?

When running a simulation produces no states, there is an inconsistency between either the facts specified in the scenario and the consequences of a model fragment, or between different consequences in different active model fragments. Notice that, if no model fragments fire, a single state would be present describing the scenario. A possible way to debug the situation in which no states are produced, is by deactivating all model fragments. The simulator should now generate a single state (namely the scenario itself). By activating model fragments one at a time and running the simulation, the contradicting model fragment can be found and corrected.

3. Why doesn't my model fragment fire?

A model fragment only fires if a state fulfils its conditions. If one or more of the conditions is not fulfilled the model fragment does not fire. Check the following to see if this is the case:

- Structural conditions. Entities of the correct type should exist, the configurations should exist and be in the correct direction, and attributes should exist and have the correct value.
- Behavioural conditions. Check whether quantities are present with the correct quantity space, as well as values and inequalities. Note that when inequalities or values are unknown, they can be assumed by the engine allowing the model fragment to fire.
- Conditional model fragments and parents of a model fragment have to fire in order for a model fragment to fire itself. If one of these model fragments does not fire, check why these do not fire first.

In some cases, the consequences may be the cause of the problem, rather than the conditions. This happens when multiple model fragments fire but have consequences that cause conflicts, thereby preventing the state from being generated.

4. My simulation contains only one state - why aren't there more?

If only one state is generated, this means that the scenario has only one possible interpretation (which is often a good thing), and that no successor state can be generated. If successor states were expected, check whether the model fragment(s) that should induce changes to the system have fired as expected, resulting in quantities increasing or decreasing.

5. The simulation does not generate the states that I would expect. Why not?

Often, this is due to specifying too many constraints in the scenario, or in model fragments, e.g., value statements, inequality statements, or correspondences. The more knowledge is specified, the higher the chances of conflicts arising.

6. How to fix a model which does not generate all expected states?

Pick a state that should lead to one (or more) successor state(s), and try to find out why this is not generated. Check whether the terminations you expect are generated, by turning on the Simulation Display option 'show termination nodes', and opening the transitions view for the originating state. If the termination is there, but doesn't lead to a successor state, the tracer can provide information about why the termination was unsuccessful (especially, the options 'Show search for possible terminations', 'Show

ordering and removal of possible terminations', and 'Show search for successor states' should be turned on). If the terminations you expected are not being generated, check whether the quantity derivatives are as expected, and check whether the model fragments fire as expected. When you understand the problem, try to fix the model in such a way that the successor state(s) is/are generated, and proceed with the next state.

7. The simulation contains too many states. What can I do about this?

There may be several reasons for this:

- When quantities can change independently from each other, this may lead to exponential growth of the state-transition graph because of all the possible combinations of quantities that can increase, remain steady, or decrease, either synchronously or asynchronously. The number of successor states is 2^q-1, where q is the number of unrelated (i.e. non-corresponding) quantities changing in an interval (this calculation does not take quantities changing in different directions into account). Check if there are bogus states and try to fix the model in such a way that it prevents them from being generated. For example, quantities may be fluctuating when they shouldn't. Check if there are quantities which should behave synchronously, and add correspondences or other means to enforce that behaviour.
- When there are model fragments that have too many interpretations, or model fragments that (erroneously) exclude each other, this may also cause too many states to be generated. Check if model fragments fire as expected, and adjust them if necessary.
- When there are multiple initial states which all lead to many successor states, it can help to simplify or constrain the scenario so that only one initial state is generated.
- When far too many states are generated, it can help the investigation process to (temporarily) add some constraints to the model to limit the number of states in a clear-cut way. For example, add correspondences to make a set of quantities behave synchronously (as described above), or add an inequality statement that only allows certain states but not others.

8. How do I determine whether the behaviours in the simulation are correct?

If the number of states is approximately what you would expect, look at paths of behaviour and see if the behaviour makes sense. Look at the values of (all) states to see if the right combinations of values occur. Look at individual states to check at a more detailed level. Do the right model fragments fire? Do the end states represent proper end states, in which a maximum/minimum is reached, or the system behaviour has stabilized? If cyclic (repetitive) behaviour was expected, check whether this is present in the simulation.

9. Why does it take so long for my simulation to run?

The simulation probably contains many states and/or transitions which have to be calculated. If this is not what you expected, try running the simulation again step by step and check if you can notice anything going wrong by looking into individual states and their transitions. Another option is to limit the depth of reasoning in Garp3, by setting a

value (e.g., 5) for Maximum Inequality Reasoning Depth, in the Simulation preferences (Advanced).

10. How do I run a simulation step by step?

At the bottom of the Simulate main screen, there are three buttons 'Terminate selected states', 'Order selected states', and 'Find successors for selected states'. In the display menu, turn on 'Show termination nodes'. Select a particular state (or multiple states) and press one of the three buttons to run the simulation step by step.

11. Should I run the full simulation or run it step by step?

If you have a small model and expect a small simulation, running a full simulation will produce the results faster, but running it step by step can help you detect problems early, especially when the simulation gets large.

12. How do I select a state with a particular number?

On the bottom of the Simulate screen, in the selected states field, enter the state number between square brackets, for example '[45]'.

13. How can there be multiple identical states in the simulation?

This is in fact impossible - there must be a difference between states, in either the values or the equations. If you can't see a difference in the quantity values between the two states, there must be a difference in the equations. You can check this by opening the equation history for the states in question.

14. Why are there two inequalities X > Y and Y < X in this state?

It could be that the inequality is used in different forms in different model fragments, or it could be that the machine adds it in a particular form, in case there is a transition involving an inequality change. In general, only one of the two is necessary, because Garp3 knows that they mean the same thing.

15. How to ensure that a proper end state is reached?

There may be several reasons why an expected end state is not reached. In many cases, it has to do with quantities that should stabilize having reached their lowest (e.g., zero) or highest value. You can try using the option to 'remove inactive quantities' to remove a quantity which is prohibiting the end state from occurring now. Use this option with care, however. Sometimes, processes should become inactive in an end state, while opposing forces still remain. If this is not handled adequately, it might imply a discontinuous change, thus preventing a successor state from being generated.

16. How can I inspect what happens in the transitions between states?

When a state or path is selected, pressing the button 'Transition history' will present the list of relevant transitions. When selecting one of the transitions in this list and pressing the button 'Transition details', or double-clicking on a transition directly, more details about the selected transition are shown.

17. Why does a quantity have no derivative?

When a quantity has no derivative there are two possible issues. Firstly, the quantity is being influenced via a proportionality relation by a quantity that has no derivative. In that case, making sure that quantity has a derivative solves the issue. Secondly, the quantity is being influenced via an influence by a quantity that has no magnitude. Giving the influencing quantity a magnitude will solve the issue. Another possible solution is to give the quantity an exogenous behaviour that gives the quantity a derivative.

18. Why does a quantity have no magnitude?

When a quantity has no magnitude there are three possible issues. Firstly, it could be that there was no value set of the quantity in the simulated scenario. Setting the value in the simulated scenario solves this issue. Secondly, it could be that the quantity magnitude was known in the previous state, but not in the current state, and the derivative of the quantity was unknown in the previous state. Then, making sure the derivative of the quantity is set in the previous state will allow Garp3 to determine the magnitude of the quantity in the current state. Thirdly, the magnitude of a quantity can be calculated by using the plus or minus operator (adding or subtracting two quantities, and indicating that a third quantity is equal to the result). It could be that the model fragment that adds the operator does not fire, or that one of the quantities that is being added (or subtracted) has an unknown magnitude. Making sure that the model fragment fires, or the magnitude of the quantities is set, will solve the issue.

19. I added a value in the scenario, but this value is lost in the simulation. Why?

Values specified in a scenario are only initial values. This means that they indicate the value of a quantity in the initial states, but can change during simulation. Therefore during simulation the value of a derivative can become unknown if it is not influenced in some way, and the magnitude of the quantity can change to other values. If a value of a magnitude or derivative has to have a specific value during simulation a value assignment has to be set to that specific value in a firing model fragment (e.g. in combination with a conditional assumption to allow simulations in which the value does change). Another option is to define exogenous quantity behaviour in the scenario.

20. I have two opposing influences on a quantity. When one is removed I get no successor state. Why?

Consider the following. If a quantity is changing into the direction of the dominant influencing quantity in the context of two opposing influencing quantities, and the stronger influence is removed, the resulting overall influence suddenly changes into the other direction. Due to continuity constraints this is not possible; a derivative can only change direction if it first becomes stable. One way to solve this issue is by making the stronger influence become smaller when it approaches zero. Another representation is to introduce a new quantity and subtract the two influencing quantities using the minus operator and use an equality relation to specify that the result is equal to the new quantity. This new quantity then causes a single influence.

21. An influence originating from a quantity with a non-zero value is not causing the target quantity to increase (or decrease). Why?

The issue is probably that the originating quantity does not have a zero value in its quantity space. As a result the engine cannot determine whether the influence should cause a negative or a positive influence. Therefore it is unable to determine the derivative value of the target quantity.

22. I get an "Out of local stack" error. What is going on?

It could be possible that the number of possible transitions is too large to be calculated without running out of memory. This does not really matter, as the resulting state graph would have become too large to understand anyway. A possible solution is to try to reduce the number of states. (See: "I have far too much states. How can I reduce them?")

3.1.8. Modelling Process

1. When should I start running a simulation?

It is a good idea to run a simulation as soon as you have a small scenario and a few model fragments ready, to test if they work as expected. Try to predict what will happen first, and check your predictions. If something is wrong, determine whether it is the model, or your prediction. It is generally wise to fix any error before adding additional elements to the model. This way, you will develop a feel for how the simulation engine works, and how subsequent changes to the model affect your simulation results.

2. How to model human actions?

Think about whether they should be agents or processes. Conceptually it may be nicer to model human activities as agents, although it may also depend on whether the human behaviour is affected by something else. In that case, the human behaviour may be considered a process rather than an external influence. From a technical point of view, it does not matter. To decide about the type of causal dependencies to use, consider the following. If you have only one rate (I), that will generally drive the simulation in one direction, without being able to control it. Therefore, to model a certain action happening, it may be handier to use a P, than an I. But if you have opposing actions or processes, having multiple I's may be an even better solution.

3. Is it possible to model cyclic system behaviour?

Yes, it is possible to model cyclic behaviour. Whenever a particular state-transition leads to a state which already exists, a cycle is created, which represents a possible behaviour that repeats itself.

4. How do I create a model that shows cyclical system behaviour?

The proper way to do this involves modelling explicit causes (i.e., processes) that cause quantities to increase and then decrease again (or vice versa), in such a way that a path can lead back to an earlier state. A shortcut way to model this is to open a quantity definition in a scenario and specify that it should be an exogenous quantity, with sinusoidal behaviour.

5. How do I come up with good names for entities, quantities, model fragments, etc.?

If you are in doubt over the name of a term that you use in your model, this may be an indication that something is wrong with it. Will other people understand what you mean by it? Is it in the right place? Check if the concept you want to model is not already present somewhere else in the model. Does the concept fit in the perspective of the other elements in the model? Discuss the concept with somebody else, or study the literature for inspiration. Make sure you use the Remarks-field to enter your thoughts about the concept, including your doubts. However, note that from a technical point of view, the actual name does not matter for Garp3, as long as it is unique within its context.

Tip: using labels with numbers to precede the name of MFs (e.g., 'Mf15...' and scenarios (e.g., 'Sc3...') makes it easy to refer to them in discussions, and gives you control over how they are ordered.

3.1.9. Save and Print

1. How do I create images of models for publications?

Garp3 has functionality to export diagrams as Encapsulated Post Script (.eps) files. EPS is a so-called vector graphics format, which means it allows infinite zooming without degrading the quality of the image. It is the perfect format for publications. EPS files can be imported as an image in Word. Word creates a preview of the image which usually looks quite bad on the computer screen. However, when the Word-file is converted to PDF the actual EPS graphic is used, and the resulting document has the best quality. PrimoPDF is a free PDF converter (printer driver) which can be used to convert Word files to PDF (Windows only).

2. How do I create images of the diagrams of models or simulation results for presentations?

The EPS images of diagrams that can be exported from Garp3 are not suitable to use in PowerPoint presentations, since EPS support in PowerPoint is poor. It will not show the EPS images in presentations. However, creating screenshots of Garp3 is a bad alternative, since the screen-shots do not have a high enough resolution to create nice looking images. Instead, the EPS should be converted into another format, as explained below.

Probably the best file format for images in PowerPoint (if the PowerPoint file is not converted to pdf) is Portable Network Graphics (.png). Adobe Photoshop and Adobe Illustrator have built-in eps/png support, but are commercial solutions. The GIMP is a free image processing solution which can be used to convert eps to png. It requires the GTK+ toolkit to work, which is also available from the GIMP website. There is a slight drawback, as the eps support is not built-in. Ghostscript has to be installed and the GS_PROG environment variable has to be set to the full path of gs.exe, as described in the GIMP FAQ. The guide to setting environment variables in Windows XP might be a helpful resource to set the GS_PROG variable.

3. How do I create an image of the Garp3 interface?

Garp3 does not support creating images of the Garp3 interface. The only way to create images of Garp3 is by creating a screenshot of the application using functionality native to your operating system.

4. Improvements to the Garp3 Reasoning Engine

The core Garp3 Reasoning Engine was updated substantially during the course of the project. We have made changes and added reasoning capabilities both in response to bug reports and user feedback as well as based on previously known deeper conceptual issues that needed handling.

Although the issues discussed originate mainly from working with the NNR case studies, simple 'standard' models (taken from our model repository, available at http://hcs.science.uva.nl/QRM/models) are used in this section to illustrate and explain detailed issues.

4.1. Simulation Preferences

Garp3 has the possibility of controlling certain aspects of the reasoning engine. In the latest version of Garp3 these simulation preferences were updated. We have added some preferences to accommodate new capabilities and removed some preferences whose function had become obsolete. The new preference window is shown in Figure 4.1.

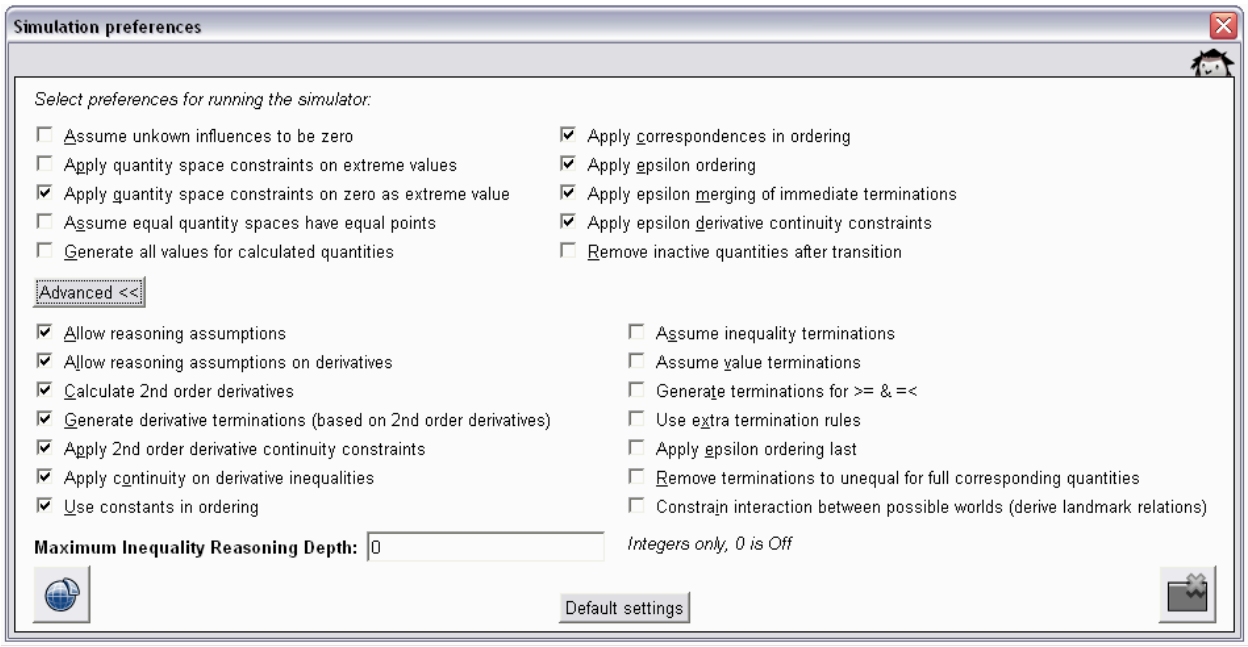

Figure 4.1: Simulation preferences

4.1.1. Basic preferences

- Assume unknown influences to be zero

  Was renamed from: 'Apply closed world assumption in influence resolution'.
  This function sets all unknown, influencing quantities to zero right before evaluating the effects of influences and proportionalities. This is done only for influences that are not influenced themselves.

In some cases this zero value[3] or derivative may conflict with later results of the influence resolution procedure, but only if inequalities connect it to the effected quantities. For example: Quantity A is proportional to quantity B and C. Furthermore dC is plus and dB is unknown, so dB will be assumed to be zero. Now if an equality is present dA = dB the influence resolution will calculate dA to be plus while the equality suggests it should be zero, a contradiction.

- Apply quantity space constraints on extreme values

  These constraints stop a quantity from increasing or decreasing in the highest or lowest point of the quantity space respectively. NB This setting does not apply to intervals or the point zero.

- Apply quantity space constraints on zero as extreme value

  These constraints stop a quantity in zero from increasing or decreasing if this is the highest or lowest point of the quantity space respectively.

- Assume equal quantity spaces have equal points

  Normally quantity spaces for different quantities are unconnected. With this preference switched on quantity spaces based on identical definitions will have their identical points connected with an equality relation automatically. A more elaborate explanation of this new preference can be found in section 4.3.

- Generate all values for calculated quantities

  This preference will make the reasoning engine assign all possible values to each unknown quantity that is defined by an addition or subtraction. All other values in this inequality must be known. A more elaborate explanation of this new preference can be found in section 4.5.

- Apply correspondences in ordering

  Correspondences (for values) are a good information source for ordering terminations in the transition process. For example if one value changes, its corresponding value should change at the same time. In general correspondences are valid throughout a simulation, so this preference should only be switched off in models where correspondences change from state.

- Apply epsilon ordering

  Changes from a point and from equality will happen instantly and therefore before all other changes which take at least some small epsilon amount of time. In case of an instant change, non-immediate changes are removed. Turning of this preference (and other epsilon preferences) will partly remove the point-interval distinction present in Garp3.

[3] In all of section 4 the word value can be read as the word magnitude unless otherwise indicated.

- Apply epsilon merging of immediate terminations

  Changes from a point and from equality will happen instantly and therefore they should happen all at once. This function groups all immediate terminations as much as possible. A full grouping is not always possible because mutual exclusive terminations may be present.

- Apply epsilon derivative continuity constraints

  When this preference is switched on quantities will not stabilize over an immediate transition. The rationale is that derivatives may not change to zero in an immediate transition because this is a non-immediate event. A more elaborate explanation of this new preference can be found in section 4.11.

- Remove inactive quantities

  This function removes quantities in a state that are not mentioned in the set of active model fragments. Typically this happens after a state transition where a quantity associated with a process disappears because the process has stopped.

4.1.2. Advanced preferences

- Allow reasoning assumptions

  Reasoning assumptions are made in the model fragment selection procedure when conditions are assumable. With this preference switched off, model fragments will only fire when all conditions are positively known. A more elaborate explanation of this new preference can be found in section 4.12.

- Allow reasoning assumptions on derivatives

  With this preference switched off, no reasoning assumptions will be made in the model fragment selection procedure about derivative values and derivative inequalities. A more elaborate explanation of this new preference can be found in section 4.12.

- Calculate 2^{nd} order derivatives

  When this preference is switched on 2^{nd} order derivatives are determined for quantities with multiple influences. This is done using the causal model. Results are used to generate derivative terminations and derivative continuity constraints. A more elaborate explanation of this new preference can be found in section 4.4.

- Generate derivative terminations (based on 2^{nd} order derivatives)

  When this preference is switched on 2^{nd} order derivatives are used to trigger derivative terminations. Note that a quantity derivative may still change normally through the continuity regime, but not if a derivative termination for this quantity is present in the set of terminations. A more elaborate explanation of this new preference can be found in section 4.4.

- Apply 2nd order derivative continuity constraints

  When this preference is switched on 2nd order derivatives are used to generate derivative continuity constraints. These constraints are applied in the successor state right before the influence resolution procedure. Each constraint is only applied if the causal model has not changed for the affected quantity. A more elaborate explanation of this new preference can be found in section 4.4.

- Apply continuity on derivative inequalities

  The continuity regime in Garp3 releases derivatives over state transitions under constraints to allow continuous change. With this preference switched on this regime is also applied to derivative inequalities.

- Use constants in ordering

  When this preference is switched on constant relations are used to order terminations in the transition process. Relations can be labelled constant in the scenario and relations involving addition/subtraction will be assumed constant.

- Assume inequality terminations

  An equality will normally terminate only when it is known that their derivatives are unequal. With this preference switched on, Garp3 will implicitly assume unequal derivatives when possible and fire terminations accordingly.

- Assume value terminations

  A value will normally terminate only when it is known that its derivative is known to be positive or negative. With this preference switched on, Garp3 will implicitly assume derivatives when possible and fire terminations accordingly.

- Generate terminations for >= and =< inequalities

  Normally these ‘weak’ relations will not terminate, with this preference switched on, they will. The >= inequality can terminate into > if the quantity on the left side is rising relative to quantity on the right side. Or it can terminate into = and < in the opposite case.

- Use extra termination rules

  Legacy Garp had a rule based transition procedure. With this function domain specific termination rules in this format can be used. For example: Rules can be constructed that remove entities (e.g. Water may be evaporating and once its amount reaches zero the entity is removed).

- Apply epsilon ordering last

  Use this preference to use the old ordering procedure that applies the epsilon ordering concept last. A more elaborate explanation of this new preference can be found in section 4.10.

- Remove terminations to unequal for full corresponding quantities

  Turn this function on to remove terminations from equal to unequal from the transitions for quantities that have a full correspondence. The rationale behind this is that these quantities may also share equal quantity spaces (preference: Assume equal quantity spaces have equal points) and therefore they will have to be equal on each point of their quantity space. Now an unnecessarily complex simulation is generated if the quantities can become unequal in the intervals only to become equal again in the following points.

- Constrain interaction between possible worlds (derive landmark relations)

  Renamed from: 'Derive landmark relations'. In the course of a simulation new inequality relations between landmark values may become derivable. With this function these relations are added to the state description to prevent simulation branches describing different possible worlds from interacting.

- Maximum Inequality Reasoning Depth

  In inequality reasoning two parent relations are combined to derive a new relation. This new relation can in turn become a parent of a new relation. This preference controls the amount of parent relations a derived relation can have. In models with a lot of additional calculus this can be used to speed up the engine during model development. In such models 6 is often a fast setting but completeness is not guaranteed in this case. A setting of 10 can still provide some improvement while adding more chance of finding all possible derivations. To be safe however this preference should be turned off by setting it to 0. A more elaborate explanation of this new preference can be found in section 4.8.

4.1.3. Removed preferences

The following preferences were removed because they are not used or needed in the current modelling practice.

- Application of analyse zero equality
- Allow assumptions on derivatives in reclassifying MFs
- Specify and Match instead of Subsumption
- Use complete rule based transition procedure

Their previous default values dictate the current reasoning engine behaviour.

4.1.4. Conceptually related preferences

Two groups of preferences are associated with a common theoretical concept. These are the preferences concerning epsilon ordering and 2nd order derivatives. Preferences in these groups have been placed together to invite the user to consider them all at once. Furthermore a message such as the one in Figure 4.2 will be displayed if a user simultaneously selects *and* deselects preferences in a particular group. This informs the user that these preferences are related to the same theoretical concept and should therefore be turned on or off simultaneously in most cases.

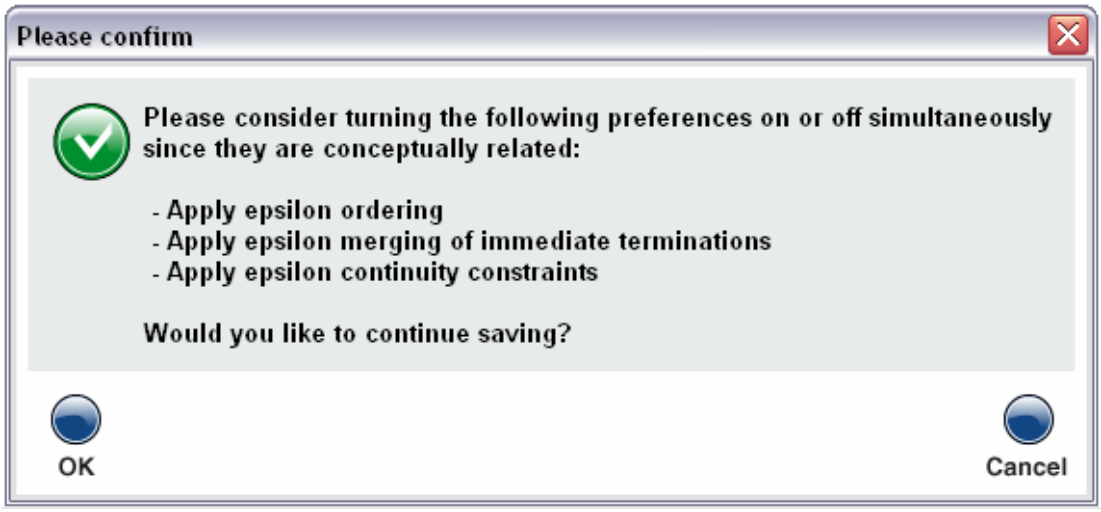

Figure 4.2: Conceptual relation warning

4.2. Quantity value bound by extreme points of quantity space

Whenever a quantity is introduced in the simulation, certain associated dependencies are added to the model. These always include inequalities describing the ordering of quantity space points. And depending on simulation preferences, some conditional dependencies concerning the derivative are also added. For example, if zero is the lowest point of the quantity space then the quantity cannot be decreasing in that point. Or: If max is the highest point then the quantity cannot be increasing in that point. Etc.

In addition, it turned out that dependencies stating that the quantity value is greater or equal to its minimum and smaller or equal to its maximum are also necessary. Consider the U-tube pictured in Figure 4.3.

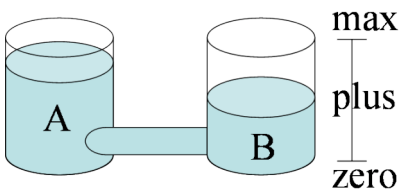

Figure 4.3: Communicating Vessels

Here we see that A and B are both plus, A is bigger then B and both maximum heights are equal. Now consider a state where the following conditions hold:

- A is max
- B > A
- max(A) = max(B)

Of course this state is impossible because it can be derived that B is bigger than its maximum. However, this invalid state *was* generated because the quantity space constraints on extreme values were not active. If the constraint B =< max(B) is applied this problem does not occur, because this relation contradicts the relation B > max(B) which follows from the above conditions.

In earlier versions of Garp3 these last constraints were present but they were coupled with the quantity space constraints for derivatives which are active only under the

simulation preference: Apply quantity space constraints on extreme values. In the latest version of Garp3 this coupling was undone so quantity values are always bound within the extremes of their quantity spaces.

4.3. Default equalities for quantity space points

Normally the points of different quantities are unrelated even if quantities use similar quantity space definitions with similar points (Note that zero is an exception here, it is the same point for every quantity space using it). For example in a U-tube model, there are three possible worlds as can be seen in Figure 4.4.

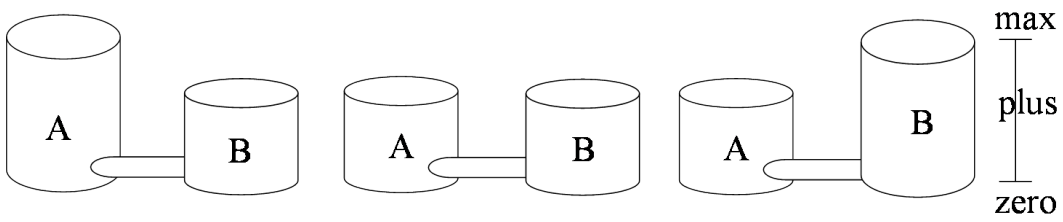

Figure 4.4: Possible U-tube worlds

Of course it is possible to specify the modelled world by adding an equality like: max(A) = max(B) or max(A) > max(B). In the latest version of Garp3 we have added a feature that does this automatically. It is controlled by the simulation preference: Assume equal quantity spaces have equal points. When this feature is active all points with the same quantity space definition are treated as being equal by default. This removes the need for modellers to explicitly model this equality and it therefore adds brevity and clarity to the model and to its resulting simulation. Also it is often behaviour that novice modellers will expect and therefore the preference can be used to smooth out the Garp3 learning curve.

4.4. Maintain inequality consistency: 2nd order derivatives

Multiple influences often create an ambiguous effect on a quantity. In this case three states are generated, each corresponding with a certain derivative value for this quantity. An example dependency view and value history is given in Figure 4.5.

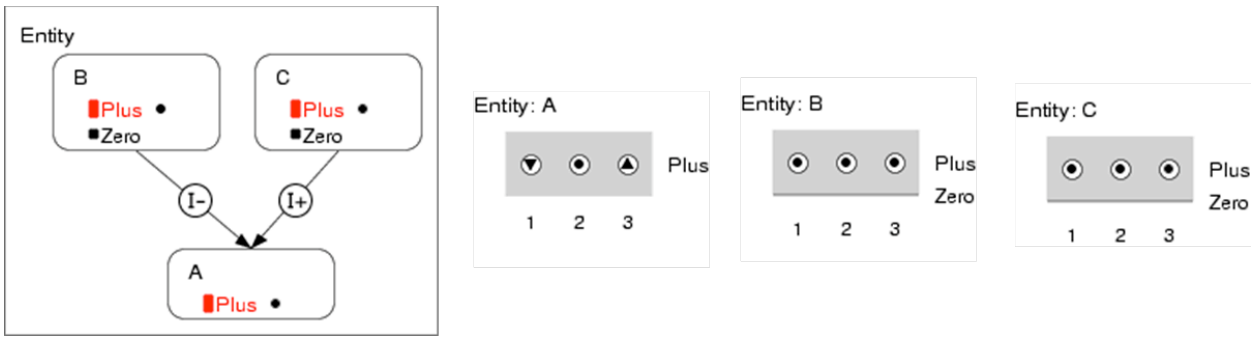

Figure 4.5: Ambiguous dependencies and values

Each state has a different derivative for the ambiguously influenced quantity and represents a certain relation between the magnitudes of the influences. In our example in state 1 the inequality B > C holds, in the second state B = C holds and in state 3 the inequality B < C holds. Now in this case it is also known that these magnitudes are stable, thus the relation between them should not change over a transition. However in

previous versions of Garp3 successor states would generate ambiguity again thus producing an invalid state graph.

A larger example of such a situation is given in Figure 4.6. The dependency view and value history for this simulation are given in Figures 4.7 and 4.8.

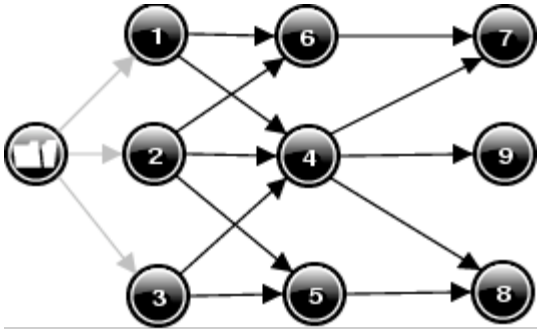

Figure 4.6: Inequality consistency not maintained; State Graph

In this model an extra exogenous quantity D is added that represents the active/changing rest of the model. It supplies terminations that the derivative of A uses to hitchhike to a different branch of the simulation. In the state graph, the diagonal transitions represent invalid events where the underlying inequality is incorrectly changing.

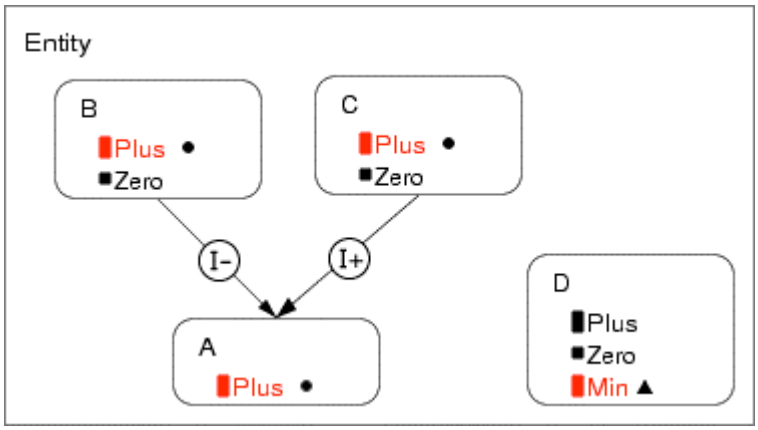

Figure 4.7: Inequality consistency not maintained; Dependencies

As can be seen in the dependency view D is unrelated to the main quantities and is simply going up from Min, to Zero, to Plus using the exogenous increasing behaviour pattern. Note that the 'hitchhiking' of A's derivative could also happen if B or C would be changing in some way where the underlying inequality would still have to remain the same.

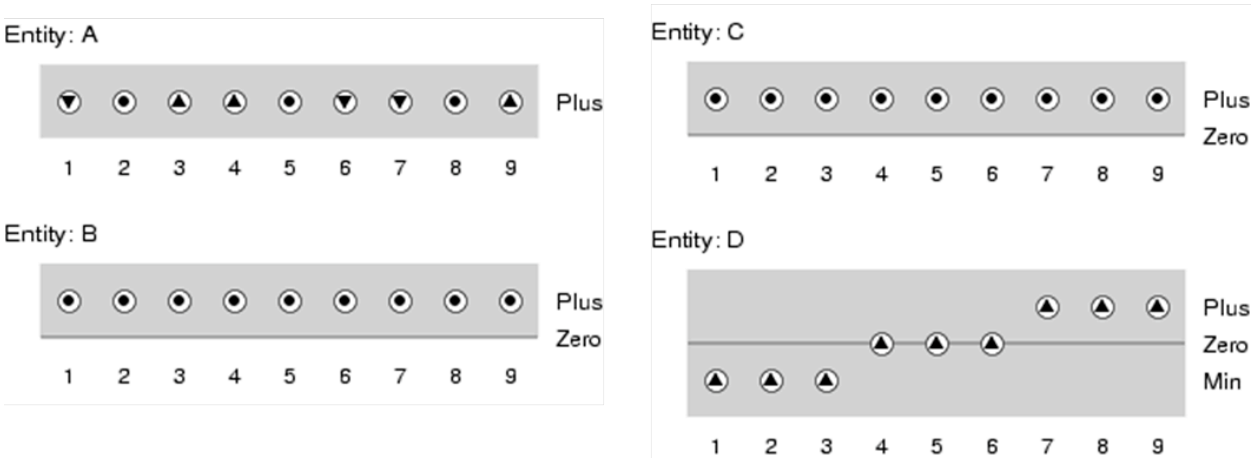

Figure 4.8: Inequality consistency not maintained; Values

The correct state graph in this example would be the one pictured in Figure 4.9.

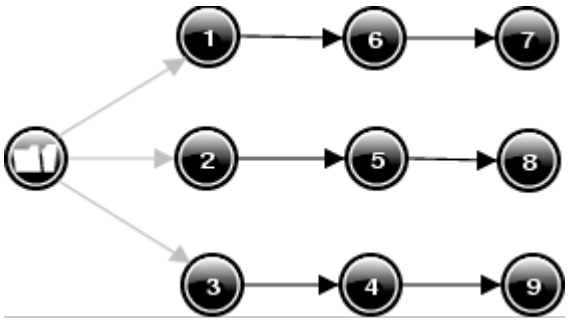

Figure 4.9: Maintained inequality consistency; State Graph

In general it can be said that ambiguous influences always have an underlying balance equation with the sum of all positive influences on one side and the sum of all negative influences on the other side. This equation determines their combined effect and it should only change if its derivatives indicate or allow this change. The generic version is as follows:

$$\sum I^{+} \ \{<, =, >\} \ \sum I^{-} \qquad \text{changes according to:} \quad \sum \partial I^{+} \ \{<, =, >\} \ \sum \partial I^{-}$$

As mentioned, in previous versions of Garp3 there was no mechanism to ensure this.

4.4.1. Maintain inequality consistency: Considered approaches

A number of possible solutions to this problem have been investigated before committing to a specific one.

1. The most obvious solution would be to generate the balance equation between influences in each case of ambiguity and add it to the simulation. This equation should then be allowed to terminate (change) to allow valid changes of the ambiguous derivative.

The major problem with this solution is that balance equations can involve more than 2 quantities (using additions) and such equations do not terminate in Garp3. They are even considered to be constant constraints describing inherent qualities of the modelled system and are used in this role in the ordering procedure.

Furthermore, to terminate balance equation, it would be necessary to determine the corresponding derivative relation. Since this equation can also involve additions it is very likely to be ambiguous itself. In this case we would not want our resulting effect to be restricted, but in practice it would be because an equality in Garp3 does not terminate if its derivative relation is unknown. So an exception procedure would be needed to terminate balance equations even with unknown derivative relations.

2. Generate all binary relations between influences. This generates more states, but terminations follow easily for binary relations. Three sets of combinations could be distinguished describing each possible balance equation.

This 'brute force' approach fails because of two reasons. Firstly there is the problem of combinatorial explosion. The number of possible combinations of binary relations and thus states for a number of *N* quantities and three possible relations (>, = and <) is equal to:

$$3^{\left(\sum_{1}^{n-1}(n)\right)}$$

This function grows very large very quickly. Even for only 4 influencing quantities, it is an unfeasible approach:

N	*States*
1	*1*
2	*3*
3	*27*
4	*729*
5	*59049*

Secondly a conjunction of binary inequalities is not expressive enough to represent an inequality with addition. For example the relation A + B < C cannot be captured using binary inequalities: The conjunction: C > A & C > B & A = B, is equally consistent with the above relation as it is consistent with the relation A + B = C or even A + B > C.

3. Generate two abstract quantities; SigmaPlus and SigmaMin, and use a binary relation between these quantities to describe the balance equation.

This approach would solve the problem of terminating non-binary inequalities which is present in the first approach. However the problem determining the derivative relation described there would remain. Furthermore the user would be confronted with an extra abstract quantity which makes the simulation more complex and less insightful.

4. Design case specific filters to rule out invalid derivative changes

An important feature of Garp3 is that it produces explanations of behaviour through its reasoning process. Therefore the filter solution was not investigated deeply since it is clearly not compatible with this explicit mechanism approach [1]. Furthermore this approach would likely not be generic enough to capture all possible cases.

5. Calculate the 2nd order derivative of quantities to determine which derivative changes are correct.

The first question for this approach is how to calculate the 2nd order derivatives. A clear point in any case would be that it is not be a good idea to branch states with an ambiguous derivative. A design choice in this solution is whether or not the 2nd order derivative would be incorporated in the user level representation and displayed accordingly in the simulator. A second question would be if terminations for derivatives should be generated if their 2nd order derivatives indicate a change is due.

4.4.2. Maintain inequality consistency: Solution

In the latest version of Garp3 we have chosen to implement the solution using 2nd order derivatives. To calculate 2nd order derivatives a procedure similar to the normal influence reasoning process is used. Figure 4.10 shows a typical causal model, starting with an influence, continuing with a proportionality and 'ending' with proportional feedback.

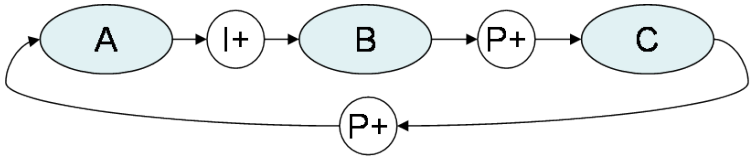

Figure 4.10: A typical causal model

The calculation of the effects of this causal model is shown in Figure 4.11. In this example 'v' indicates a value, 'd' indicates a derivative and 'dd' indicates a 2nd order derivative.

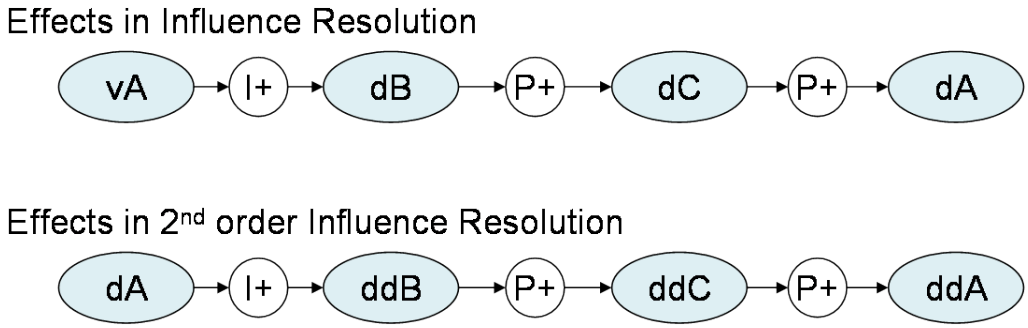

Figure 4.11: Determining causal effects.

As can be seen 2nd order derivatives can only be calculated if at least one influence is present in the causal chain and if all of these influencing quantities have a determined derivative. Therefore the procedure is carried out right after the normal influence resolution procedure. Hereby its effects are allowed to propagate on. As mentioned earlier, in the case of ambiguity no branching is done and the 2nd order derivative remains unknown since branching would result in numerous states that are hard to distinguish from a user perspective. Sometimes however a 'weak' result: >= or =< can be derived. This resulting weak 2nd order derivative is recorded because a weak 2nd order derivative can still supply strong constraints as will be shown.

To deal with unknown initial values a mechanism is built in that assumes driving variables to be zero if they are not influenced themselves. This mechanism is equivalent to the mechanism used in the normal influence resolution procedure. It is controlled by the same preference: Assume unknown influences to be zero, which is not active by

default. An idea that might be explored in future work is that, to reduce branching, 2nd order derivatives are also assumed zero when they are ambiguous.

A minor point that needs to be mentioned is that 2nd order derivatives are only used for quantities affected by multiple influences. The reason for this is that singularly influenced quantities have no need for any of the 2nd order derivative functionality:

- Constraints are not necessary since the derivative is directly dependent of the single influence. Thus if the influence does not change, the derivative won't change either.
- Derivative terminations are not necessary since the derivative can always change directly with the changing single influence. There is no hidden underlying inequality that can be changing without any value termination happening.

2nd Order derivatives of singularly influenced quantities *are* still calculated though because they may be part of a causal chain needed to calculate a 2nd order derivative of a quantity with multiple influences.

In the current implementation of Garp3 2nd order derivatives can be inspected in the tracer. They are stored with the state and used in transitions to determine extra constraints on derivatives besides those imposed by the continuity regime. Table 4.1 shows the constraints that are active in the successor state given a derivative and a 2nd order derivative.

2nd order Derivative \ Derivative	d(X) > 0	d(X) = 0	d(X) < 0
dd(X) > 0	d(X) > 0	d(X) > 0	d(X) =< 0
dd(X) >= 0	d(X) > 0	d(X) >= 0	d(X) =< 0
dd(X) = 0	d(X) > 0	d(X) = 0	d(X) < 0
dd(X) =< 0	d(X) >= 0	d(X) =< 0	d(X) < 0
dd(X) < 0	d(X) >= 0	d(X) < 0	d(X) < 0

Table 4.1: 2nd order derivative continuity constraints

As can be seen weak 2nd order derivatives can supply strong constraints. When for example a derivative is plus (first column) and it has a weak 2nd order derivative greater or equal to zero (second row) still a strong constraint is derived: The derivative must remain plus.

An important issue is the change of the causal model in a model. If this changes over a transition, constraints should not be applied. We implemented a procedure in Garp3 that checks the constraints right before the new causal model is evaluated (influence resolution). Each constraint is applied only if all influences on the quantity in question remain the same. Theoretically this change check would not be necessary if new influences would always start at zero and if removed influences would always go to zero

before disappearing. In these cases a derivative should still comply with the constraints posed by its predecessor's 2nd order derivative. And in the new state the new 2nd order derivative is determined using the new causal model. An example: Although the reason a car is moving changes, it still can never move discontinuously. Modelling practice showed however that this theoretical position does not hold. On some levels of abstraction one might for example need to model a transition where 2nd order derivative constraints of the previous causal model are not met. This happens for example when modelling a car crashing into a concrete wall without taking deformation into account.

The second important functionality that 2nd order derivatives bring is derivative terminations. A derivative can remain in place, or it can change. 2nd Order derivatives supply the required information. Derivative terminations are needed in cases where multiple influences act on a quantity. The underlying balance equation is changing, but no value terminations fire. In this case the derivative of the quantity should change but there is no termination for this. An example model is shown in Figure 4.12.

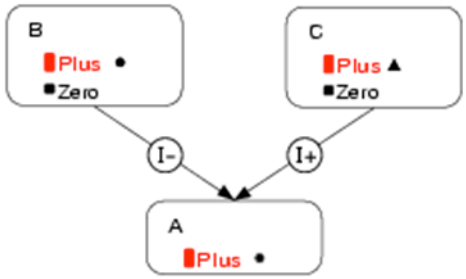

Figure 4.12: Dependencies with implicit changing inequality.

In this example the equality B = C must hold, because dA turns out to be zero. And since the dB is stable and the dC is plus, this equality should be changing and the derivative of A should become plus. In the latest version of Garp3 a derivative termination will fire in this case.

The table of constraints (Table 4.2) shows where derivative terminations are needed. The following terminations can be generated:

1. derivative_stable_to_up(Quantity)
2. derivative_stable_to_down(Quantity)
3. derivative_up_to_stable(Quantity)
4. derivative_down_to_stable(Quantity)

Their conditions and consequences are given in Table 4.2.

Derivative / 2nd order Derivative	d(X) > 0	d(X) = 0	d(X) < 0
dd(X) > 0		1: d(X) > 0	4: d(X) = 0
dd(X) >= 0		*	*
dd(X) = 0			
dd(X) =< 0	*	*	
dd(X) < 0	3: d(X) = 0	2: d(X) < 0	

Table 4.2: Derivative terminations

Note that in the conditions marked with a * a change is also possible. We decided however not to let these cases trigger a termination analogous to the case of value terminations that are not triggered if derivatives are not definitively known. Note that the two possible terminations in the middle column would be epsilon-immediate and therefore could take precedence over other terminations.

The ordering procedure in Garp3 determines which combinations of terminations are valid and which terminations take precedence over others. Some changes in this procedure were made to incorporate the derivative terminations.

The correspondence ordering procedure was updated to work on these terminations. Removing terminations when a correspondence seems to remain active is not done here like it is done with value terminations. The reason for this is that for values, terminations are the only agent of change, but for derivatives this is not the case. They may also be allowed to change by the continuity regime thereby inactivating the correspondence that suggests removing a derivative termination. On the other hand, valid and invalid combinations are in fact determined using correspondences, because in these cases a derivative termination is present on the activating side of the correspondence, thus ensuring that a termination will be the agent of change.

Furthermore a special continuity procedure has been added after determining the cross product of possible terminations and applying epsilon merging. This procedure adds constraints to derivatives that have a derivative termination: If a derivative termination is present in some sets, then that derivative is fixed in place in all other sets. Without these constraints, because of the normal continuity regime, a set without the derivative termination might lead to the same state as the same set with the derivative termination.

One drawback of having derivative terminations is that currently the visibility of 2nd order derivatives in simulate environment is low. They can only be inspected using the tracer. This not a conceptual problem however and 2nd order derivatives should be fully visible in the interface of future versions of Garp3.

As a result of the 2nd order derivative functionality Garp3 produces more sound simulations. Still complete validity can never be guaranteed, because ambiguous 2nd

order derivatives might be moving incorrectly if we would take 3rd order derivatives into account. Because of its recursive nature, this issue is impossible to solve completely. It is not a problem at the Garp3 level of detail however. Ambiguous 2nd order derivative are not branched into multiple states, these states are generalised in one state and therefore the problematic state transitions cannot happen.

In conclusion it can be said that the horizon of detail has moved substantially with the addition of the 2nd order derivative functionality. Furthermore modelling has been simplified because in many cases it is no longer necessary to explicitly model all possible inequalities between two influences.

4.5. Value Branching

Ambiguity does not only arise when determining derivatives, it also is an issue with calculated quantity magnitudes. For example the flow in the u-tube model in Figure 4.13 is calculated using other values.

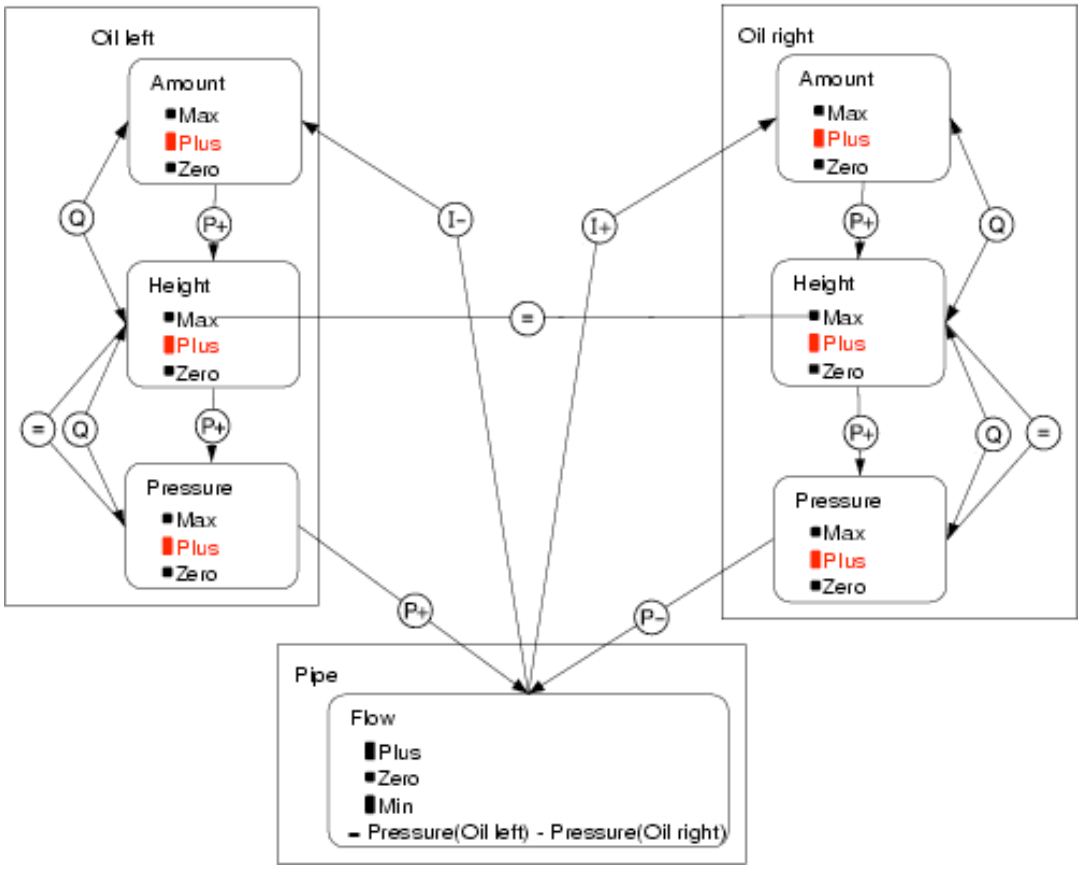

Figure 4.13: Ambiguous U-tube dependencies

Even when values are known, more inequality information is sometimes needed for a complete calculation. In our example this is the case; the flow cannot be determined. The following equations hold:

- Flow = Left – Right,
- Left = plus,
- Right = plus,

But the inequality between Left and Right is unknown so Flow can not be assigned a value. In previous versions of Garp3, to assign a value to the Flow modellers had to

construct model fragments for each possible value. The reasoning assumptions algorithm would then try each model fragment and all possible states would be generated.

We have implemented a mechanism in Garp3 to generate all possible values for calculated quantities in these cases. This branching is analogous to the branching that occurs over ambiguous derivatives in the influence resolution procedure. The value branching mechanism is active right after the search for model fragments is complete. At this point all information that can lead to a value assignment is already used. This is important because we do not want to generate unnecessary dead-end branches.

Criteria for branching over possible values of a quantity X are the following:

- The value of X is unknown
- The value of X is defined by an inequality of the form: X = Y + Z or X = Y – Z, where Y and Z may be additions or subtractions themselves

If these criteria are met, the algorithm simply tries to assign each value in the quantity space of X.

A sorting algorithm is present to order the set of candidate quantities. Take for example the following inequalities:

- A = D – B
- B = E + F
- C = A + G

And say that the values of A, B and C are still unknown and D, E, F and G are known. Now, the best order to try these candidates is: B, A, C. This is because A and C use results of their predecessor. Of course the sorting algorithm cannot perform a correct ordering if loops exist in the structure of inequalities. In this case candidates are tried anyway and a mechanism is present to make sure that unresolved candidates are retried as long as there is progress.

Another proposed solution to the value branching procedure was to generate all values of a calculated quantity right after finding the calculus relation. This approach was not taken since it may result in unnecessary work caused by dead-end branches.

An open issue that could be addressed in future work is maintaining the implicit inequalities present in each branch. Take for example the following situation.

- X = Y – Z
- X = ?
- Y = Z = plus
- dX = dY = zero

Three branches are generated where X is equal to min, zero and plus. In these states an implicit inequality between Y and Z exists (Y < Z, Y = Z and Y > Z respectively). And in our case it should not be allowed to change because of the zero derivatives of X and Y. Please note that this discussion is very similar to the one in section 4.4. The problem is less urgent however because derivatives may change through the continuity regime

and quantities still always need an explicit termination to change their value. As will be shown this allows the modeller to prevent invalid behaviour.

A naïve solution would be to add this inequality to the simulations and let it terminate explicitly. However determining a suitable implicit relation may be impossible in case a larger quantity space then ‘mzp’ is used. And also a quantity could be the result of a calculus over more then 2 other quantities. The suitable implicit relation would be a calculus relation itself and there is no termination mechanism for these relations in Garp3 at this moment. Therefore in the present situation it will be the modellers’ responsibility to model a ‘logical’ causal connection between the derivative of the calculated quantity and the derivatives of the calculus quantities. This way no ‘illegal’ transitions will occur.

4.6. Exogenous behaviours

4.6.1. Parabola

Two new exogenous behaviours were added by user request: ‘parabola (positive)’ and ‘parabola (negative)’. Quantities using these behaviours go up until their maximum and then down to their minimum or down until their minimum and then up to their maximum respectively.

4.6.2. Decreasing and Increasing stop in extreme point

The original design of the exogenous quantity behaviours ‘decreasing’ and ‘increasing’ does not allow a zero derivative. Not even in extreme points of the quantity space. Still a model can disallow for example a quantity to be increasing in the uppermost point of the quantity space. See preferences: ‘Apply quantity space constraints on zero as extreme value’ and ‘Apply quantity space constraints on extreme values’.

In the latest version of Garp3 these two exogenous quantity types are allowed to stop in the extreme points of their quantity space.

4.7. Matching states

In large models with many states, comparing new state descriptions with all existing states becomes computationally expensive. This matching process has been speeded up: Firstly by testing on the most distinguishing aspect first: values (NB magnitudes *and* derivatives). And secondly by storing these state values separately in a small data structure: state_values(StateNr, Values). This saves time because retrieving the complete state description: state(StateNr, Description), is relatively expensive since it is a large data structure. Now this retrieval and further comparison is only done if the values already match, which means a complete state match in most of the cases.

4.8. Inequality reasoning depth limit

Some models feature a lot of 'additional calculi'. Because of the nature of the inequality reasoning procedure this can lead to numerous time consuming but uninformative derivations. To make these models more tractable in the development phase a depth limit has been implemented in Garp3. This limit constrains the number of ‘parent’ relations a new derivation can have. When the limit is reached, no further combinations are made with that relation.

To further clarify why additional calculus can slow down inequality reasoning: Simple one to one relations are in general most informative in Garp3. Additions or subtractions

of quantities are mostly only useful if they appear in an explicitly modelled inequality like: Flow = Pressure_Left – Pressure_Right. Arbitrary combinations of quantities are not very informative. This idea is captured in the restriction that results of transitivity reasoning are only saved if they can be simplified.

E.g. combining:

- X > Y
- Y >= Z

Results in:

- X + Y > Y + Z,

Which simplifies into a useful new fact:

- X > Z

On the other hand X > Y and P >= Q can be combined to form X + P > Y + Q, but this result will not be recorded because no common quantity can be subtracted.

Now when a lot of additions are present in the set of relations there are many possibilities for results which can be simplified and which involve arbitrary quantity combinations. Therefore numerous extra derivations are made.

Furthermore, if an addition is present with equal quantities:

- A + B = C
- A = B

Then the 'equal quantities share equal pointers' mechanism cannot be used. This causes the set of relations to grow considerably which in turn causes more combinations to be possible.

The solution to limit the depth of the search for new derivations is a pragmatic one. It provides the means to develop and work with an otherwise intractable model. Unfortunately no guarantee can be given that all possible derivations are made when the limit is applied.

If our depth/derivations function would be monotonously decreasing, an iterative deepening algorithm could be used, stopping when no more new relations are derived. Sadly we never know if new derivations are not waiting for us around the corner: If at depth 3 no more derivations are made then at depth 5 new relations could be found. We would be happy to inform the user of the class of derivations that cannot be made given a certain limit but unfortunately this is not possible.

Most models have shown correct behaviour with a depth limit of 5. Models with some inequality reasoning happening generally show significant efficiency gains in this case. Small models even behave correctly given a depth limit of 3, there are however models known that need a depth of 10 to arrive at important derivations.

There are some possible explanations why a limit of 5 does not cause any problem in many cases. Note that these are hypotheses that need further research if claims are to be made.

- Each transitivity operation adds, replaces, or removes at least one variable. Since a typical relation in Garp3 has no more than 3 or 4 variables this relation can be completely transformed in 4 steps.

For example:

X = Y + Z	&	A < X	yields:	A < Y + Z
A < Y + Z	&	Y = B	yields:	A < B + Z
A < B + Z	&	Z = C	yields:	A < B + C

- Quantity spaces in Garp3 typically have no more then 3 points (intervals not counted). This means they can be traversed in just a few derivations.

For example:
Deriving: A > D takes only two steps given: A > B, B > C and C > D. So if our full partial ordering of all quantities and quantity space points contains paths even as long as six points the full transitivity can be derived in 4 steps. An example partial ordering like this is shown in Figure 4.14.

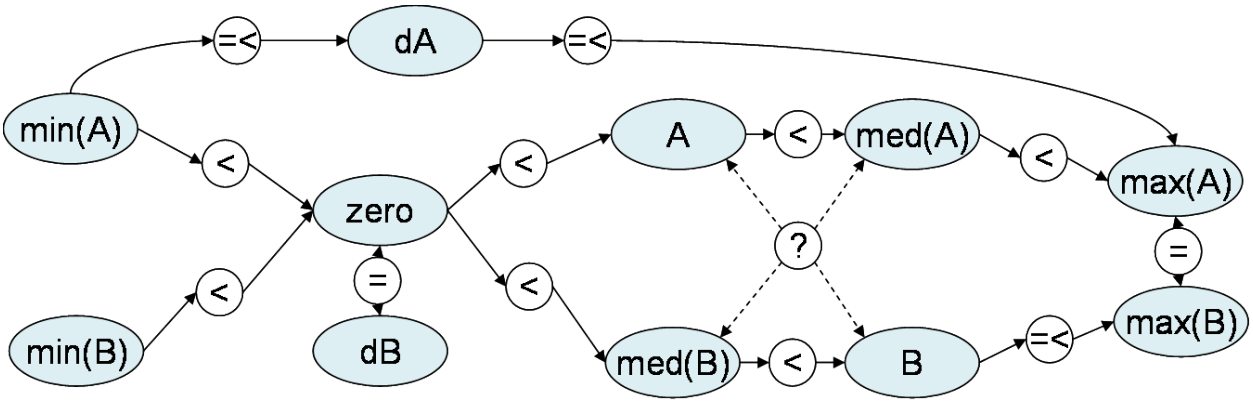

Figure 4.14: Partial ordering of all quantities and quantity space points.

This partial ordering corresponds to the situation where 2 quantities A and B both have a quantity space with points/intervals: min(X), below, zero, low, med(X), high, max(X). Quantity A has the value low; B has no value but is bigger than the point medium. The derivative of A is unknown; the derivative of B is zero. The binary relations pictured are directly known as facts to the engine. One transitive reasoning step is made by combining two facts into a new fact. As can be seen one of the longest paths in this partial ordering is for example: min(B), zero, A, med(A), max(A). And min(B) < max(A) can be derived in only 3 steps.

In conclusion we can say that the depth limit should by default be turned off to maximize the possibility of finding all possible derivations. (Note that other efficiency restrictions may still prevent this from happening.) The depth limit can be applied in a practical way to work with intractable models in the development phase. The limit size should be chosen and tested keeping in mind the number of additions/subtractions, the number of quantities and the size of the quantity spaces used.

Besides the chosen solution some other ideas have been tested that did not provide satisfying results:

- Limit the size of additions in relations. This concept does not work because uninformative additions are on average not different in size of informative additions.

- As a heuristic the depth limit could be reset to zero when a simple (thus informative) relation is derived. This could provide some more assurance for completeness. However, the efficiency gain is negatively affected quite strongly by this concept.

In future work the following ideas could be investigated:

- An efficiency gain in the inequality reasoning could be made by perfecting the normalisation of relations. In the current implementation '=' relations can be present in reverse form thus a double check is sometimes needed. The translation procedure to the internal form could be perfected by using a standard canonical form for equalities. For instance the part with the lowest pointer in it could always be on the left.
- It would be good not to combine relations from different 'worlds', (quantities that do not interact). Determining these contexts is hard however and checking context membership could be an expensive operation in itself.
- A set of items could be constructed that incorporates all known quantities, additions and quantity space points. These items may then be used to determine if a derivation is useful. This approach might be more certain about not excluding wanted derivations.

In general it can be said that Garp3 has always needed constraints on the inequality reasoning to keep simulations tractable and completeness has not been a major issue in the past. However, more research on efficient algorithms determining the full transitive closure of a set of inequalities is of course always a good thing.

4.9. Inequality reasoning deficiency, transitivity over zero needed

By design Garp3 originally never applied transitivity reasoning over the point zero. Restrictions on the inequality reasoning procedure such as these are needed because computing the full transitive closure of a set of relations can be very expensive. The rationale behind this particular restriction was that most relations can be derived in many ways and since zero is a universal value it uselessly connects many otherwise unrelated quantities.

Still this restriction proved too strict. Some models where assuming model fragments that had derivable conditions (Communicating vessels 2, Ants garden). This is obviously not acceptable.

A modification was made to the representation of the point zero so that transitivity over zero is now possible. Therefore in the latest version of Garp3 all needed derivations are made. Of course the downside of this change is that more derivations can be made and this means more processing time is required. Worst case performance became up to 30% slower, but this proves to be of no practical concern from a user point of view. Performance is no less acceptable then before.

Because inequality reasoning became more expensive after transitivity over zero was allowed it became more worthwhile to constrain its use. The influence resolution procedure provided an opportunity here because it weighs influences using inequality reasoning. In many cases however sign addition can also provide conclusive results. Sign addition is computationally inexpensive and therefore in the latest Garp3 version it is tried first. Only if the result is still ambiguous the more expensive balance equation method is used.

4.10. Epsilon ordering first:

The epsilon ordering concept distinguishes immediate terminations (from a point, from equality) and non-immediate terminations (to a point, to equality) [2]. The concept is based on the idea that a point in a quantity space occupies no space on the line of real numbers. Therefore if a quantity is on a point and it moves it will leave the point in an infinitely small amount of time. On the other hand a quantity moving to a point will have always some 'epsilon' amount of space between itself and the point. Therefore the transition to the point is not immediate. And because of this immediate transitions take precedence over non-immediate transitions. This idea is illustrated in Figure 4.15.

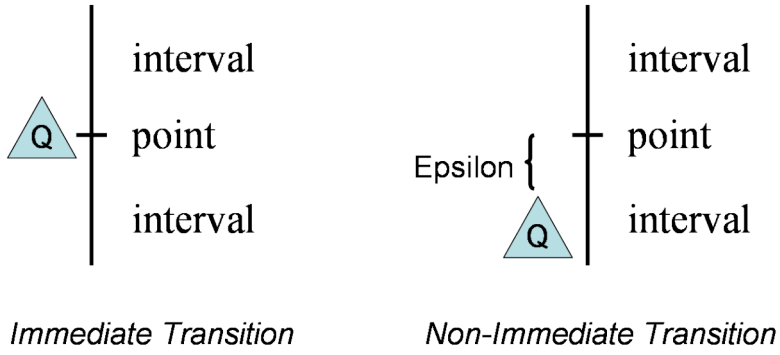

Figure 4.15: Epsilon Types

The ordering procedure in Garp3 not only determines which terminations happen first (Epsilon Ordering), it also determines which possible combinations of terminations can happen.

In previous versions of Garp3 epsilon ordering was done after determining all possible combinations of terminations. This process is shown in the right diagram of Figure 4.18. The reason for this was that if it is done first and non-immediate terminations are removed then other ordering concepts may remove all immediate transitions later on. This way we would end up with and empty set of terminations while valid non-immediate transitions are present. A drawback of this method is that the general ordering procedure works on large sets of terminations (both immediate and non-immediate). These ordering concepts are computationally expensive and these costs can grow exponentially as the size of the sets grows.

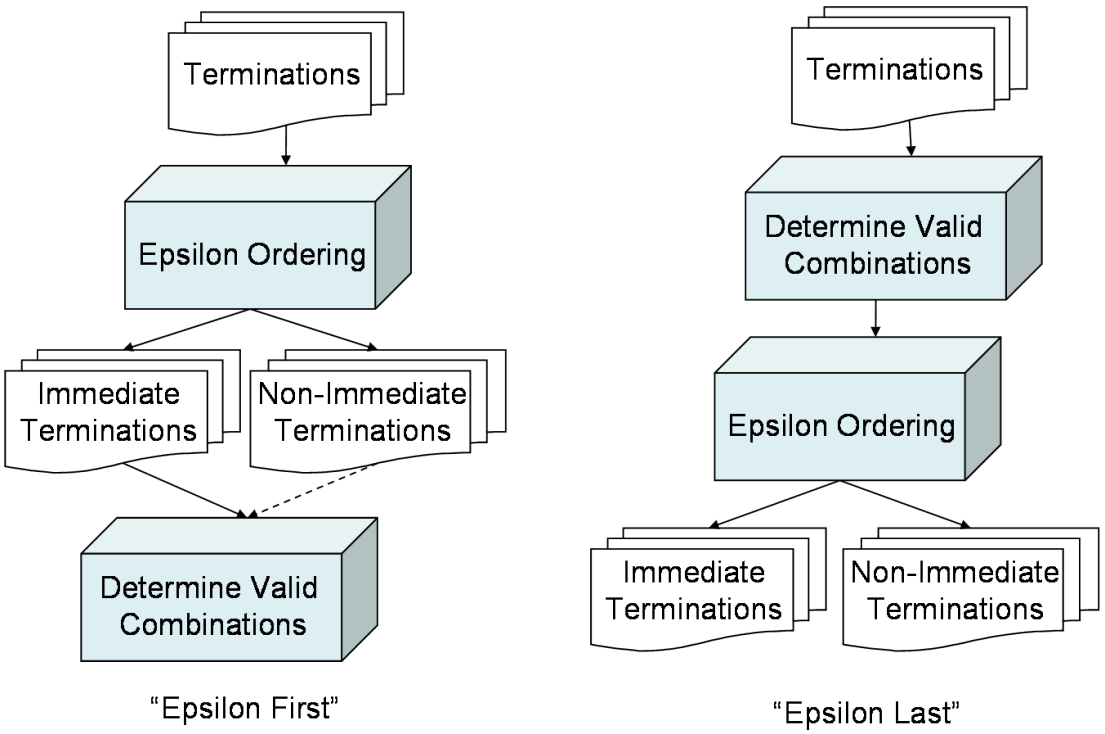

Figure 4.16: Epsilon ordering context

To improve this engine behaviour we implemented a new version of the ordering algorithm in Garp3 that applies epsilon ordering first. If there are immediate terminations, these will be ordered first. The resulting set is checked and only if it is empty the set of non-immediate terminations will be ordered. This way, valid terminations will never be removed without reason. This procedure is illustrated in the left diagram of Figure 4.16.

An extra consequence of this approach is that mixed combinations (involving immediate and non-immediate terminations) are not possible anymore. This is a positive effect since a strict interpretation of the epsilon concept would not allow these combinations either. Note that in all known models these combinations do not occur. But still for backward compatibility reasons the old ordering procedure can still be activated using the simulation preference: Apply epsilon ordering last.

The efficiency gain on our standard set of models was only moderate. This most probably results from the high number of correspondences used in these models. These already constrain the number of possible combinations very effectively. In less elaborate models with more unrelated quantities the efficiency gain is very substantial however. This situation can occur for example in models that are still in a developmental stage.

Discussion about the new approach has been mostly about whether 'mixed' transitions were possible. One question has been whether an interval and a point could be corresponding since this correspondence could demand a 'mixed' transition. Another proposed solution has been to put the epsilon ordering first in the ordering procedure and when the set of ordered immediate terminations is empty on exit then the old procedure would be used applying epsilon last. This approach was not chosen because the 'mixed' transition is quite a hypothetical and dubious case and the disadvantages of doing some work twice and working with large termination sets are far more substantial.

4.11. Epsilon derivative continuity constraints

A new member was added to Garp3 to the family of epsilon concepts. It is that derivatives may not change to zero in an immediate transition. The rationale behind this rule is very similar to the epsilon ordering concept. Firstly it is a non-immediate event since it is a transition towards a point. Secondly it is a non-immediate event since it represents the event that the balance of influences on this quantity (whose derivative is going to zero) changes from unequal to equal. And this type of inequality change is of course non-immediate. A dedicated implementation was needed here because derivatives normally do not change through an explicit termination. (An explicit termination would of course be epsilon-ordered.)

4.12. Optional reasoning assumptions

Reasoning assumptions are made in the model fragment selection procedure when no more applicable model fragments can be found and the conditions for a certain model fragment are assumable but unknown. Modelling practice during the project showed however that this system can be inconvenient in certain special cases.

The model in question had a model fragment which should only fire when a certain quantity had a certain derivative. As can be seen in Figure 4.17 however, derivatives are typically only calculated after the model fragment search is done.

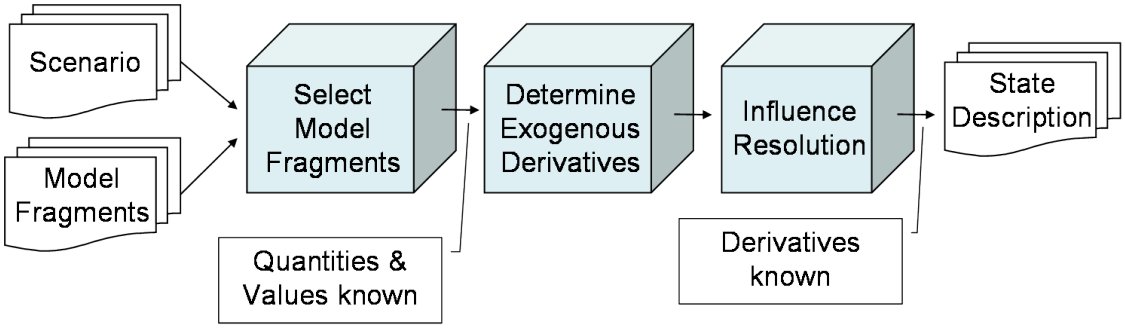

Figure 4.17: Determining the state description

Therefore, the derivative in question is not known during the model fragment selection process. And the result is that the conditions of this model fragment are assumed to be true. An unwanted side effect of this reasoning assumption is that all other possible values for this derivative are not tried or generated.

The resulting simulation was especially problematic since the quantity in question had an exogenous behaviour pattern that would set the derivative to a different value. This resulted in a contradiction which caused no states to be found. Of course a similar problem would arise if the causal model would set the derivative to a value conflicting with the reasoning assumption on the derivative.

A modelling solution to this problem is to construct model fragments with all possible derivative values and combinations to force the reasoning assumption mechanism to try all possibilities. This is however difficult to do for less advanced users. And moreover it makes the model less concise and insightful.

Two simulation preferences have been made in response to this problem. The first one is a switch turning off the reasoning assumption mechanism for conditions about derivative values and derivative inequalities. This effectively solves the problem

described and further minimally changes the reasoning engine behaviour. The option does not interfere with the average model since most reasoning assumptions are not about derivatives.

The second simulation preference is a switch turning off the reasoning assumption mechanism completely. Now model fragments will only fire when all conditions are known to be true. This option off course represents a major change in engine behaviour and should be used only when needed. However it may prove useful in tutorial programs for novice modellers and may therefore be used as a feature to smooth out the Garp3 learning curve. Beginner users typically do not expect reasoning assumptions to be made and it is probably instructive and intuitive not to fire model fragments unless all conditions are known. In other words: It is good to enforce novice modellers to build very complete models in terms of all conditions and consequences.

4.13. Minor fixes and efficiency improvements

4.13.1. Two unconnected entities trigger a child model fragment only once

This bug concerned the algorithm that searches for applicable model fragments. In the case that a scenario had two unconnected entities of the same type a child model fragment with that entity type would not be found twice but only once. To solve this problem the check for the isa-conditions: "is the parent there" was improved.

A side effect of this fix is that empty model fragments do not become active anymore. This is not a negative effect since the rationale for firing model fragments is that there are conditions that match the given situation. Since these are not present in an empty model fragment it will not fire.[4]

4.13.2. Engine runs out of memory.

Prolog has a limited amount of memory assigned. This way a faulty program will run out of memory in a reasonable amount of time. However, for large models more memory is needed. This is in line with the trend of larger more elaborate programs being built in prolog. In response to these modern memory requirements Swi-Prolog had more memory assigned by default.

When using older versions of prolog a start-up prolog script file can be used to assign more memory. The only line of this script should be:

‘ #!/usr/bin/pl -G32m -T32m -T16m -s startup.pl ’

NB: The ‘pl’ directory is only important for non windows machines. Windows prolog will ignore this directory setting.[5]

4.13.3. Exogenous sinusoidal behaviour in single interval

The exogenous quantity behaviour ‘Sinusoidal’ had a problem when using a single interval quantity space. As expected, 3 states were found: respectively with a negative, zero and positive derivative. But no transitions were generated.

[4] In the Mantis bug report system, this issue can be found as bug number 43.
[5] In the Mantis bug report system, this issue can be found as bug number 59.

After fixing this bug the sinusoidal quantity behaviour worked as expected. The resulting simulation is given in Figure 4.18.

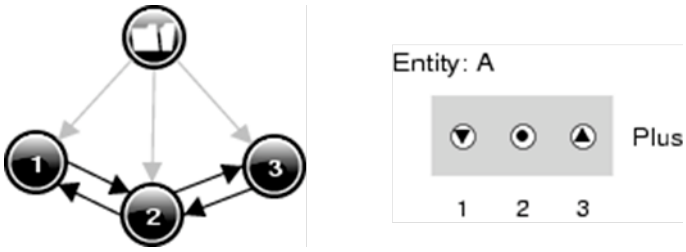

Figure 4.18: State Graph and Values of Sinus in single interval

An interesting aspect of this simulation is that invalid paths are possible. Sinusoidal would be: 1->2->3->2->1 etc. (up, stable, down, stable, up). But in this graph: 1->2->1 etc. (up, stable, up) or 3->2->3 (down, stable, down) is also possible, which is not sinusoidal.

No 'fix' was implemented for this case because it would result in a second stable state which is qualitatively not different from the first stable state. The difference is that one state comes from the upward state and must go to the downward state; the other comes from the downward state and must go to the upward state. But as a state-snapshot they are equal and therefore this difference is not expressible in the Garp3 vocabulary. Note that this is not a conceptual poverty on the vocabulary side, but an inherent conceptual poverty on the side of the exogenous variable that does not satisfy the explicit mechanism approach [1]. The causes of its behaviour are not modelled and therefore the state does not have qualitative information to distinguish the two states and rule out this unwanted behaviour. Lastly, a very practical reason for not 'fixing' this case is that having such 'twin' states would be very confusing to modellers.

In conclusion we can say that the sinusoidal exogenous behaviour does not work completely correct in a single interval quantity space and therefore it should not be used in these circumstances.

4.13.4. Remove weak relations / Analyse Zero

Efforts on fixing the bug in the exogenous quantity behaviour 'Sinusoidal' in a single interval quantity space uncovered another problem. A transition from a state to itself was found. In this case the inequality reasoning failed to derive an obvious contradiction. The fix concerned a procedure that removes superfluous weak relations. E.g. X >= Y is superfluous if X > Y is known. This procedure was not functioning correctly in rare cases. The fix also concerned the 'analyse zero relations' procedure that removes superfluous relations by optimising pointer assignments.[6] This procedure can also detect inconsistencies and is now done more often. To compensate for the extra reasoning effort the procedure was speeded up by making use of low-level bitvector notation, thereby saving translation time.

In future efforts the analyse zero procedure may be incorporated in the more general pointer optimisation procedure. This is possible because of the changes made to the representation of zero by the transitivity over zero fix. The expected performance gain is quite low however.

[6] Internally the reasoning engine uses pointers that refer to quantities and bitvectors to represent sets of pointers. This language allows fast, processor level computation of transitivity reasoning.

5. Support for Multiple Languages and Tooltip

Garp3 has been augmented with two features that help users in better understanding a model and its details. The multiple languages feature allows modellers to create a single model using multiple languages. When a model is created this way, users can select a language that best matches their needs and abilities. The tool-tip feature gives modellers the opportunity to provide textual explanations for ingredients defined during modelling. When simulating and inspected a model the tool-tip function shows these textual explanations when hovering over the ingredients shown in the Garp3 interface.

5.1. Multiple languages

Humans are by nature most accustomed to their own native language. Working with a model in that language would be best to appreciate its contents most. Garp3 has therefore been augmented with the facility to have a single model present itself in different languages. The translation between languages is not automatic however, because making such a translation-feature would be outside the scope of current NaturNet-Redime activities. Instead, multiple languages have to be supplied by the modeller (or user). Garp3 provides the means to do so and allows users to select the language they like best from the languages available. The multiple languages feature can be assessed from the Build environment (pull-down menu: View → Translations…). A screenshot of the interface is shown in Figure 5.1. The interaction with the 'Language editor' is somewhat different from other editors. After selecting a field in the editor the keys 'Control <arrow up/down>' can be used as shortcuts to move to the next field (up or down). If, while a field is selected, new text is type, this text will replace the old one at the moment of moving fields using the control option. Although at first this interaction may seem a bit unusual, it works very fast which makes it easy to do a translation quickly.

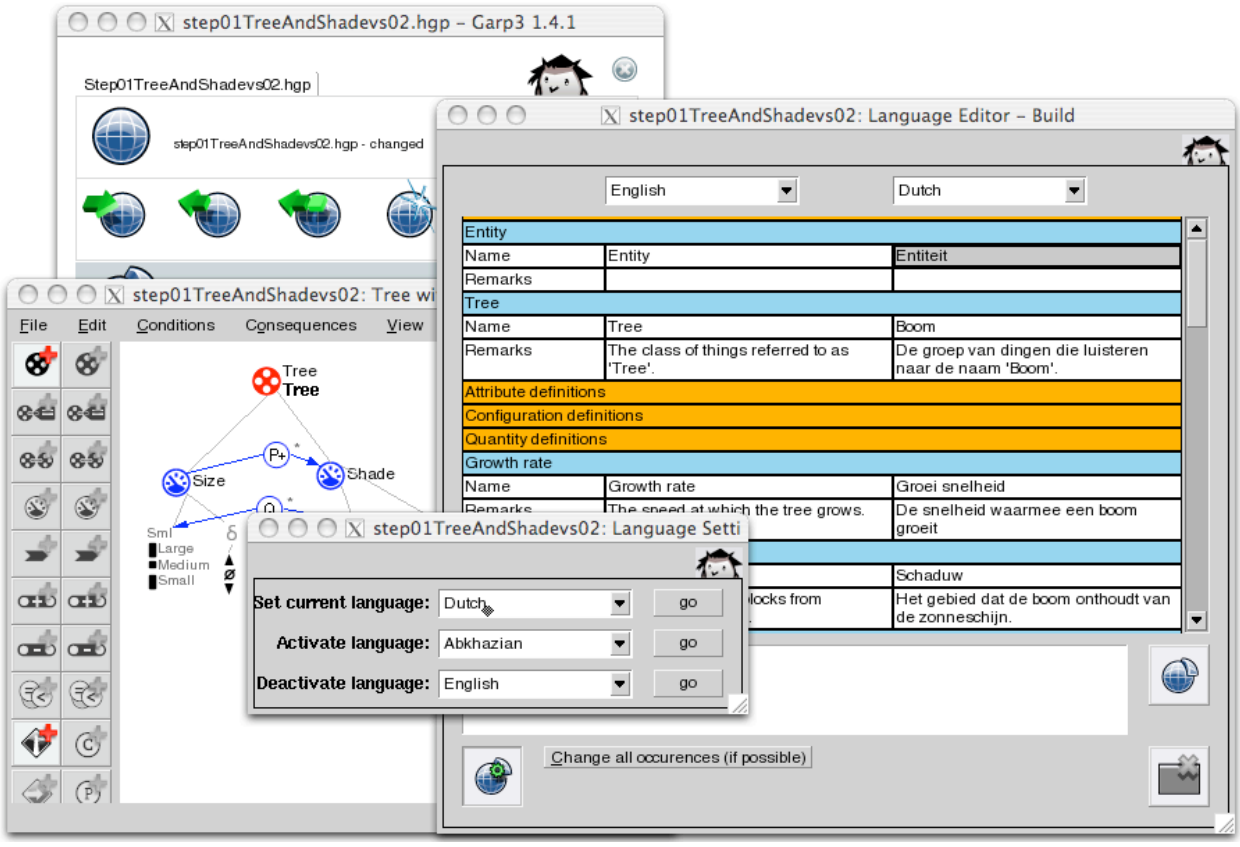

Figure 5.1: Illustrating the use of the multiple languages feature.

Some specific details of the multiple languages feature are:

- The set of languages that can be provided is ISO639 compliant.
- Translating is done in a dedicated editor, in which all the terms from the model are enumerated and sorted, and presented as a list of items. This simplifies the translation process by minimizing the number interface activities.
- When a new language is activated, the terms from the 'current language' are automatically copied to the new language, after which the modeller can change each of the terms in the new language.
- Names of ingredient types are changed throughout the model.
- Names of instances are changed throughout the model when the option 'Change all occurrences (if possible)' is activated.
- When, after having provided additional languages, in the current language new ingredients are added, these new ingredients get 'undefined' labels in the other languages, such as: 'Please_translate_this_30057344'. If these are not translated, the model will just use these labels when the language is set to be the 'current language'.

5.2. Tool-tip

The tool-tip facility was already part of the single user implementation of Garp3 [3]. However, it turned out that the feature was too much oriented towards the ingredient *types* and not enough towards the 'remarks' provided as explanations within a model. Moreover, the feature was mainly present in the Build environment and much less in the Simulate environment. These two aspects have been changed and the tool-tip feature together with the remarks field in the Build environment can now nicely be used to capture and communicate ground-level explanations of the ingredients defined and used in a particular model. By separating the 'model ingredient tool-tip' from the 'interface action button tool-tip' the new feature was further enhanced. The tool-tip text explaining buttons is always active, because it is an essential support in operating the Garp3 software. The tool-tip for showing ingredient explanation can be turned on or off, and thus be activated as needed in a context of use. Switching the ingredients tool-tip on or off is typically done the Build environment (pull-down menu: View → Hide/Show model ingredient tooltips) or in the Simulate environment (pull-down menu: Display → Settings → Show model ingredient tooltips). The basic idea of the tool-tip ingredient is shown in Figure 5.2. On the RHS (bottom) it shows that while creating a correspondence between the quantities Size and Shade the modeller added a brief explanation in the 'Remarks' field, namely: "Size and Shade have corresponding magnitudes." When simulating the model this text is shown as tool-tip whenever the icon of this particular correspondence appears in the interface (both in the Simulate and the Build environment), in this case in the dependencies screen opened for state 3: "MF Tree and shade: Size and Shade have corresponding magnitudes." Although the example is relatively simple, the idea is that modellers can now provide essential explanations (by added text in the 'Remarks' field) for all the ingredients they create in a model. These ground-level explanations will greatly support later users of the model in understanding what the modeller had in mind when creating it.

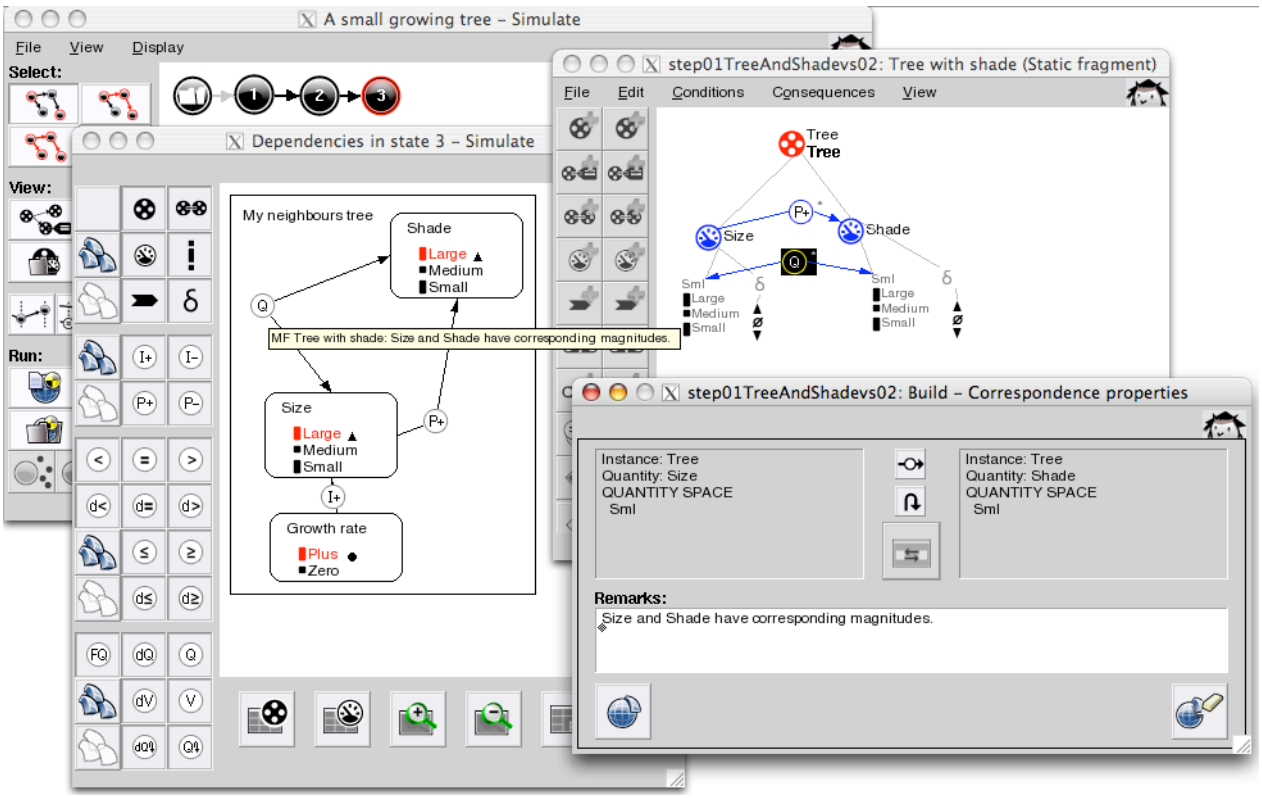

Figure 5.2: Illustrating the use of the ingredient tool-tip feature.

Some specific details of the ingredient tool-tip feature are:

- Whenever an ingredient is created and a 'Remarks' field is available, the model builder can provide text to explain the role of the ingredient in the model, which will then be used as ingredient tool-tip text.
- There are typically three places where an ingredient is defined and/or re-used that may call for entering explanation text in the 'Remarks' field:
  - Definitions. This happens when an ingredient is newly defined, e.g. an entity in the 'entity hierarchy editor'.
  - Re-use in model-fragments. This happens when an ingredient is defined within the context of a model-fragment, e.g. placing a previously defined entity as a condition in a model-fragment.
  - Re-use in scenarios. This happens when an ingredient is defined within the context of a scenario, e.g. placing a previously defined entity as an existing 'thing' in a scenario).
- A particular ingredient may be defined multiple times within a single model. The tool-tip feature accumulates all these occurrences (as far as they are active in a particular simulation) and shows them as a list of items when hovering over the ingredient on the screen. The name of the context in which the text was given is provided as a reference for the user (e.g. in Figure 5.2 "MF Tree and shade:"). This reference has two aspects: the type of editor in which it happened and the name of the ingredient involved, thus: <Type Name>. The following shorts are used for types: EnH (Entity Hierarchy editor), AgH (Agent Hierarchy editor), AsH (Assumption Hierarchy editor), AD (Attribute Definitions editor), CD (Configuration Definitions editor), QD (Quantity Definitions editor), QS (Quantity Space definitions editor), MF (Model Fragment editor), and SC (SCenario editor).
- A tool-tip feature is in general not meant for displaying large bulks of text. It is therefore advised that modellers keep the texts they provide in the 'Remarks' short and insightful. Modellers should also beware that text are being

accumulated and that a set of ingredient explanations should ‘aligned’. Brevity is also advised for the names given to ingredients in general. Long names will clutter the tool-tip text, potentially masking the important text.

- The ingredient tool-tip feature can be Activated or De-activated via the ‘View’ pull-down menu in Build environment and via the ‘Display’ pull-down in the Simulate environment.
- In the interface of Garp3 the display of Quantity spaces are often intertwined with the display of Quantities. As a consequence, ingredient tool-tips for these two items are often shown together.
- The ingredient tool-tip feature is not present in all windows that show an ingredient, mainly because in some cases this would be rather superfluous and partly because resources to rigorously implement the feature are missing. The feature is however available in all key interface windows.

6. Improvements to the Simulate environment

6.1. Improvements to the State-Transition Graph

The State-Transition Graph has been improved in several ways to address comments from users asking for more efficiency and flexibility in the selection of states and paths. The path search algorithm has been improved to increase efficiency. Furthermore, the user is provided with more control over the search mechanism with the following new options, which can be accessed from the Display settings in the main window of the Simulate environment, under 'Search and selection preferences' (see Figure 6.1):

- *Search for cyclic path (default: on).* When this option is turned on, the algorithm will first look for cyclic paths, before considering non-cyclic paths. Turning off this option can therefore speed up the search process, if one is not interested in cyclic paths.

- *Search for path from begin to end state or end loop (default: on).* When this option is turned on, the algorithm will automatically extend the selection of paths to include the nearest begin and end state, or the nearest begin state and loop at the end. This means that the shortest 'full path' through the selected states is selected. When turned off, the algorithm will result in the shortest path connecting the selected states. Turning this option off may save time.

- *Select end states in individual states mode (default: off).* When this option is turned on, Garp3 will automatically select all (temporary) end states when the simulation is run, or when the button 'Select individual states' is pressed.

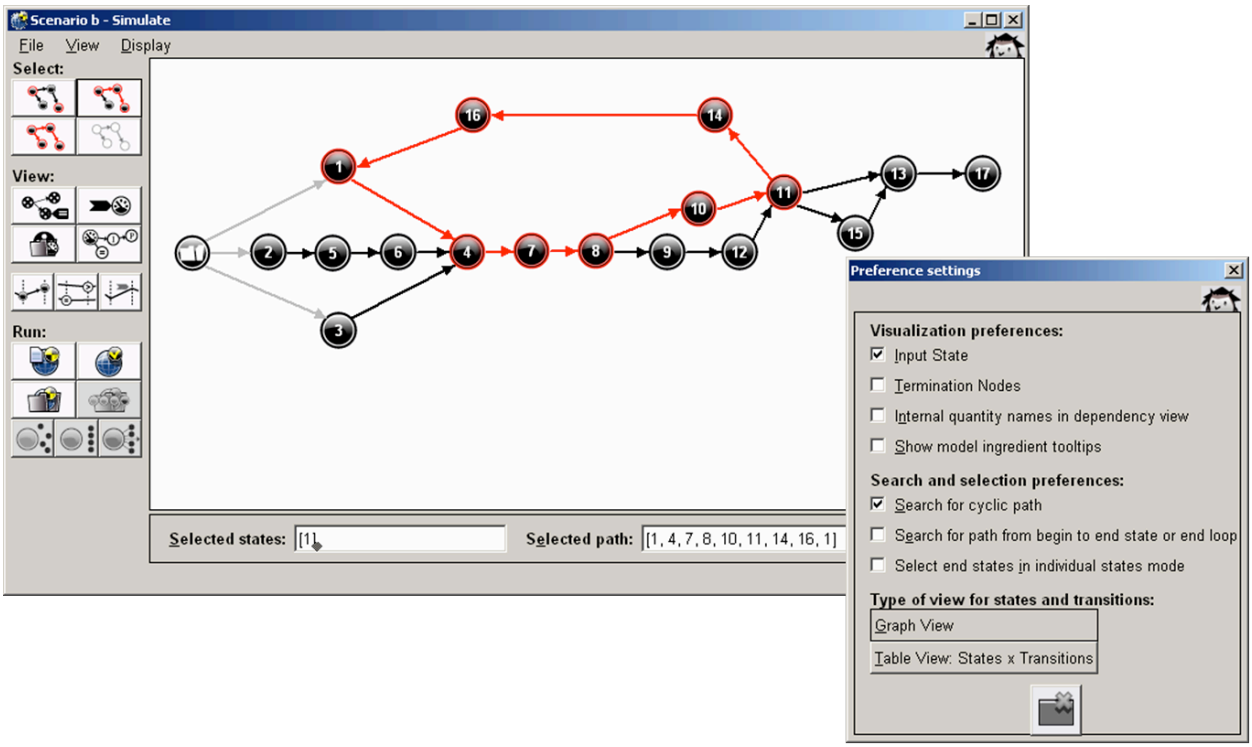

Figure 6.1: The improved State-Transition Graph View, with search and selection preferences. The figure shows the selection of a cyclic path.

Also, it is possible again to select particular states by entering state numbers (as a comma-separated list of integers between square brackets, e.g., *[1,10]*) in the 'selected states' text field at the bottom of the main Simulate window. This option was present in VisiGarp, but had been lost in the integration process leading to Garp3.

Finally, when memory runs out during the search for a path, Garp3 does not crash anymore, but instead returns the message *'No path found'* in the 'Selected path' text field. When Garp3 is busy searching, this text field displays the message *'Searching…'*.

See also section 6.2, which describes a new view (The Table View) to complement the State-Transition Graph View.

6.2. The Table View

The Table View is a view that complements the Graph View in displaying states and state-transitions. As shown in Figure 6.2, the Table View displays all states and state-transitions in the simulation in a grid with the states (and the scenario) on the X-axis and the transition identifiers for each state shown below the originating state on the Y-axis. The transition identifiers are identical to the ones used in the Transition History View and Transition Details View, showing the originating state, followed by a '*t*' and an index number for the transition, *e.g.*, '*6t2*' indicates the 2^{nd} transition originating from state *6*. In the grid, every other column is coloured light grey (which is easily discernible from the other, white columns, but subtle enough as a background colour for the display of the other graphical elements), to make it easier to discern between states with uneven and even numbers. For the Table View, the same options for selecting and inspecting states and paths have been made available as for the Graph View. Selected states and transitions are highlighted in red, which is especially useful in the Table View when a path is selected.

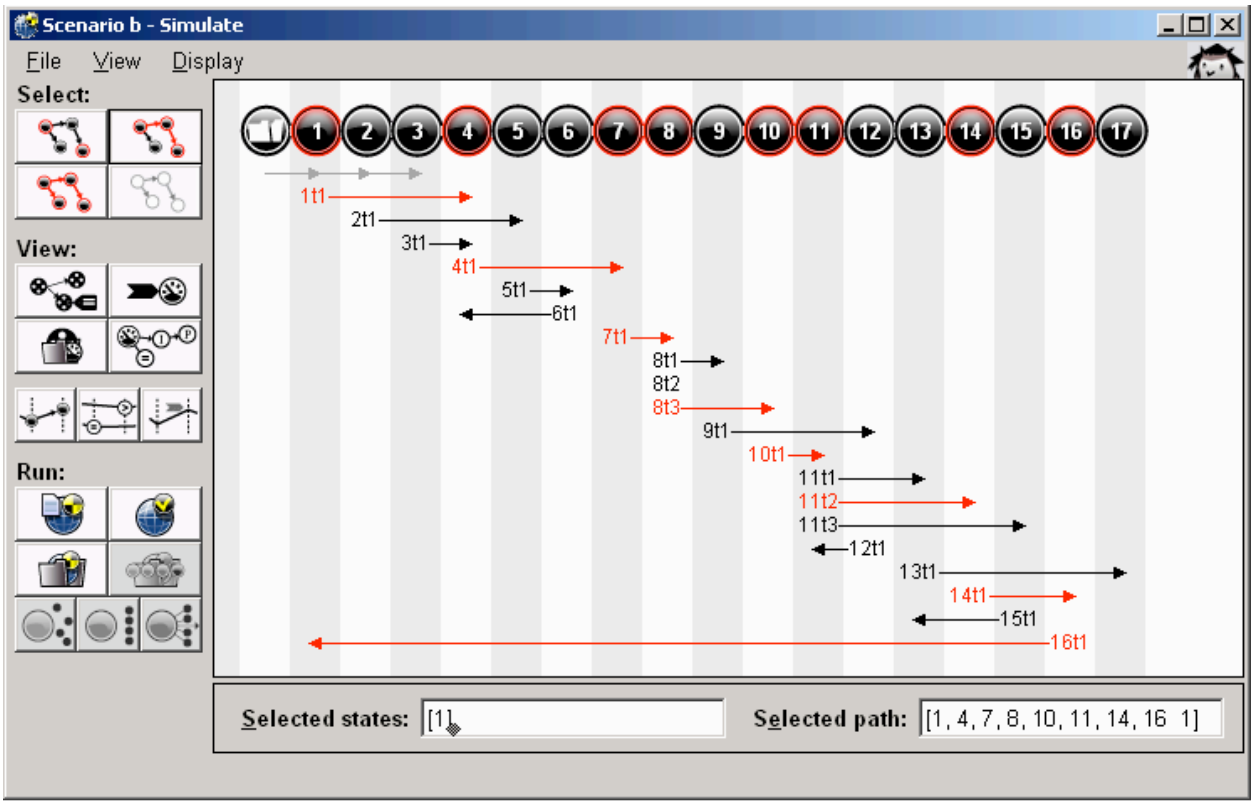

Figure 6.2: The Table View, displaying the same states, transitions, and selection of a cyclic path as the Graph View in Figure 6.1.

The Table View facilitates searching for particular states and state-transitions, since their location is systematically determined by the table format. This makes it different from the Graph view, in which the layout of the Graph is determined using a non-deterministic algorithm, which makes it harder to find a particular state in a large graph. Since both views have advantages (the Graph View facilitates recognition of paths, and is generally more space-efficient), the user is offered the choice between the Graph and Table view. This choice is built in as an option (*Type of view for states and transitions*) within the Display settings in the main window of the Simulate environment, which allows switching between the two alternative views (*Graph View* or *Table View: states x transitions*). The default option is the Graph view.

Note that the Table view is not strictly a table, since it also includes arrows pointing from the transition identifier to the column of the destination state. Another way of displaying the destination state that has been considered was to show the destination state number within the 'table cell' with the transition identifier, but since the transition identifier already includes two numbers, this option would have been confusing, and was therefore discarded in favour of the arrow representation. The arrow representation also resembles the representation of transitions in the Graph view, which facilitates switching between the two views.

7. Improvements to the Sketch environment

At the third WP4/WP6 workshop on qualitative reasoning and modelling, held in Birini, Latvia, 25-29 September 2006, domain experts participated in an evaluation session with the Sketch environment in Garp3 (see NNR milestone report M7.2.3 [4]). This evaluation session resulted in comments and feature requests, which have been addressed by improvements to the different editors in the Sketch environment.

Most importantly, the sketch environment now includes functionality for importing information from one sketch to another, which facilitates creating sketches that form a consistent and coherent set of representations about the domain. The rationale behind the import functionality is based on the relationships between intermediate representations in our structured modelling methodology, as depicted in Figure 7.1. The top half of this figure shows how the concept map is refined into the other sketches, and how these intermediate representations provide details and starting points for more detailed sketches, such as the state-transition graph (i.e., *expected behaviours map*). The metadata and order of Sketch editors have also been reorganized slightly, to conform to this structured modelling framework.

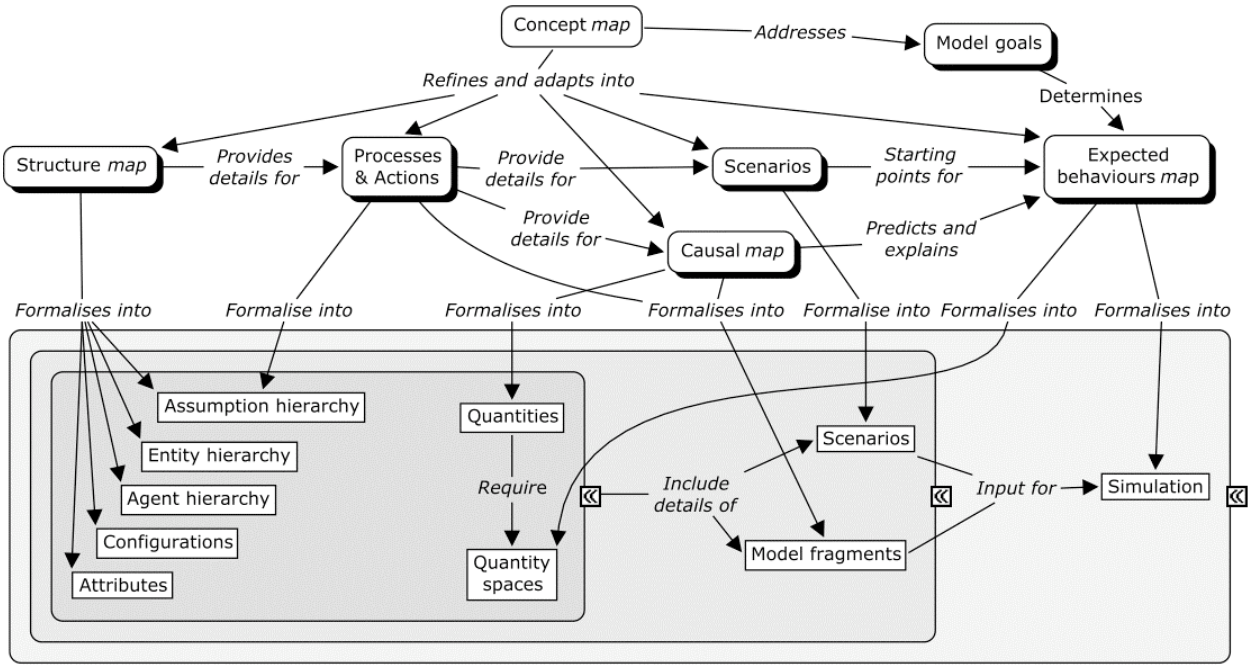

Figure 7.1: An overview of intermediate representations in the structured modelling methodology.

More details about each of the different editors in the Sketch environment are presented in the following sections.

7.1. The Concept Map Editor

As requested by users, it is now possible to save multiple concept map sketches within one model and open a saved sketch to edit. Furthermore, functionality has been added which allows importing concepts and relations from the concept map editor to the other sketch editors. The way in which this is done differs slightly for each target editor, and will be discussed there in detail.

The following issues remain to be considered for future work:

- the use of differences in node size, text font, or colour for different types of concepts (e.g., to focus on particular concepts in a map);'
- allow the creation of a different tool tip for every single relation instance;
- add more hooks to the graphical concept nodes for relations to connect to;
- the metadata window disappears when you push 'save', which is slightly unexpected;
- when adding a new relation or concept, delete the existing text with one click (instead of deleting after full selection).

7.2. The Structure editor

The structure editor allows saving multiple structure model sketches and opening a saved sketch to edit.

Import functionality has been added to allow importing (part of) the concepts and relationships from the concept map into the structure editor. It is possible to assign specific structural types (*entities, assumptions, agents*) to concepts, or leave the type *unknown*, as shown in Figure 7.2.

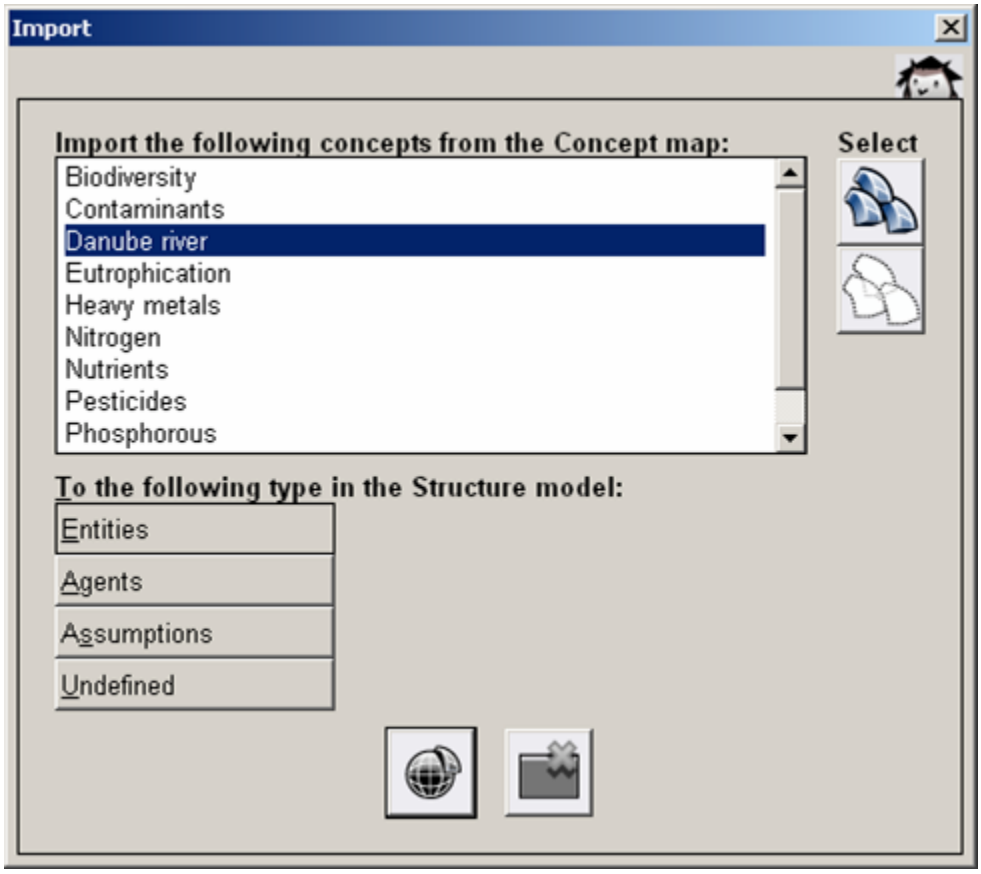

Figure 7.2: The import functionality for the Structure editor.

A bug related to updating remarks in sketch (structural) relations has been fixed. Zoom functionality to allow more concepts to be represented in one screen is still considered useful, but has not been implemented yet. This will be considered as future work.

7.3. The Causal model editor

The causal model editor allows saving multiple causal model sketches and opening a saved sketch to edit.

Import functionality has been added to allow importing (part of) the concepts from the concept map as *quantities* into the Causal model editor. It is also possible to import quantities from Process definitions and Agent and external influence definitions. Other types than *quantities* (such as the structural types offered in the Structure editor) are not relevant in the context of the Causal model editor, so they are not provided here. Importing relationships from the Concept map editor (to become causal dependencies) was also not deemed useful, because in our experience, a concept map generally contains very few, or no causal relationships.

The following issues remain to be considered for future work:

- find a way to differentiate quantities which will be associated to different entities. Currently, unique names for quantities are required, but this is not the case in in the Build and Simulate environment, where different entities can have a quantity with the same name;
- think about including assumptions in the causal model (e.g., this influence only happens under certain conditions).

7.4. The State-transition graph editor

The state-transition graph editor allows saving multiple state-transition graph sketches and opening a saved sketch to edit. The following bugs and issues have been fixed:

- sometimes a statement is not inserted after saving;
- changes to quantity names in the state graph also affect the causal model;
- when you change the state ID to a non-numerical value, things go wrong;
- multiple instances of same quantity name possible in the state properties window;
- added graphical element (bubble-tail) to state to make this type of sketch easily discernable from the state-graph in the Simulate environment.

Furthermore, the interface for editing inequalities, quantities and values has been improved in several ways to allow more flexible interaction, and to use more explicit representations (see Figure 7.3):

- added radio buttons to select the type for the right-hand side of the equation (*Values* or *Quantities*);
- added buttons (+ and -) to add and delete a quantity or value;

Regarding the import functionality, importing quantities from the causal model editor was built in as an automatic procedure in previous versions of Garp3, but this automatic procedure can now be turned on or off (default), and a manual import procedure has been added which works analogously to the import functionality in other sketch editors.

Remaining issues to be considered for future work include:

- find a way to differentiate different entities with the same quantity (see also the related issue for the causal model editor);
- replacing the ‘Save’ button, which is located on the bottom of the window, far away from the equation that will be saved, by an ‘arrow-up’ button directly below the list of equations.

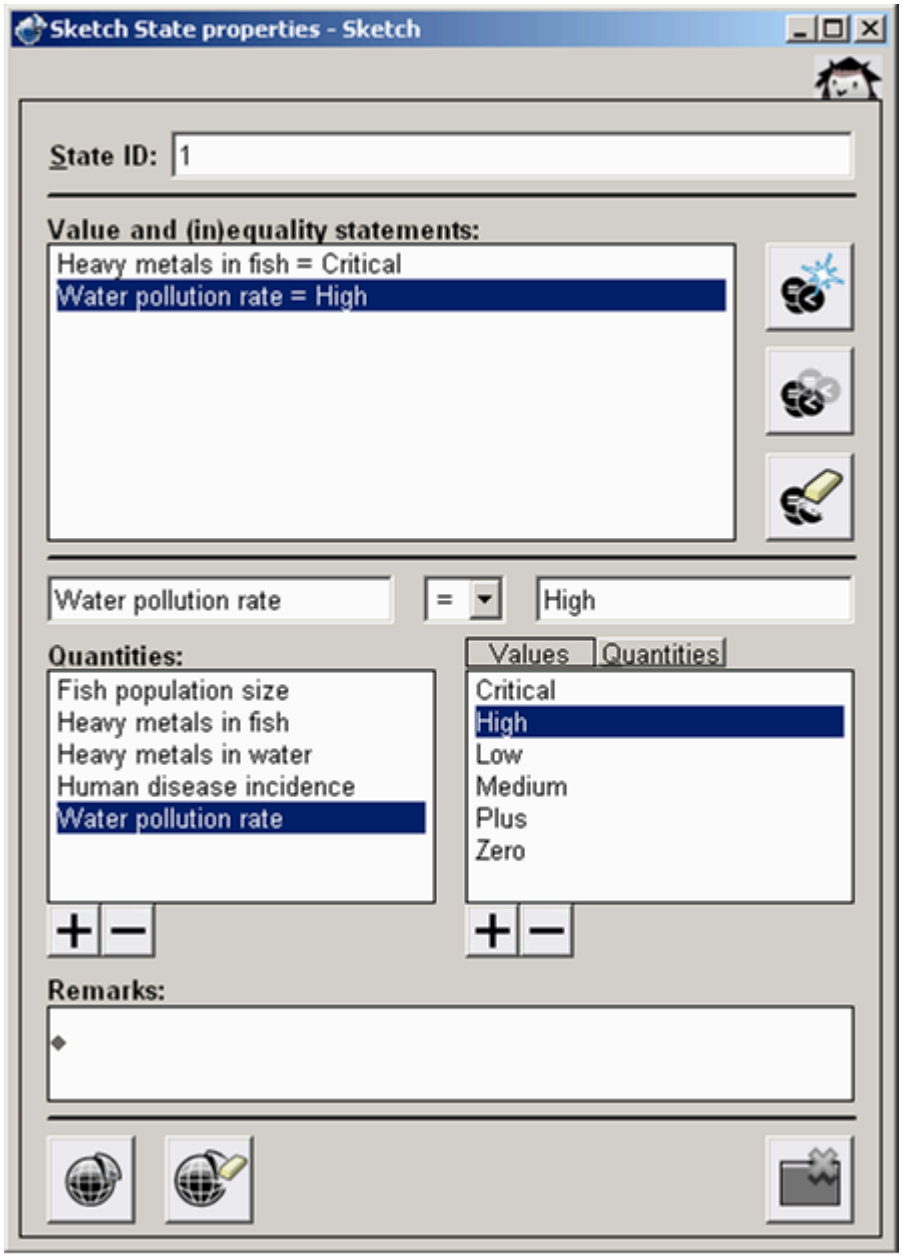

Figure 7.3: The improved interface for the Sketch State properties editor.

7.5. The Process, agent and scenario definition editors

In the Process definition editor, the Agent definition editor, and the Scenario definition editor, the specification of quantities, entities, agents, and assumptions has been expanded. Instead of consisting of one text field, they can now be specified individually, with name and remarks. Definitions from old versions are updated automatically.

Import functionality has been added to allow importing concepts from the concept map editor and entities, (agents), and assumptions from the structural model editor to process/agent/scenario definitions. The import window offers a choice for the target type between *entities, (agents), quantities,* and *assumptions*. As agents are not relevant in process definitions, the choice for the *agent* type is left out in the import window for the process definition editor. In the scenario definitions editor, it is also possible to import quantities from the causal model editor.

7.6. General improvements to the Sketch environment

In addition to the improvements mentioned in the previous sections, the following small improvements have been made to the Sketch environment:

- the keyboard shortcut CTRL+N is used throughout the Sketch environment to add a new element. The keyboard shortcut CTRL+C is no longer used, to be reserved for 'Copy' functionality, to be added to the Sketch environment in the future.
- the Sketch environment now also utilizes relative positions for recording screen layout, as is used in the Build environment.
- added ps-files for sketch icons, allowing export from sketch to eps-files.
- added tabs with more room for text in abstract, model goals, intended audience, and general remarks

In order to clarify the goal of each sketch editor, and the incorporated representations, we have created help pages on the QRM website for each editor, which are directly accessible from Garp3 by pressing the Owl-icon. The help pages are described in more detail in NaturNet-Redime project report [5].

7.7. Future work on the Sketch environment

The following feature requests will be considered for future work:

- the possibility to generate textual output from the Sketches (or even the whole model);
- the possibility to enter characters which are currently forbidden in the names of concepts, entities, agents, assumptions and quantities, such as hyphens, apostrophes, etc., as well as capital letters, which are currently transformed to lowercase (except for the first character, which is always capitalized);
- the addition of an undo option/key (it should be noted, however, that many Garp3 dialogs already have a 'cancel' or 'undo changes' button);
- make the direction of the arrow more predictable when creating relationships;
- the possibility to select or delete multiple (or all) items at once;
- have a keyboard shortcut for 'Select all';
- add copy functionality analogous to the Build environment.

In addition, future work will address support for transforming the sketch representations into running simulation models (i.e., support the bottom part of Figure 7.1). This is expected to further enhance structuring of the model building process, and will support users in refining their initial conceptual ideas into detailed representations of expert knowledge.

8. Other Improvements

8.1. Improved EPS Support

Diagrams and definitions created in the build, simulate and sketch environments can be exported as Encapsulated Postscript Files (EPS) files [6,7]. EPS files are vector-based image files, i.e. they mathematically describe the image. The advantage over traditional raster-based images (such as JPEG, GIF and PNG) is that EPS files have infinite precision, that is, you can zoom in indefinitely without the image degrading. Due to these qualities these images are particularly suitable for use in publications. Raster-based screenshots do not have high enough resolution for printing. In the latest version of Garp3 we have improved the EPS export by perfecting the output (particularly some of the images and the bounding boxes of the diagrams). Figure 8.1 shows an EPS file converted to a PNG file that shows the quality difference compared to normal screenshots (Figure 8.2).

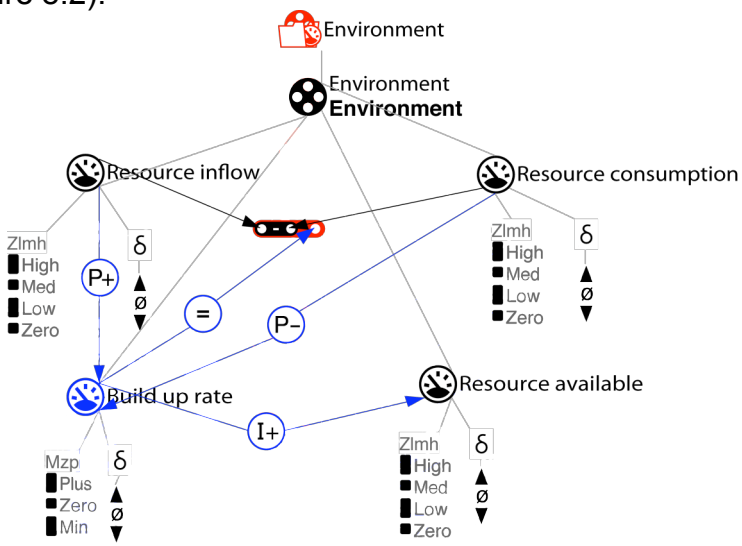

Figure 8.1: An EPS image of a model fragment converted to PNG.

8.2. Improved Support for Multiple Models

One of the recent major changes to Garp3 is the addition of multiple model support [7]. This feature makes it possible to work on multiple models at the same time, and was a requirement to be able to copy model parts from one model to another. Since the introduction of multiple model support several improvements have been implemented.

Firstly, multiple users commented that it was difficult to remember to which model a certain editor belongs. Since every editor in Garp3 opens in a new window, it was difficult for modellers to keep track. To remedy this issue all editors (and dialog windows opened by editors) now indicate the name of model that they are editing as the label of the window (see Figure 8.2).

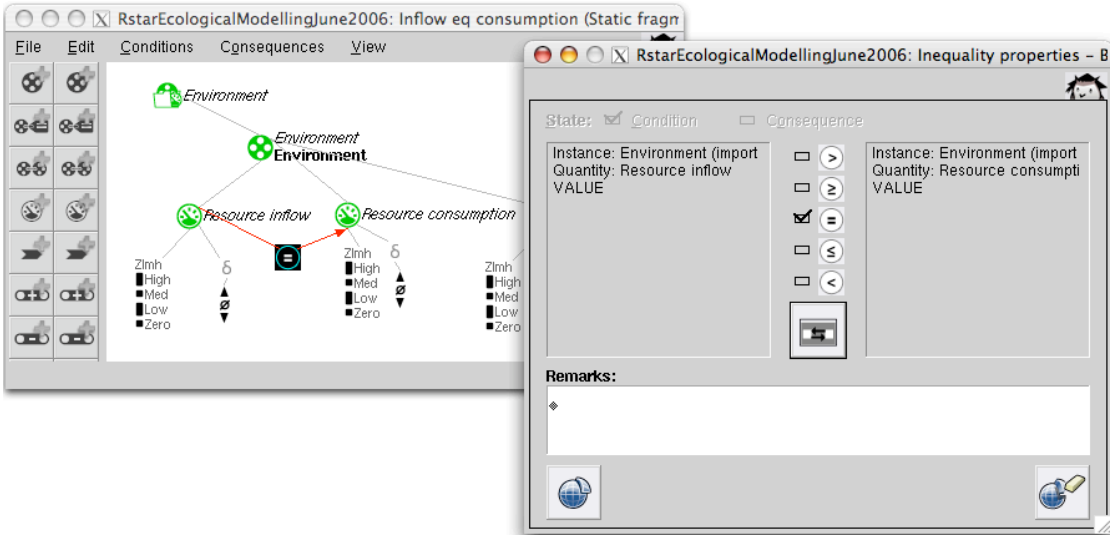

Figure 8.2: An editor and dialog window showing the name of the model in the window titles.

Secondly, a modeller noticed that in some rare cases editor windows were not correctly updated. This was caused by the solution chosen to implement multiple model support, and only happened when a change to a model happened due to a sub-change. Sub-changes are changes to a model that occur due to another change in the model. For example, if a quantity is removed from a model fragment, dependencies such as inequalities, proportionalities and influences connected to this quantity are also removed. The issue occurred with a specific type of sub-changes. When a model ingredient was removed from a parent model fragment, which had dependencies connected to it in a child model fragment, and both editors were open, the child editor was not correctly updated. For example, when a quantity in a parent model fragment was removed, and there were dependencies on that quantity in a child model fragment (which inherited all the ingredients), the child model fragment editor was not correctly updated. Since this child model fragment editor was not often open at the same time as the parent model fragment this issue was seldom noticeable. The reason this issue occurred was that sub-changes, unlike normal changes, are not passed on to an editor as an argument (and it is not possible to change this). This makes it impossible to determine which model the sub-change acts on based on the editor argument. As a solution, each change (and sub-change) now stores the model it is acting on, as does every editor. After a change to the model, all editors that are associated to the same model as the change are updated (if the type of change requires it).

Finally, clicking on dialog windows (the small windows opened by editors) did not make the correct model active. This could potentially cause Garp3 to crash. Modellers could switch to another model, and when returning to the dialog window the correct model would not become active. As a result, the change applied by the dialog window would be applied to the wrong model, sometimes causing crashes. In practice, modellers immediately work through dialog windows before making other models active. As a result, this bug seldom caused issues in practice. This issue is now resolved by making the correct model active when clicking on a dialog window.

8.3. Changes due to multiple language support

8.3.1. Changes to copy functionality

As a result of the addition of the support for multiple languages (Section 5.1) the copying functionality required some updating. Several new issues came up which were not present before. For example, two models might overlap, i.e. contain the same model ingredients, but might have a different active language. As a result, the overlap would not be detected and redundant model ingredients would be added. Another issue is that for model ingredients only the active language would be copied. Translations of the content of the model ingredients would not be copied. Changes were made to ensure that these issues are correctly resolved.

To support multiple languages the data structures of all the model ingredients were changed, so that each name, comment and value (of attributes and quantity spaces) can be translated. The copy functionality was changed to work on these adapted data structures. Furthermore, the copying algorithm required several functional changes to deal with multiple languages.

Firstly, when a model ingredient (a definition of a type of model ingredients, a scenario or a model fragment) is copied to another model, the languages are now synchronized. This means that each of the active languages in the source model is activated in the target model (if not already activated). This assures that each of the model ingredients in the source model can be defined in the target model including all its translations.

Secondly, the active language of the target model is set to be the active language of the source model. This assures that the model ingredients are compared in the same language, which prevents redundant knowledge to be added.

Thirdly, when copying a model ingredient, the ingredient is first created using the current language (which is now the same in both models). Directly afterwards, the translations of this source model ingredient are added to this newly created target definition. The algorithm already made sure that the names in the current language did not clash with names in the target model, but now it also makes sure that the translated names in both models do not clash with each other.

8.3.2. Changes to OWL export/import

The OWL export/import functionality [7] exports qualitative models to the Web Ontology Language, which is a description-logic based knowledge representation language based on RDF(S) and XML. These models can be uploaded to the qualitative reasoning model repository[7]. This format allows the repository to read the content of the model and use the semantics to generate a representation of the content of the model. This representation is used to provide search support. Other modellers can download the models in the OWL format and import them again to extend the model, or reuse parts of it.

The addition of multiple language support did not break the OWL export/import functionality, but only the current language was exported/imported. The functionality was extended so that each of the languages is exported and imported. Each of the translations of strings of model ingredients is added to the OWL file using the *xml:lang*

[7] http://hcs.science.uva.nl/QRM/models/repository/

construct (see Figure 8.3). The languages were converted to two letter ids as specified in ISO 639-2[8]. During import these two letter ids are translated back into the full names of the languages.

```
<owl:Class rdf:about="&qrm;owl_ae_Human_action">
  <rdfs:comment xml:lang="en"></rdfs:comment>
  <rdfs:comment xml:lang="nl"></rdfs:comment>
  <rdfs:label xml:lang="en">Human action</rdfs:label>
  <rdfs:label xml:lang="nl">Menselijke actie</rdfs:label>
  <rdfs:subClassOf rdf:resource="&qrm;owl_ae_Action"/>
</owl:Class>
```

Figure 8.3: The OWL definition of the Human Action entity in English and Dutch.

[8] http://www.loc.gov/standards/iso639-2/php/code_list.php

9. Conclusion

This document has presented a list of frequently asked questions related to the use of Garp3 for building qualitative models, as well as numerous improvements to the Garp3 workbench. Domain experts had regular virtual meetings with Garp3 modelling experts to discuss their progress in the model-building process, supported by electronic communication tools, such as Skype, VNC, and FlashMeeting. Data gathered from these sessions, in the form of chat logs, models, intermediate model representations, and video registrations, has been analyzed to determine the kinds of questions that arise when using Garp3 to build, run, and inspect qualitative models. A list of such questions (and answers from experts) has been compiled and organized based on the topics they address, and is now available online for future users as a resource for learning about modelling with Garp3.

In addition, these sessions, and a specific user evaluation session have provided useful information about possible improvements for Garp3. A whole range of improvements has been implemented, varying from improvements in the Garp3 reasoning engine, and support for multiple models, to different aspects of the graphical user interface, such as support for multiple languages, improved tool-tip functionality, and improvements to the Simulate and Sketch environments. Together, these efforts are expected to further increase the usability of Garp3 and support domain experts in using it to build qualitative simulation models.

References

1. K. D. Forbus, 1984, Qualitative Process Theory, *Artificial Intelligence*, vol 24, p.85-168.
2. J. de Kleer, J. S. Brown, 1984, a Qualitative Physics Based on Confluences, *Artificial Intelligence*, vol 24, p.7-83.
3. Bredeweg, B., Bouwer, A., Jellema, J., Bertels, D., Linnebank, F. and Liem, J. Garp3 - A new Workbench for Qualitative Reasoning and Modelling. 20th International Workshop on Qualitative Reasoning (QR'06), C. Bailey-Kellogg and B. Kuipers (eds), pages 21-28, Hanover, New Hampshire, USA, 10-12 July 2006.
4. Liem, J., Bouwer, A., Bredeweg, B., Salles, P. and Bakker, B. 2007. 3rd NNR user group workshop on Qualitative Reasoning and Modelling, Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006), Project no. 004074, Project Deliverable Milestone M7.2.3.
5. Liem, J., A. Bouwer, F. Linnebank, and B. Bredeweg, 2007. Intelligent Help System, Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006), Project no. 004074, Project Deliverable Report D4.4.
6. Bredeweg, B., Bouwer, A., and Liem, J. 2006. Single-user QR model building and simulation workbench , Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006), Project no. 004074, Project Deliverable Report D4.1.
7. Liem, J., A. Bouwer, and B. Bredeweg, 2006. Collaborative QR model building and simulation workbench, Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006), Project no. 004074, Project Deliverable Report D4.3.