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**

D6.9.1
Curriculum for learning about QR modelling

Due date of deliverable: 01/01/2006
Actual submission date: 21/02/2006

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

Organisation name of lead contractor for this deliverable:

University of Amsterdam[1]

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 Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

[1] AUTHORS: Bert Bredeweg, Jochem Liem, Anders Bouwer and Paulo Salles

Abstract

This document is part of the NaturNet-Redime project, and presents curriculum materials for practitioners to develop their expertise in Qualitative Reasoning and Modelling (QRM). The following material is included:

- Theory (Chapter 2)
- Brief overview of the QRM workbench (Garp3) (Chapter 3)
- Model examples (Chapter 4)
- Brief section on QR-based modelling in Ecology (Chapter 5)
- Explanation of typical QRM vocabulary (Appendix A)
- Assignments (Appendix B)

This document (D6.9.1) is part of a set of deliverables that together highlight different aspects of Qualitative Reasoning and Modelling:

- D4.1 – Single-user QR model building and simulation workbench (software): refers to the software that is available for capturing and simulating qualitative models.
- D4.2.1 – User-manual for single-user version of QR workbench (document): is the user-manual that explains how to use the software.
- D6.9 – Curriculum for learning about QR modelling (*this* document): presents a curriculum that modellers can follow in order to learn about essentials of Qualitative Reasoning and Modelling, particularly focussing on the technical details required to actually build qualitative models.
- D6.1 – Framework for conceptual QR description of case studies (document): presents a structured methodology on how to capture qualitative knowledge, particularly focussing on the trajectory of developing a detailed model from a general idea.

Readers interested in seriously developing their QRM expertise are advised to obtain all these documents and use them as needed. There is also a mailing list to which modellers may subscribe (see: http://hcs.science.uva.nl/QRM/) and e.g. get answers to FAQ.

Acknowledgement

Tim Nuttle (University of Jena) and Peter Barz (Environmental Network Limited) provided many helpful comments on the first version of this deliverable

Document history

Version	Status	Date	Author
01	First draft (a largely complete version of the document).	29/12/2005	Bredeweg
02	Final version (comments addressed made by Peter Barz (received Feb 9th), Tim Nuttle (received Jan 11th), and Paulo Salles (received Feb 7th)).	21/02/2006	Bredeweg

Contents

1 Introduction

Conceptual knowledge is important for understanding the behaviour of systems. This conceptual knowledge is often qualitative and fuzzy, expressed verbally and diagrammatically. Qualitative Reasoning is an area of Artificial Intelligence that provides means to formally represent and automate reasoning with that kind of knowledge. Qualitative Reasoning does not use nor require numerical data.

Part of the NaturNet-Redime project focuses on learning through modelling and simulation. People learn about the behaviour of systems best when they can construct mental models of how the system works. The goal of the NaturNet-Redime project is to develop tools, particularly a workbench based on Qualitative Reasoning technology that can be used by stakeholders (novices and experts alike) to help them articulate and understand sustainable development issues, and by doing so improve their decision-making potential.

In addition to this workbench, the knowledge developed by the Artificial Intelligence community on Qualitative Reasoning needs to be made available to other audiences, audiences with possibly no knowledge of Artificial Intelligence. This document is one the first attempts in this direction. It presents typical aspects of Qualitative Reasoning using different perspectives. The prime target audience of this document is the NaturNet-Redime partners who will use the workbench within the project.

1.1 Guide to the Reader

Readers are not required to read through the whole document in a linear way, on the contrary, they should browse, select and read parts that fit their knowledge needs. In order to make this possible the information below will be helpful.

The material is organized as follows:

- Chapter 2 discusses the key ideas and principles behind Qualitative Reasoning. In addition to explaining issues, references are included to original publications. This provides readers with the option to further explore ideas when desired.
- Chapter 3 illustrates the Qualitative Reasoning ideas using the Garp3 workbench.
- Chapter 4 presents a set of models to explore and study Qualitative Reasoning issues in further detail. The models are presented in a sequence of increasing complexity.
- Chapter 5 presents a review of related research. That is, scientific results using ideas closely related or equal to Qualitative Reasoning. It turns out that most of this work originates from the field of ecology.
- Appendix A presents the Qualitative Reasoning vocabulary. It can be used as a reference to read and learn about the meaning of the ingredients that are part of this vocabulary.
- Appendix B mirrors chapter 4. It presents a series of assignments by means of which readers can acquire hands-on experience in constructing and simulating qualitative models. Appendix B addresses the same models as presented in Chapter 4. The latter can thus be seen as possible solutions to the assignments. The assignments are also presented in a sequence of increasing complexity.

In what order should the above material be processed?

Readers who have some knowledge of Qualitative Reasoning will probably find their own optimal route through the material presented in this document. For readers with

less experience, the following order is advised:

- *Step 1 – Getting started with the theory*
  It is probably best to start by reading and studying Chapter 2 and 3 first.
- *Step 2 – Doing a simple assignment*
  After reading Chapter 2 and 3, work through the 'Tree & Shade' assignment presented in Appendix B.1 and compare your solutions to those presented in Section 4.1.
- *Step 1 and 2 – Alternative approach*
  Some people like to acquire some hands-on experience before they want to study the theoretical ideas. This is also possible, namely by first carrying out the 'Tree & Shade' assignment and after that going through Chapter 2 and 3. The 'Tree & Shade' assignment is organized such that novices should be able to follow the step-by-step approach to building the model also.
- *Step 3 – Process the theory again*
  If you took the approach to first study Chapter 2 and 3 and then work through the assignment in Appendix B.1, you may want to read Chapter 2 for a second time, in order to better understand the ideas involved.
- *Step 4 – Doing all the assignments*
  After sufficiently understanding Chapter 2 and 3, and having acquired some hands on experience with the assignment in Appendix B.1, work through *all* the assignments in Appendix B (in the order as presented) and after each assignment study the solution in Chapter 4.
- *Additionally*
  - Appendix A should be used as a reference and consulted whenever needed.
  - When all the assignments have been solved, it is probably wise to digest Chapter 2 again.
  - After the reader has developed sufficient understanding of Qualitative Reasoning s/he may be interesting to read Chapter 5 and learn about 'related research'.

2 Qualitative Reasoning – Background and Essentials

Qualitative Reasoning (QR) is an area of research within Artificial Intelligence (AI) that deals with *conceptual* knowledge. It is an innovative technique that involves non-numerical description of systems and their behaviour, preserving the important behavioural properties and qualitative distinctions. QR technology is of great importance for developing, strengthening and further improving education and training on topics dealing with systems and their behaviours. It is well known that an essential part of modern education and training involves the comprehension of systems and their behaviours. That is, being able to distinguish a system from the environment in which it operates, to identify the parts that it is made of, and to predict and explain its behaviours. Cognitive science research has shown that when learners develop a causal model of the system's behaviour that they are better able to apply their knowledge to new situations. QR models are a way to develop such causal models, because they capture the fundamental aspects of a system or mechanism, while suppressing much of the irrelevant detail. An important advantage of QR over other (traditional) AI techniques, such as expert or knowledge-based systems, is that QR transfers not just predictions based on expert knowledge, but also makes this knowledge explicit, allowing its transfer to and reuse by others. This is an important prerequisite for educational software, such as interactive learning environments.

QR technology has proven to be cost effective, reliable and efficient, as a means to analyse the behaviour of systems without numerical information. The behavioural aspect studied most is qualitative prediction of behaviour, i.e. analysing how the behaviour of a system evolves as time passes. Although any system can be an object of such a reasoning process, traditionally the majority of research deals with physics and engineering (Weld and de Kleer, 1990). Successful application areas include autonomous spacecraft support (Williams et al., 2003), failure analysis and on-board diagnosis of vehicle systems (Price and Struss, 2003), automated generation of control software for photocopiers (Fromherz et al., 2003), and intelligent aid for learning about thermodynamic cycles (Forbus et al., 1999). Thus, QR is relevant for researchers that are interested in important AI issues as well as for stakeholders such as managers, engineers, developers, and citizens who are looking for potential (industrial) benefits of AI.

2.1 Why use Qualitative Reasoning?

Sustainable development requires understanding of the structure and functioning of nature. The structure consists of objects, such as individuals, populations, communities, and their relations with the physical world organized in ecosystems and landscapes. Functioning is explained by imposing causal relationships on observable features of those objects in a way that it is possible to understand why things happen. However, sustainable development has features that put strong barriers on research and knowledge representation, including: the complexity of any system, and the difficulty to obtain long-term good quality data and to run controlled experiments. Hence, knowledge on sustainable development is heterogeneous, including both quantitative and qualitative aspects. There is need for new and efficient computer-based tools to adequately capture that knowledge.

Models and simulations are important tools for research. Scientists often frame their ideas in modelling expressions and test them through simulations. A distinction can be made between statistical models and structural models (Bossel, 1986). Statistical models usually do not capture the available structural knowledge and their parameters

usually have no counterpart in the real system. Structural models, on the other hand, are tools for describing system structure and system elements as close as possible to real systems. This way, such models mimic real objects, structural connections, parameter values and may provide behavioural predictions grounded in the system structure.

QR models are structural models that are particularly adequate to support the understanding of the behaviour of systems. The following QR features are especially attractive for modelling knowledge on sustainable development:

- Approaches to QR provide a rich vocabulary for describing objects, situations, relations, causality, assumptions, and mechanisms of change. Using this vocabulary it is possible to capture *conceptual* knowledge about systems and their behaviour and use that knowledge to automatically derive relevant conclusions without requiring numerical data.

- QR modelling uses a compositional approach to enable reusability. This is achieved by constructing libraries of partial behaviour descriptions (model fragments) that apply to the smallest entities relevant within a domain. As larger systems are built from these basic elements, reasoning about the behaviour of larger systems means combining the behaviour of these elements. This prevents having to develop dedicated models for each system encountered.

- QR models provide causal explanations of system behaviour. As causal relations are explicitly represented in model fragments, it is possible to derive the behaviour of a complete system from the behaviour of its constituents and to automatically generate insightful explanations that *causally* explain the functioning of the overall systems in terms of its constituents.

- QR creates representations for continuous aspects of the world to support reasoning with little information, including incomplete knowledge or knowledge expressed just in qualitative (linguistic) terms (without any numerical information).

Qualitative models automate *conceptual* knowledge. Being explicitly represented this knowledge can be inspected, possibly modified, by users and by other modellers. The construction of such *qualitative* models is of particular interest for education, training, management, and decision-making, because they facilitate structured expression and communication of insights among participants. After all, many questions of interest can be answered in terms of ‘better or worse’, ‘more or less’, ‘sooner or later’, etc. (cf. Rykiel, 1989).

2.2 What is Qualitative Reasoning?

Early work on QR focuses on automatic generation of explanations (Brown et al., 1982; Hollan et al., 1984) in the context of interactive learning environments, that is, educational software that fosters learning by having learners interact with a simulation of the subject matter. Key QR publications present approaches to having *computers* perform conceptual analysis of system behaviour (Bobrow, 1984). From this work originates the idea of using *qualitative* models and simulations, also referred to as *articulate* simulations (Forbus, 1988; Bredeweg and Winkels, 1998). A typical QR model captures a representation of both the structural and the behavioural aspects of a system. A qualitative model abstracts from quantitative information by using an ordered set of qualitative values, usually a set of alternating points and intervals referred to as a

quantity space. *Quantities* are assigned values from such quantity spaces, allowing the capture of qualitative distinct behavioural features of a system. Changing behaviour is represented using a qualitative derivative for each quantity, representing decreasing ∂=[–], steady ∂=[0], and increasing ∂=[+].

Another typical aspect of a QR model is the explicit representation of *causality*. Different types of *modelling primitives* have been introduced in this respect, each type having a specific conceptual meaning and a formal defined calculus allowing implementation in computer programs. Following these basic ideas a wide range of topics have been tackled. To name a few: order of magnitude reasoning (Raiman, 1986), alternative approaches to inferring causality (Iwasaki and Simon, 1986; de Kleer and Brown, 1986), reasoning with multiple models (Addanki et al., 1991; Weld, 1988), compositional modelling using assumptions (Falkenhainer and Forbus, 1991), integration with numerical simulation (Amador et al., 1993), and varying granularity in the representation of time (Rickel and Porter, 1997).

Using the QR representational primitives, libraries of model fragments can be constructed that capture knowledge from a certain domain. QR engines use these libraries to automatically generate qualitative models of systems belonging to such a domain. Building a library is thus a fundamental aspect of using QR technology. In the past, considerable effort has been put in building qualitative models for the domain of physics (e.g. Collins and Forbus, 1989; Kim, 1993). Libraries for other domains still need to be developed and made to use. Lately, libraries capturing ecological knowledge are being created (Salles and Bredeweg, 2003).

2.2.1 A Working Example

Let us consider a simple two-tank system, with tanks of equal width, for which it is known that both tanks contain a certain amount of oil and that the oil-column is higher on the left-hand side (LHS). Let us assume that the relative heights of the two tanks are unknown. Now suppose that the two tanks are connected via a pipe with a valve, placed at the bottom of the containers. When the valve closing this pipe is opened, what behaviours may happen? Figure 2.1 illustrates the answer to this question.

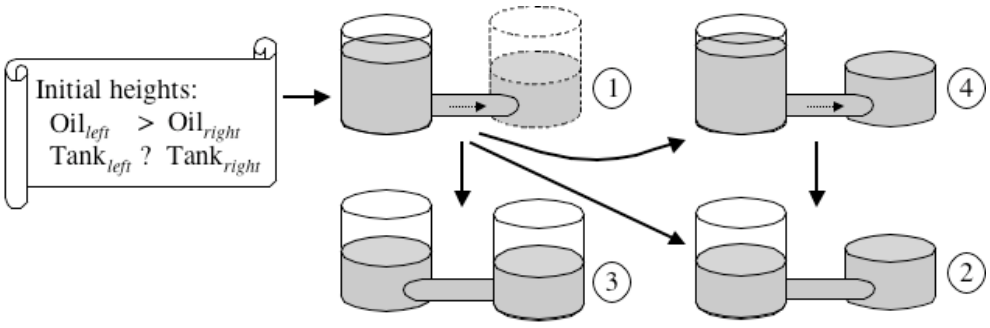


Figure 2.1: Possible behaviours of a two-tank system

The oil-column on the LHS is higher than on the right-hand side (RHS). Hence, oil will flow from the LHS into the RHS tank until the pressure-difference becomes zero and the system reaches equilibrium. Since the initial description does not specify the relative heights of the tanks (as visualised by the dashed line in situation 1) multiple behaviours are possible. There are three qualitatively distinct possibilities. If the tank on the RHS is high enough it will be able to contain all the inflowing oil (situation 3). Alternatively, the RHS tank may at the start already be lower than the LHS oil-column. In this case, oil will be spilled (situation 4) until the height of the decreasing LHS oil-column becomes equal

to the height of the smaller RHS tank (situation 2). Finally, it may be the case that the RHS tank is smaller but still high enough to contain all the inflowing oil. The system stabilises at the moment the RHS tank becomes fully filled (situation 2).

Notice that different behaviours would be predicted when more (or less) information is initially known. For instance, knowing the relative tank heights would result in predicting less possible behaviours. Humans are flexible in this respect. They apply the same basic knowledge to different situations producing appropriate conceptual analysis. This is also one of the features of QR and rather different from traditional approaches using numerical simulations. Instead of having a single fixed model, a QR engine automatically assembles a unique model to fit a particular situation. The sections below discuss this idea in more detail as well as other prominent features of QR. Together they show how conceptual behaviour analysis can be formalised and reasoned with automatically by computers using QR technology.

2.3 World-view: Ontological Distinctions

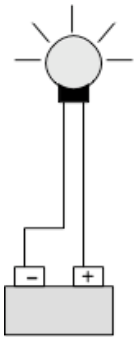
QR provides explicit representations of the conceptual modelling layer, rather than only an executable mathematical expression. This is crucial to any attempt to support and automate model building and represents one of the major issues of QR. This section discusses the two main ideas that have been developed in this respect, as well as an alternative approach.

2.3.1 Component-based Approach

De Kleer and Brown (1984) describe a component-based approach to Qualitative Reasoning. In their approach the world is modelled as *components* that manipulate *materials* and *conduits* that transport materials. Physical behaviour is realised by *how* materials such as water, air and electrons, are manipulated by, and transported between, components. How components manipulate materials is described in a library of component models. In these descriptions a component is associated with qualitative equations known as confluences: relations between variables that describe the characteristics of the materials. The model of a certain component may consist of a number of qualitative-states, each specifying a particular state of behaviour. More specific, a qualitative state consists of a name, one or more specifications and a set of confluences. The specifications define the conditions that must be true for the qualitative state to be applicable. The confluences describe the specific behaviour of the materials in this state of behaviour. Figure 2.2 illustrates the basic idea. It shows a device consisting of two components, a battery and a lamp. The battery has three qualitative states of behaviour: fully charged, partially charged, and empty. The lamp has two qualitative states: it can function normally (OK) or it is broken. The behaviour of the device as a whole is generated using the cross-product. That is, all the possible behaviours (qualitative states) of each component are combined with all the possible behaviours of all other components (see Table in Figure 2.2).

Generating the cross-product and determining the consistency of each potential state of behaviour is referred to by de Kleer and Brown as the *intrastate* analysis. After this analysis, the problem is to find out which states of behaviour will be successors as time passes by. This is referred to as the *interstate* analysis, which tries to determine whether the behaviour within a certain state may lead to the termination of that state. In other words, to find out if the values of variables are changing such that they, when time passes by, no longer fall within the specifications of the overall state of behaviour. In the component-based approach this is realised by applying rules that must hold between states. An example of such a rule is the *limit rule*: if in the current state a variable has a

value and increases or decreases, then it will respectively have the adjacent higher value, or the adjacent lower value, in the next state. For instance, the overall system behaviour ‘soft light’ may move into the behaviour of ‘no light’ when the battery power decreases to zero (and thus moves from qualitative state ‘partially charged’ to ‘empty’). Another important rule is the *continuity rule*: each variable value must change continuously over states. For instance, the battery cannot immediately change from ‘fully charged’ to ‘empty’.



		Battery		
		Fully charged	*Partially charged*	*Empty*
Lamp	*OK*	Bright light	Soft light	No light
	Broken	No light	No light	No light

Figure 2.2: Possible behaviours of a lamp connected to a battery

2.3.2 Process-based Approach

Forbus (1984) describes a process-based approach to Qualitative Reasoning, in which the world is modelled as consisting of physical *objects* whose properties are described by *quantities*. Physical behaviour refers to these objects being created, destroyed, and changed. Although in principle anything can be represented as an object, there is a commitment in Qualitative Process Theory (QPT) to represent physical objects as closely as possible to how humans perceive the physical world. Two important primitives in the process-based approach are *individual views* and *processes*. Views describe the characteristics of an object or a group of objects, e.g. of a container containing a liquid. Processes describe mechanisms of change. These changes are represented by *influences*. They describe the changes that occur when a process is active. Typical examples of processes are heat-flow and liquid-flow. The former describes energy exchange between objects with unequal temperatures. The latter describes how liquid flows between connected containers with unequal pressures. Figure 2.3 depicts the behaviour of a boiler system. It illustrates the basic idea of how QPT uses *processes* and the idea of *limit-analysis* as the basis for behaviour prediction.

The boiler system consists of a heater and a container. The container contains water and is being heated. What behaviours may occur and which processes cause them?

1. After the heater is turned on, a heat-flow process causes energy to flow from the heater to the container and the water. This causes the water temperature, the container temperature, and the internal container pressure to increase. This behaviour may lead to three other behaviours (2, 3 or 4), due to limits being reached.

2. The boiler explodes because the internal pressure becomes too high. The reaction force generated by the container is lower than the pressure exerted by the substance it contains. Once the boiler system is broken the simulation is terminated.

3. The temperature of the water reaches its boiling point. A new process 'boiling' becomes active which causes the generation of steam. This behaviour may lead to three other behaviours (2, 4 or 5), due to limits being reached.

4. The temperature of the substance in the container (be it water or steam) is now equal to the temperature of the heat source. From here on, no further changes can take place.

5. All the water has now turned into steam. The boiling process has stopped, but the heat-flow continues. This behaviour may lead to three other behaviours (2, 4 or 6), due to limits being reached.

6. If the heater is warm enough it may ultimately cause the container to melt, because the container temperature will reach its melting point. The boiler system is broken after this behaviour. Hence the simulation stops here.

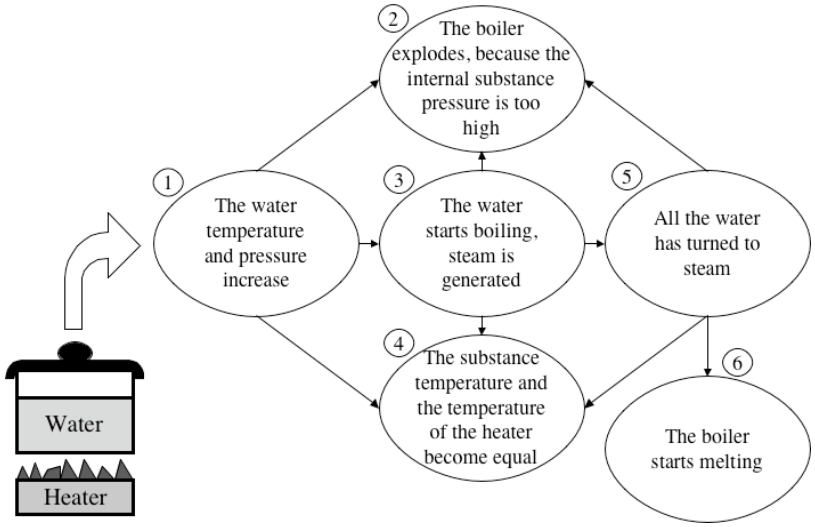


Figure 2.3: Possible behaviours of a boiler system

Notice that, despite many differences (see Bredeweg, 1992) the global idea of using a library of model fragments, albeit consisting of views and processes, is similar to the idea of using a library of component models in the component-based approach.

2.3.3 Constraint-based Approach

Kuipers (1986, 1994) describes the constraint-based approach. This approach takes a qualitative version of a differential equation as its starting-point. The basic assumption is that Ordinary Differential Equations (ODE's) can be rewritten into Qualitative Differential Equations (QDE's) which can be used for qualitative simulation. In the constraint-based approach there is no explicit representation of entities from the (physical) world. Furthermore, this approach does not use a library of any kind from which models can be assembled during simulation. Instead, the qualitative reasoning engine is provided with a description of some aspect of the (physical) world in terms of the qualitative constraints between variables as shown in Table 2.1. Notice that each qualitative constraint maps onto a specific aspect of the ordinary differential equations.

Behaviour prediction with constraint models is done by applying a kind of generate and test cycle that produces the possible behaviours of a system. The generation part determines how a state of behaviour may change into a new state of behaviour, by

applying *transition* rules to each function in the current state of behaviour. Testing is concerned with determining the consistency of a certain state, by applying constraint satisfaction to the constraint model that represents the behaviour in that state.

Table 2.1: Qualitative constraints and mathematical functions

Qualitative constraints (QDE's)	*Mathematical functions (ODE's)*
ADD(*f*, *g* ,*h*)	$f(t) + g(t) = h(t)$
MULT(*f*, *g* ,*h*)	$f(t) \cdot g(t) = h(t)$
MINUS(*f*, *g*)	$f(t) = -g(t)$
DERIV(*f*, *g*)	$f'(t) = g(t)$
M+(*f*, *g*)	$f(t) = H(g(t)) \wedge H'(x) > 0$
M–(*f*, *g*)	$f(t) = H(g(t)) \wedge H'(x) < 0$

2.3.4 Suitability of Approaches

Although the constraint-based approach is probably the most common approach it has drawbacks, the foremost being that it does not support deriving behaviour from the physical structure (see also next section). Instead, it takes differential equations as a starting point. In a way, the constraint-based approach has the same drawbacks as traditional numerical approaches. The modelling primitives provided by this approach do not allow *symbolic* modelling of the conceptual knowledge that domain experts have. Notions such as processes, static properties, causality, and physical structure cannot be represented by this approach explicitly. The component-based approach does facilitate the representation of much of this kind of knowledge. However, the ontology of interconnecting components seems more suitable for human made artefacts than for natural systems. From an ontological perspective, QPT is probably most suitable for building models about sustainable development (Salles, 1997).

2.4 Inferring Behaviour from Structure

In general, a qualitative reasoning engine takes a *scenario* as input and produces a *state-graph* capturing the qualitatively distinct states a system may manifest (Figure 2.4). A scenario usually includes a structural description of the physical appearance of the system. Such a description models the entities (e.g. physical objects and components) that the system consists of, together with statements concerning the structural organisation of these objects (e.g. a container *containing* liquid). Often a scenario also includes statements about behavioural aspects such as relevant quantities and in/equality statements between some of those quantities.

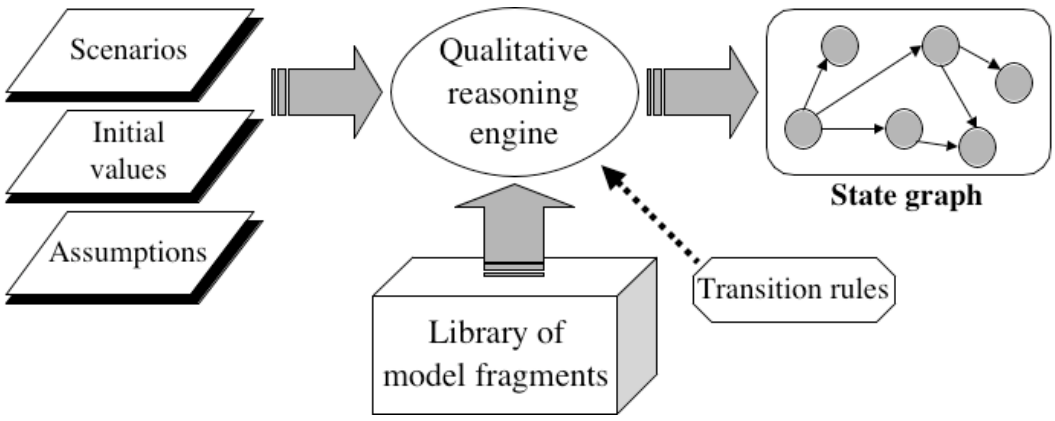


Figure 2.4: Basic architecture of a qualitative reasoning engine

A state-graph consists of a set of states and state-transitions. A state refers to a qualitatively unique behaviour that the system may display (e.g. a possible state of behaviour). Similar to a scenario, a state consists of a set of declarative statements that describe the physical structure of the system and the behaviour it manifests at that moment. A state is typically characterised by a set of qualitative values of relevant quantities representing their magnitude and direction of change. A state-transition specifies how one state may change into another state. A sequence of states, connected by state-transitions, is called a behaviour-path, but is also referred to as a behaviour trajectory of the system. A state-graph usually captures a *set* of possible behaviours-paths, because multiple state-transitions are possible from certain states. To further detail these notions, consider again the two-tank system from Figure 2.1. The simulation results obtained from a qualitative model of this system are shown in Figure 2.5. The state-graph (LHS) shows the four possible states that the two-tank system may manifest. Each black circle refers to a possible behavioural state, the state numbers refer to identifiers created by the reasoning engine[2], and the arrows indicate which states may succeed each other. Thus, the conditions set in the scenario (referred to as 'input' in Figure 2.5) lead to state 1. This means that (with the knowledge the engine has) there is a unique interpretation of the scenario. From state 1 three successors are possible: 4, 2, and 3, if state 4 occurs it is always followed by the behaviour represented by state 2; States 2 and 3 have no successors, they are end-states, and represent new equilibria.

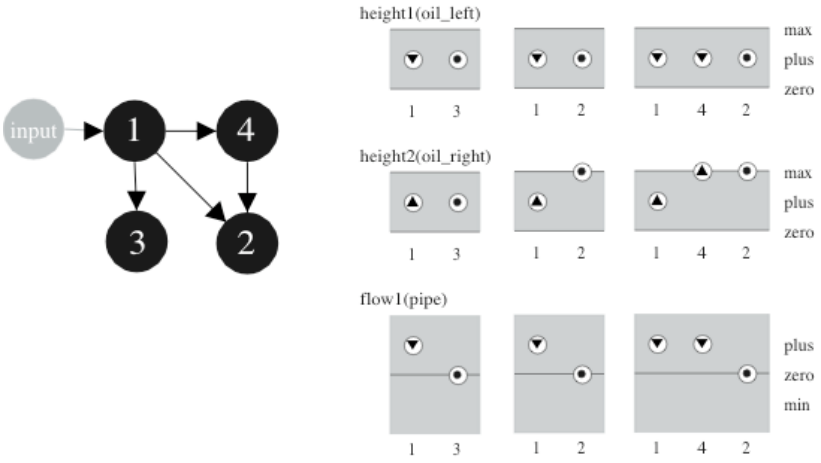


Figure 2.5: Simulation results of a model for the two-tank system

Notice that a QR engine generates all possible solutions. That is, given a scenario (a structural description) it will generate *all* behaviours that are consistent and thus possible to infer, with the details defined in that scenario. This is rather different from a numerical simulation that usually produces one specific answer. One of the interesting features of QR is the awareness it creates for all possible interpretations of a certain situation. This can for instance be useful to support management tasks. The results obtained by the simulation show all that may happen. If certain behaviours are not acceptable, preventive actions can be taken.

2.5 Qualitativeness and Representing Time

Qualitative prediction of behaviour is concerned with reasoning about the properties of the physical world that change over time. Particularly, to include only those qualitative distinctions in a behaviour model that are essential for solving a particular task for a

[2] Notice that the state numbers are *identifiers* created by the reasoning engine, they do not necessary reflect the order in which states of behaviours occur.

certain system. The goal is to obtain a finite representation that leads to coarse, intuitive representations of systems and their behaviour. Thus, central to qualitative reasoning is the way in which a system is described during *a period of time in which the qualitative behaviour of the system does not change*. The notion of change is subtle, because numerical values of variables may change whereas from a qualitative point of view the behaviour of the system remains constant. During a heat-flow process, for example, the temperature of a liquid may increase, but from a qualitative point of view it is still a liquid, until another process (boiling) becomes active and the liquid becomes a gas.

In QR the representation of time is closely intertwined with the representation of quantity values. Changes in the values of quantities represent time passing. The possible qualitative values of a quantity may be divided into points and intervals. A quantity can therefore, during a certain period of time (of constant qualitative behaviour), have its value either at a point or at an interval. The intuitive understanding behind this approach is illustrated in Figure 2.6 for the quantity temperature as it is used to describe the physical state of a substance.

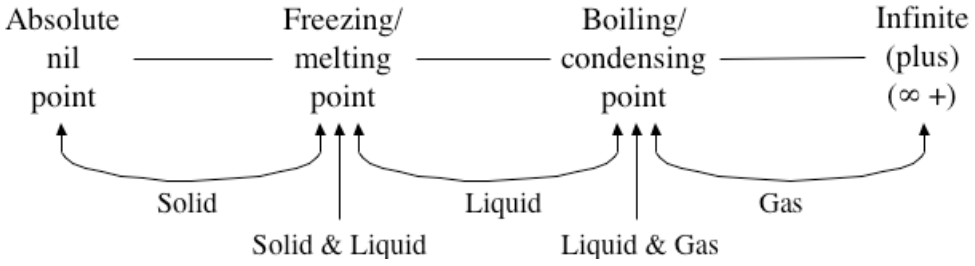


Figure 2.6: The quantity space for the temperature of a substance

All the quantitative values a substance temperature can have are divided into six qualitative values, consisting of three intervals and three points. Each value resembles a characteristic period of constant qualitative behaviour for the substance. If, for example, the temperature has a quantitative value somewhere between freezing point and boiling point and this value increases, then the substance shows constant qualitative behaviour, namely 'being a liquid', until it reaches its boiling point. As soon as it reaches this boiling point, the substance arrives at a new time interval in which it again shows constant qualitative behaviour, namely boiling.

In qualitative models this knowledge is formalized as follows. A quantity value is represented as the pair <Magnitude,Derivative>. Magnitude represents the amount of a quantity and the Derivative represents the direction of change over time. The values a Magnitude can take on are represented in a Quantity Space (QS). Consider again the two-tank system (Figure 2.1). The amount of substance in a tank can be represented as having three possible magnitudes: QS={zero,plus,max}, respectively meaning there is no substance, there is some substance, and the amount of substance in the container has its highest possible value: maximum. Values for the Derivative are also represented by a quantity space, namely QS={min,zero,plus}, meaning the Magnitude is decreasing, steady, and increasing. Thus, if amount has the value *amount=<plus,plus>* this can be read as: there is an amount and in the current state it is increasing.

A *value-history* diagram shows the qualitative values generated by a QR engine. Figure 2.5 (RHS) shows the value-history for the quantities[3] involved in the model of the two-tank system for all the behavioural states. For instance, in state 1 the *height* of the

[3] The model also includes *amount* and *bottom-pressure* for each tank, but these are not shown.

oil_right has magnitude *plus* and is *increasing*, hence: <*plus, plus*>. Next, in state 4 this quantity has the value *max* and is still *increasing:* <*max, plus*> (representing overflow). Finally, in state 2, it has again value *max*, but now it is steady: <*max, zero*>. This can be inferred from the diagram (Figure 2.5) as follows. The possible values that *height* can take on are shown on the RHS: QS={zero, plus, max}. The circles above the state numbers designate the specific value the quantity has in that state. In addition, the circles contain a small arrow pointing up, or down, or a small black circle. These indicate that the quantity is increasing, decreasing, or steady, respectively. A sequence of quantity values is referred to as a *value-history* and it follows a *behaviour-path*. In the case of the two-tank system there are 3 such paths: [1 → 3], [1 → 2], and [1 → 4 → 2].

Determining the relevant quantity space for each quantity is an important aspect of constructing a qualitative model because it is one of the features that determines the variety of possible behaviours that will be found by the engine when the model is simulated. Inequality statements (e.g. *height oil_left* > *height oil_right*) are also important in this respect. In fact, each qualitative distinct state of behaviour is defined by a unique set of values and inequality statements. Transitions between behavioural states are the result of changes in these values and inequality statements. State transitions are shown in a state-graph as arrows connecting the circles (Figure 2.5). For example, while going from state 1 to state 4, the magnitude of *height* (for *oil_right*) changes from *plus* to *max*. Going from state 4 to state 2 the oil heights in the two tanks become equal (not shown in Figure 2.5) and the *flow* becomes *zero*. In addition, the heights for both columns stop changing (∂=0).

2.6 Causality

Analyzing and explaining the behaviour of a system in terms of cause-effect relations is central to human reasoning and communication. When we think that ‘A causes B’, we believe that if we want B to happen we should bring about A, and if B happens, then A might be the reason for it. Causality can also be perceived as being indirect: ‘A causes C indirectly’ if ‘A causes B’ and ‘B causes C’. Formalizing the notion of causality and exploiting it in automated reasoning is the basis for explanation facilities in QR systems. QPT explicitly distinguishes between changes that are caused *directly* or *indirectly* (Forbus, 1984). Forbus refers to this as the *causal directness hypothesis*: changes in physical situations are caused by processes (*influences*, represented as {I+, I-}), or by propagation of those direct effects through functional dependencies (*proportionalities*, represented as {P+, P-}). This hypothesis puts three further constraints on how influences and proportionalities should be applied. Firstly, all changes are initialised by influences. Without an influence, or for that matter a process, there is no change and therefore no behaviour in the physical world. Proportionalities are used to propagate changes, introduced by influences, throughout the whole system. Secondly, both influences and proportionalities are *directed*, i.e. their effect propagates in one direction only. The influencing quantity has to be known before the dependent quantity can be determined. The relations may not be used the other way around, because this would violate the causal chain of changes, which is one of the essential features of QPT. Thirdly, no quantity may be influenced directly and indirectly simultaneously. According to Forbus, a physics that allows a quantity to be influenced both directly and indirectly at the same time must be considered inconsistent, because it also violates the essential, non-recursive, chain of causality.

Both direct influences and qualitative proportionalities are modelling primitives that express causal relationships between quantities, and have *mathematical* meaning. Direct influences determine the value of the derivative of the influenced quantity. For

example, the relation I+(Y,X) means that dY/dt = (... + X ...). By definition, the quantity X is a *rate* and its value should be added to Y. Qualitative proportionalities carry much less information than direct influences. For example, the relation P+(Y,X) means that there is some monotonic function (f) that determines Y, and Y is increasing in its dependence on X, such that Y = f(... X ...) and dY/dX >0. A quantity that is not influenced by any process is considered to be constant. Notice that a single direct or indirect influence statement does not determine, by itself, how the quantity it constrains will change. Its effect must be combined with all the active influences on that quantity. Ambiguities may arise when positive and negative influences are combined and their relative magnitudes are not fully known. In such cases, the reasoning engine either considers all the possible combinations or any explicitly represented assumption that may constrain the system's behaviour.

Figure 2.7 shows a subset of the dependencies that hold in state 1 of the simulation of the two-tank system model (Figure 2.1). Such a set of dependencies is often referred to as the causal model. The diagram shows that the two oil-columns have unequal *heights* and (bottom) *pressures*. The *flow* (rate) between the two tanks depends on the difference between those pressures (and is qualitatively proportional to it). The flow has a negative influence on the oil-column with the higher pressure and a positive influence on the other, decreasing and increasing the two amounts of oil respectively. Changes in the amounts propagate to changes in heights, which in turn change the pressures. Notice that this diagram also shows the quantity space for each quantity, the current value, and the direction of change. The latter is visualised by triangles pointing up (increasing), or down (decreasing), and by small black circles (steady), (as no quantity is steady in state 1, circles are not shown in Figure 2.7). The direction of change icon is placed adjacent to the current value of the quantity, highlighting the latter in the context of its quantity space. For instance, for the *oil_left* holds: *height=<plus,min>.*

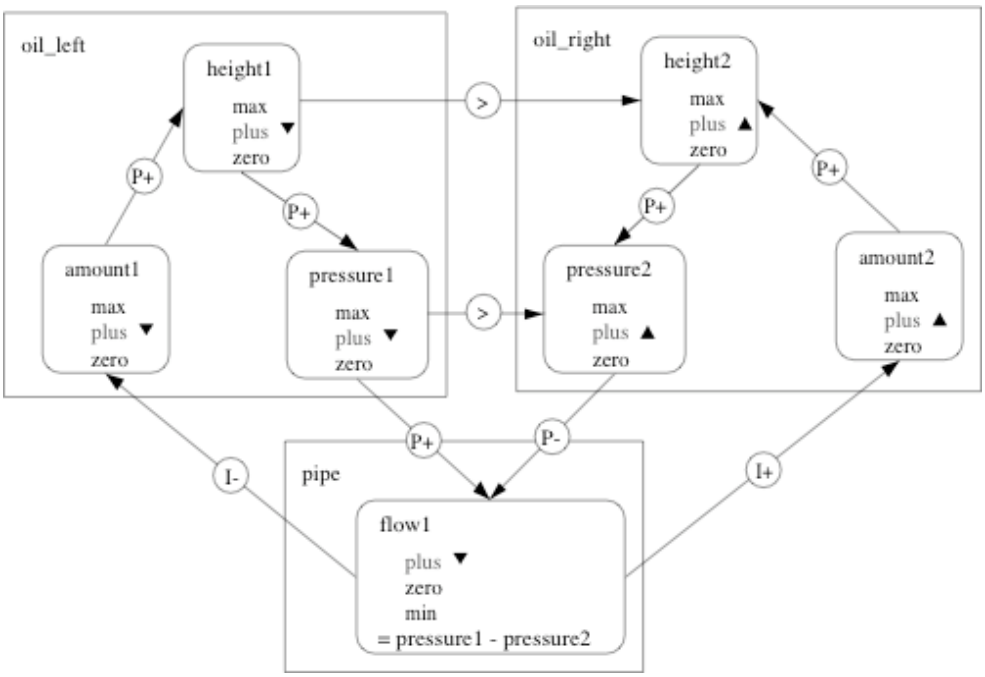


Figure 2.7: Causal dependencies for the two-tank system

It is relevant to mention that QR models represent changes in the causal structure that may happen during the simulation. For example, when the heights in the two tanks become equal (in Figure 2.5, states 2 and 3) the flow stops and the direct influences

resulting from the process no longer exist. Therefore, the causal model of the two-tank system in states 2 and 3 is different from the one shown in Figure 2.7 (because the inequality sign will be replaced by an equality sign, and the arrows representing I+ and I- will be deleted). Being able to change the causal model is an important feature of QR.

The notion of causality is complex and competing ideas exist about how to capture it in models, such as the notion of causal ordering (Iwasaki and Simon, 1986). However, the approach taken by QPT is generally seen as a principled one. It also seems most promising as a means to capture causal reasoning in the domain of sustainable development (Salles, 1997).

2.7 Model fragments and Compositional Modelling

Most QR systems aim at building libraries of elementary context independent model fragments (illustrating component behaviour, processes, etc.). This provides the basis for automating the model composition and for the reuse of models, a highly desirable feature both for theoretical development and for applications. Model fragments can thus be seen as re-usable conditional statements that capture knowledge about the phenomena existing in a certain domain. Model fragments applicable to a scenario are assembled by the engine and used to infer the behaviour of the system specified in that scenario (Figure 2.4). They are also used to infer the facts true in each of the successor states. This implies, among other things, that the set of facts may change and can be different for alternative states. In general, a model-fragment requires certain structural details to be true (e.g., a tank, a liquid and a contain relation between these two entities). If the required structure exists the model-fragment is instantiated for that structure and introduces the behaviour details that apply to it (e.g., the quantities amount, height, pressure and the dependencies that hold between them). A specific model-fragment can be instantiated multiple times, namely for each occurrence of the structure to which it applies.

Preferably, model fragments implement the ‘first principles’ (the fundamental laws) relevant to a domain, enhancing their usability across different systems. Reusability requires that model fragments represent behavioural features independent from the specific environment in which they operate. De Kleer and Brown (1984) discuss a set of modelling principles for realising this objective. One principle is the ‘no-function-in-structure’, which states that the model of a specific component may not presume the functioning of the device as a whole. For instance, the qualitative states of a lamp (Figure 2.2) may not specify ‘lit’ or ‘not lit’, because ‘being lit’ depends also on the battery (and not just on the lamp). A properly functioning lamp will still not produce any light when the battery is empty or not connected to the circuit. The no-function-in-structure principle is general and applies to any approach to QR that uses a library of model fragments. Given a sufficiently well developed library for a certain domain the qualitative reasoning engine can predict the behaviour of all kinds of systems belonging to that domain.

3 Garp3 – Workbench for QRM

The availability of software and tools to construct and simulate QR models is limited. QPE (Forbus, 1986) is a reasoning engine that implements QPT, using this package requires programming skills in LISP. QSIM (Kuipers, 1986) is the implementation of the constraint-based approach and can be downloaded on-line. Easy to use QR model-building learning environments have recently been developed, notably Betty's Brain (Biswas et al., 2001) and Vmodel (Forbus et al., 2001). These packages are used for teaching in middle schools and are optimized for that purpose. Although useful in classroom situations, essential features of QR are missing and hence these tools are limited in their potential to capture expert knowledge.

Recently Garp3 has been developed and implemented as part of the NaturNet-Redime project. Garp3 is based on previously develop software and provides a seamless workbench for building, simulating, and inspecting qualitative models. Garp3 is implemented in SWI-Prolog[4] and can be downloaded on-line[5]. To further discuss the use of QR for capturing knowledge on sustainable development, this section focuses on the workbench Garp3.

3.1 Workspaces in Garp3: Build

The Build environment in Garp3 provides nine workspaces for creating model ingredients, divided into two categories. Building blocks are used to define ingredients (types) that can be reused and assembled into constructs.

- Building blocks
  - *Entities*: represent physical objects or conceptualizations that are part of the system to be modelled. They form an important backbone to any model that is created. Entities are organized in a *subtype hierarchy*.
  - *Agents*: represent external influences enforced upon a system. They are thus exogenous to the system. For instance, the sun providing energy.
  - *Assumptions*: are labels that can be used to hide or show certain detail in a model. Typical examples are *operating* and *simplifying* assumptions. The notion of an open versus a closed population (migration or no migration) is an example of an operating assumption for models in ecology. It provides a certain perspective on the simulation. A simplifying assumption typically reduces the simulation complexity. For example, to consider a particular quantity value constant.
  - *Attributes*: define properties of entities that do not change (static). An example could be to represent the *colour* of an animal's fur as having value *brown*.
  - *Configurations*: are commonly called 'structural relations'. Structural relations model how entities are physically (or structurally) related to each other. For example: representing that a certain species is *part of* an ecological niche.
  - *Quantities*: represent changeable properties of entities and are typically seen as implementing the behavioural characteristics of a system.
  - *Quantity Spaces*: represent the values that quantities can take on.
- Constructs
  - *Scenarios*: describes the initial situation of the system whose behaviour is to be captured by the qualitative model. A scenario is the starting point for

[4] http://www.swi-prolog.org/
[5] http://hcs.science.uva.nl/QRM/

running a simulation and is created by defining (and relating) instances of building blocks. In/equality statements can also be added.

- *Model fragments*: define behavioural features for one or more entities. Model fragments are assembled form building blocks, have conditions (specifying their applicability) and consequences (new knowledge that is true when the fragment applies), and are organized in a subtype hierarchy. Different types of model fragments exist, notably *static*, *agent*, and *process*. An important aspect of a model-fragment is the specification of causal knowledge in terms of influences (only in processes and agents), proportionalities, and correspondences. In/equality statements are also defined in fragments.

3.2 Building a Population Model

As example consider the behaviour of a population consisting of frogs, that is only determined by 'natural' mortality and natality. Figure 3.8 shows some of the workspaces and model ingredients that may be defined for such a model.

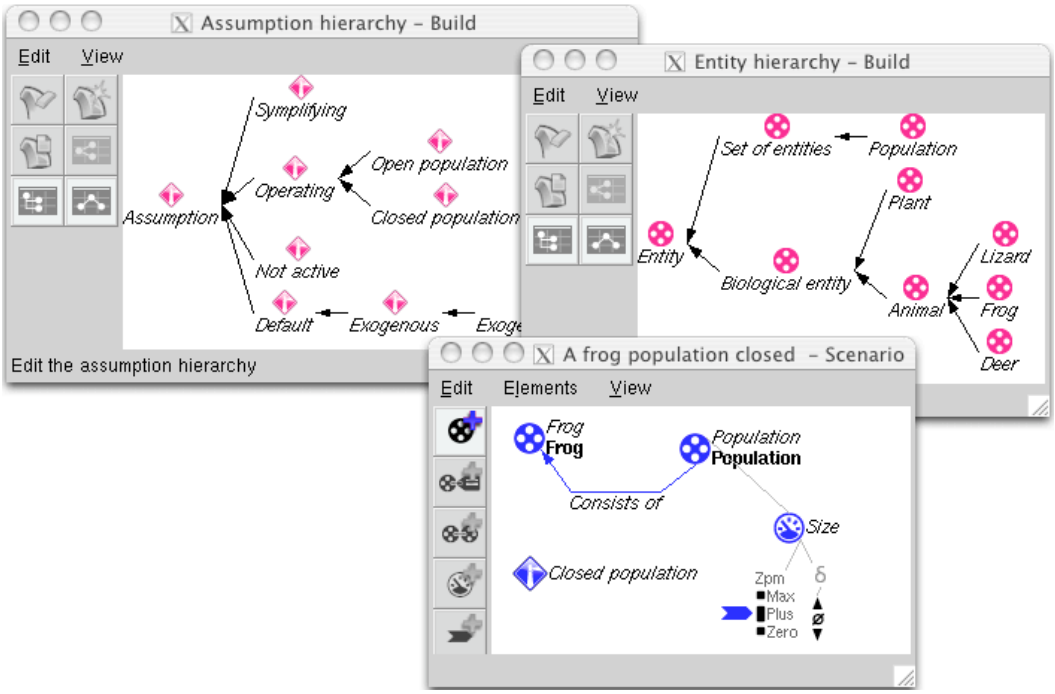


Figure 3.8: Assumption and entity hierarchy, and a scenario of a frog population model

The assumption hierarchy is shown on the LHS. Among others it defines an operating assumption for *open* and *closed* populations. The entity hierarchy is shown on the RHS. It is kept simple for reasons of clarity. A distinction is made between *Biological entity* and *Set of entities*. According to the model there are two kinds of biological entities (*Animal* and *Plant*), and there are three kinds of animals. Notice that the representation of entities follows an inheritance structure. For example, all facts that can be inferred for *Animal* also apply to *Frog*. A scenario is shown in the middle of Figure 3.8. It specifies the existence of a *Population* named *Population*[6]. This population consists of *Frog* named *Frog* and has the quantity *Size*. *Size* can take on five values QS={Zero,Low,Normal,High,Max}, and currently has the value *Normal*. The derivative of

[6] The fact that labels such as 'Population' appear twice has a meaning. The italic version refers to the *type* as specified in the entity hierarchy, where as the bold version refers to the *instance* name given in a specific situation. The user (the creator of the model) must provide the instance name. When omitted, the software inserts the default name similar to the type.

Size (shown by two arrows and zero) is unknown (no value is pointed out) and there is an assumption named *Closed population* (identified by a question mark).

In order for an engine to infer behaviour it needs a library of model fragments capturing general knowledge about population dynamics. A part of this is shown in Figure 3.9. The domain theory implemented here is a qualitative reading of the equation:

Size(t+1) = Size(t) + (Born + Immigrated) – (Dead + Emigrated)

Size stands for the number of individuals of a population. *Born*, *Dead*, *Immigrated*, and *Emigrated* refer to the amount of individual being added or removed due to natality, mortality, immigration and emigration processes. Following the QPT ontology, the representation for the four basic population processes and their effects on *Size* becomes:

I+(Size, Born); I–(Size, Dead); I+(Size, Immigrated); I–(Size, Emigrated)

The domain theory should also include feedback loops that represent the effect that *Size* has on *Born*, *Dead*, and *Emigrated*. This is obtained by means of qualitative proportionalities:

P+(Born, Size); P+(Dead, Size); P+(Emigrated, Size)

This way, the combination of *I+(Size, Born)* and *P+(Born, Size)* reads as 'the amount of individuals being born should be added to the size' and 'when the population size changes (increases or decreases) the amount of individuals being born also changes in the same direction'. *Immigration* is not included in this feedback loop, because it is considered exogenous to the system. That is, the amount of inflow resulting from immigration does not depend on the population size. Instead, it is seen as an external factor that is determined outside the scope of the system.

These ideas are diagrammatically represented in Figure 3.9, using the workspace for defining model fragments in Homer. The model-fragment *Closed population* (LHS) is a subtype of *Population* (the latter has to be included in the model before the former can become active). This previously defined fragment introduces an instance of the entity *Population* (named *Population*) and the quantity *size* with a five-valued quantity space (all coloured green). The assumption *Closed Population* (no migration) is an additional condition (coloured red). The new knowledge added by the model-fragment *Closed Population* includes (coloured blue): quantities *born* and *dead* (both with the two-valued quantity space QS={Zero,Plus}), positive proportionalities (∝+) from *Size* to *Born* and *Dead*, and bi-directional correspondences between the values *Zero* for *Size*, *Born* and *Dead* (to express the idea that when size is zero that the other quantities must also be zero).

The model fragments *Born* (Figure 3.9, middle) and *Dead* (Figure 3.9, RHS) are both of type process. They require the model-fragment *Population* (coloured red) to exist, before they may become true. In addition, there is an inequality statement specifying that the *Size* of the population is 'greater than zero'[7] (coloured red). In other words, only if a population has some individuals these processes become active. The knowledge

[7] All lines with an arrow should be read following the direction of the arrow. In this case, the inequality should be read as 'zero < (current) Quantity (value)'.

introduced by the model-fragment *Born* is: the quantity *Born*, with value *Plus* currently assigned, and a positive influence (I+) from this quantity on *size*. The value *zero* represents that there is no natality, the value *Plus* means that individuals are being born. In summary, this model-fragment captures the idea that natality becomes active as soon as there is a population with some individuals, and that the number of individuals being born increases with the size of the population. The model-fragment *Dead* is essentially the same except it introduces a negative influence on the population size (I–). Thus, when a population exists there will be individuals dying, which reduces the population size.

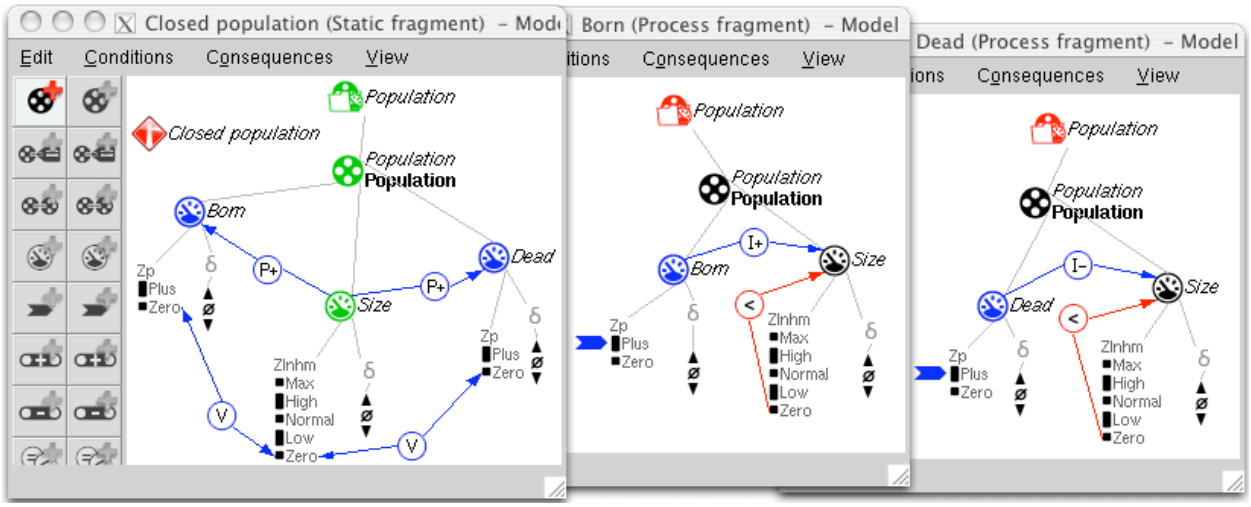


Figure 3.9: Model fragments for a simple population model

3.3 Running and Inspecting Models with Garp3: Simulate

Starting with the scenario (Figure 3.8) in which the value of *Size* is *Normal* and the derivative is unknown (<*Normal,?*>) and the library of model fragments (Figure 3.9), the reasoning engine builds up a complete simulation model that produces the results shown in Figure 3.10. The resulting causal model in state 1 is depicted at the LHS (top). It shows the two influences from *Born* and *Dead* on *Size* and the feedback via the proportionalities. *Size* has value *Normal* and is decreasing. *Born* and *Dead* are both *Plus* and decrease (because of the feedback from *Size*). As mentioned above, different states may show different sets of dependencies. For instance, the processes natality and mortality are not active in state 8, because *Size* has value *Zero*. The state-graph is depicted at the LHS (bottom). It shows that the following behaviours are possible [1 → 6 → 8], [1 → 7], [2], [3 → 4], and [3 → 5 → 9]. The matching quantity values are depicted in the value-history (RHS). Given that in the initial scenario *Size* has magnitude *Normal* and derivative unknown, the interpretation of the competing influences from *Born* and *Dead* leads to three states. In state 1 the population decreases (*Born* < *Dead*), in state 2 it remains stable (*Born* = *Dead*), and in state 3 it increases (*Born* > *Dead*). The behaviour captured in state 1 proceeds via state 6 to state 8, in which the population has become extinct. Alternatively, state 1 may proceed to state 7, in which case the population stabilises. In a similar way the increasing behaviour captured in state 3 may proceed to state 4, and stabilise, or further increase via state 5 and reach the maximum value in state 9. In summary, if both *Born* and *Dead* are positive, but their relative magnitudes are unknown all behaviours are possible. The population can grow to its maximum size, go extinct, or stabilise at any intermediate value.

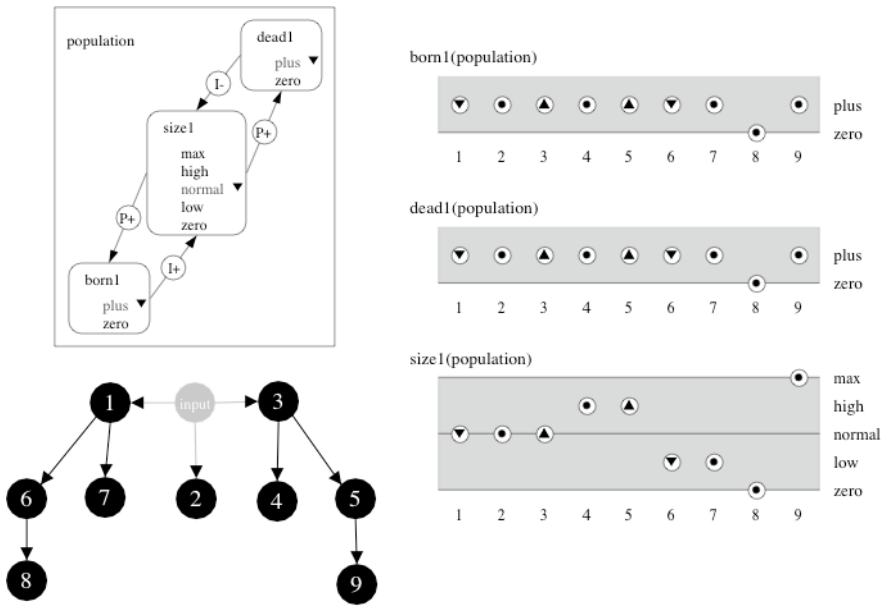


Figure 3.10: Simulation results for a closed population model.

3.4 Adding Migration to the Population model

The model constructed so far assumes a population without migration. Hence the population cannot recover from extinction and in the simulation there is no transition from state 8 in Figure 3.10. Figure 3.11 depicts three of the model fragments needed to implement migration. The *Open population* model-fragment (LHS) competes with the *Closed population* discussed above (Figure 3.9). It applies when the assumption *Open population* is true (thus, when specified in a scenario). In addition to the quantities *Born* and *Dead* it introduces *Emigrated* and *Immigrated*. Changes in population size also affect emigration, but not immigration. Next, a set of model fragments is required to implement the migration processes. *Emigration* and *Immigration* are shown in Figure 3.11 (middle and LHS). Similar to *Dead* and *Born* they start when a population exists. When active, *Emigration* has a negative influence on the population size and *Immigration* has a positive influence. *Colonization* is modelled as a special case of *Immigration* (not shown in the Figure). Descriptions of these two processes are similar, but *colonization* starts a new population where such a population does not exist. Hence, population *size* has to be *Zero*.

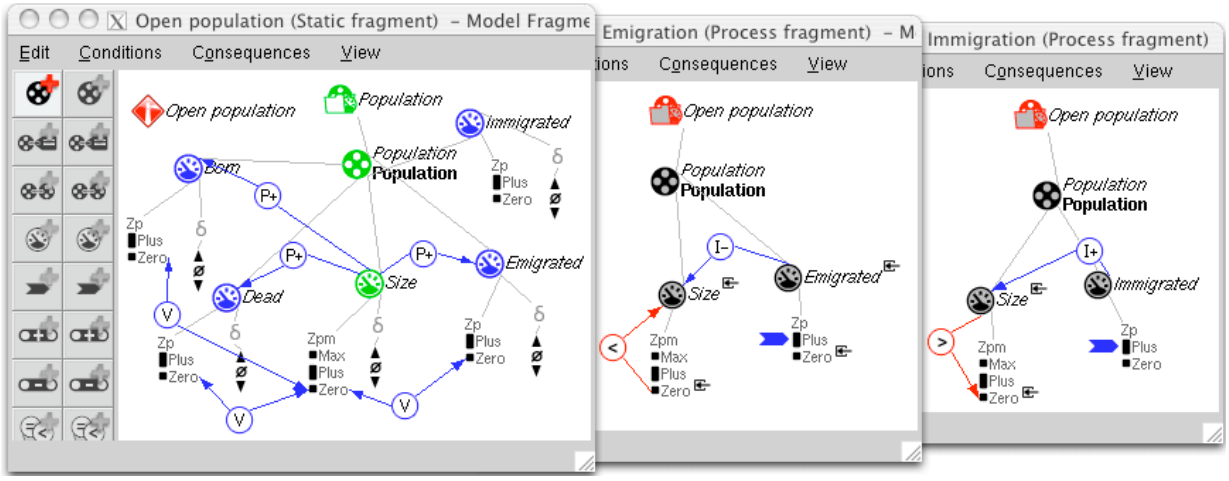


Figure 3.11: Model fragments for a population model including migration.

A simulation with this extended model is shown in Figure 3.12. For the sake of clarity, *size* has been given a three-valued quantity space QS={Zero,Plus,Max}. Also the causal model taken from state 9 (Figure 3.12, LHS, top) does not show all the available information. For instance, correspondences are not shown. Finally, the value-history shows a *particular* behaviour path, namely [13 → 14 → 18 → 9 → 11 → 15 → 1→ 13] (and not all values from all states). Notice, that this path implements a loop. In the initial scenario, the population *Size* has value *Plus* and an unknown derivative. The state-graph shows that nine states of behaviour match that initial description (states with numbers 1 through 9). State 9 is on the selected path and the value-history shows that the population is increasing, along with all the processes. This behaviour moves on to state 11, in which the population reaches its maximum size and stops growing. Next, the population may start to decrease (state15), reach the next lower value *Plus* (state 1), and then become extinct (state 13). Colonisation may then start (state 14), and create a new population which starts gaining size in state 18, and actually 'exists' in state 9.

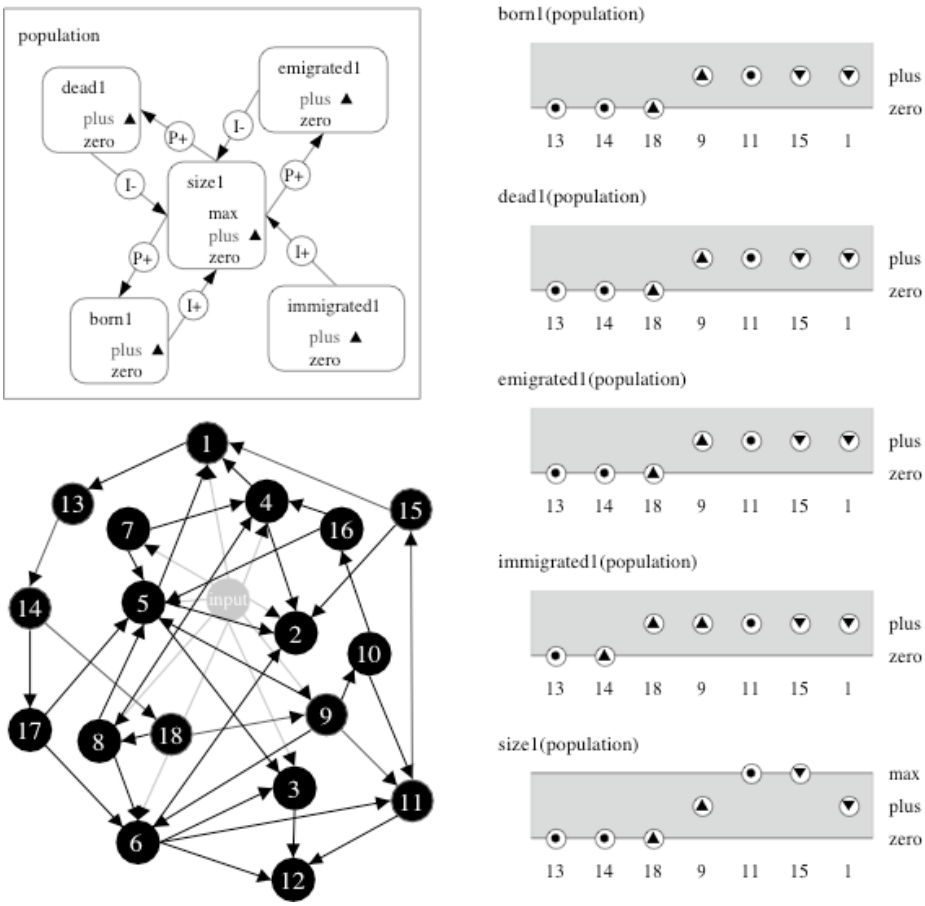


Figure 3.12: Simulation results with an open population model.

The transitions [13 → 14] and [11 → 15] are special from a QR point of view, as they do not reflect a value change due to increasing or decreasing. Instead, the derivative of *Immigration* changes, from ∂=[0] to ∂=[+] and from ∂=[0] to ∂=[–], respectively. This is the result of a special feature implemented into the Garp3 reasoning engine by means of which exogenous quantities can be assigned certain 'behaviour'. As if the external world behaves in a certain way. In this case, *Immigration* has been assigned 'exogenous sinus' (Figure 3.8) by specifying this assumption in the scenario. 'Exogenous sinus' can be used to enforce a continuous change on an exogenous quantity. As a results, *Immigration* changes following the pattern: ∂I=[0] → ∂I=[+] →

∂I=[0] → ∂I=[–] → ∂I=[0] → ∂I=[+] → etc. Assigning some default behaviour to exogenous quantities is required for representing population dynamics in order to enforce a disturbance of some kind through the system.

The examples discussed above present initial ideas on how to capture ecological knowledge using QR. Of course, many additional details can be represented yielding more advanced models and simulations that deliver insightful conclusions and explanations. The next chapter discusses examples of such applications.

4 Examples of QR Models

The goal of this chapter is to show some typical Qualitative Reasoning models. The following models are included:

- **Tree & shade.** This is a model that despite its simplicity illustrates many essential details relevant to QR models. This model has been used successfully for elementary courses on QR (see e.g. http://monet.aber.ac.uk/).

- **Population behaviour.** This set of five models describes the behaviour of a single population. The main goal of these models is to familiarise the reader with the different types of building blocks involved. The ideas presented in these models form the basis for complex models such as the Cerrado Succession Hypothesis (Salles & Bredeweg, 2003) and the Ants' Garden (Bredeweg & Salles, 2005).

- **Communicating vessels.** This is one of the most famous examples in the QR literature (see e.g. Weld & de Kleer, 1990; Forbus, 1984; Kuipers, 1984). Models of communicating vessels have been used to illustrate problems and solutions concerning many elementary issues in QR. Two models of communicating vessels are discussed within this report.

- **Heating & Boiling**. This is a typical example from Physics, focussing on issues related to thermodynamics. Two models are discussed within this report, each showing a specific approach to this system.

The models discussed here can be downloaded from the Qualitative Reasoning and Modelling (QRM) portal that is being developed as part of the NaturNet-Redime project. For downloading, visit: http://hcs.science.uva.nl/QRM/.

4.1 Tree & Shade

The tree and shade model is used to simulate a growing tree that casts a shadow on the ground. The assumption is made that a tree always grows, thereby ignoring the need for water, sunlight, air and minerals. The size of the shadow depends on the size of the tree. This is a dynamic process, as the shaded area increases as the tree becomes bigger. Indirectly, the growth of the tree causes the shaded area to increase.

4.1.1 Entity and agent hierarchy

The entity and agent hierarchies for the tree and shade model are very simple (Figure 4.1). The entity hierarchy consists of only a tree. Since there are no agents in the model, the agent hierarchy is empty.

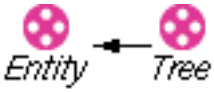


Figure 4.1: The entity hierarchy for the tree and shade model.

4.1.2 Assumption hierarchy

Since this model is meant to be as simple a possible, no assumptions are defined. A more advanced version of the model might include an assumption that indicates that the

tree always grows (i.e. that the requirements for growth are ignored).

4.1.3 Quantities and quantity spaces

The tree in our system, which grows and produces shade, has three important quantities, namely:

1. the size of the tree (*Size*);
2. the area of shade caused by the tree (*Shade*); and,
3. the rate at which the tree grows (*Growth rate*).

Since the focus of this model is on the growing of the tree, a quantity space with three consecutive qualitative values is chosen for the *Size* of the tree: *small*, *medium* and *large*. Since the *Shade* depends on the *Size* of the tree, it is logical to choose the same quantity space for that quantity. Finally, there is either no growth, or some positive amount of growth. Therefore, the possible qualitative values for *Growth rate* are: *zero* and *plus*. The quantities and their quantity spaces are summarised in Table 4.1.

Table 4.1: The quantities in the tree and shade model and their quantity spaces

Quantity	***Quantity Space***
Size	*{small, medium, large}*
Shade	*{small, medium, large}*
Growth rate	*{zero, plus}*

4.1.4 Scenarios

In the scenario named *a tree with small size,* a typical small tree is modelled (see Figure 4.2). The tree has two quantities, namely *Size* and *Shade*. Value assignments are used to indicate that both the *Size* and the *Shade* have the value *small*. A proper simulation should show the changes in the *Size* and the *Shade* of the tree growing.

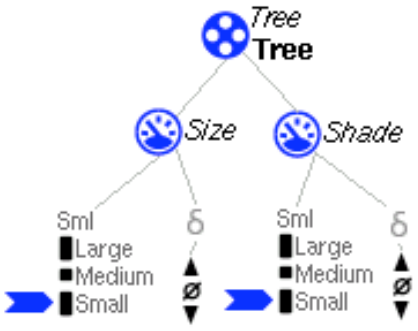


Figure 4.2: The scenario modelling a small tree casting a small shade.

4.1.5 Model fragments

The tree and shade model consists of two model fragments. The first, shown in Figure 4.3, is a static model fragment called *Tree with Shade*, which models the relationship between the *Size* of the *Tree* and its *Shade.* This is a positive proportionality from *Size*

to *Shade*, as the *Shade* increases when the *Size* increases and *shade* remains stable when *Size* is stable. There is also a quantity space correspondence in the model that indicates that during the simulation *Size* and *Shade* have the same magnitude values.

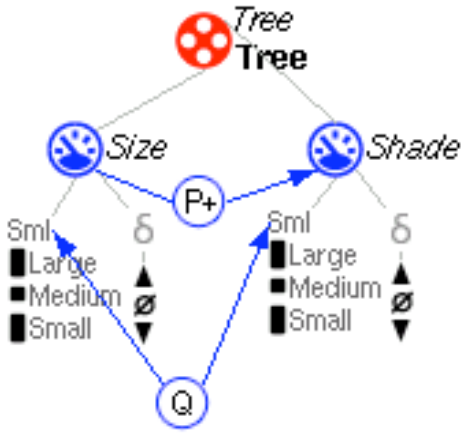


Figure 4.3: The static fragment modelling the relation between size and shade.

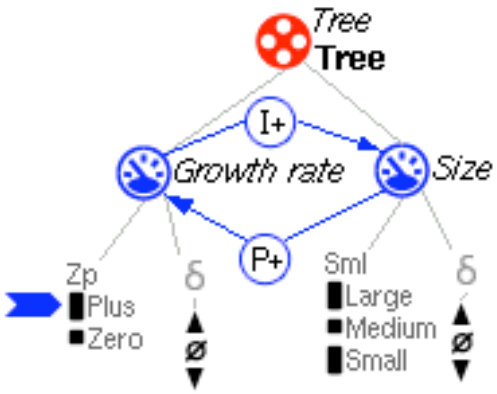


Figure 4.4: The process fragment modelling the growth of the tree.

The second model fragment, shown in Figure 4.4, is the process fragment *Growth of Tree*, which models the growth process that increases *Size*. This is modelled using a positive influence from *Growth rate* to *Size*. This has to be an influence, as *Size* can increase even when the *Growth rate* is stable. A bigger tree will grow faster, which is modelled using a positive proportionality from *Size* to *Growth rate* (whenever the *Size* increases, the *Growth rate* increases). Finally the assumption is modelled that a tree grows no matter what. This is done using by assigning the value *plus* to the magnitude of the *Growth rate.* Notice that this assumption could have been made explicit by defining an assumption and including it as a condition in this model fragment.

4.1.6 Simulation results

The state graph, shown in Figure 4.5, shows the results of simulating the scenario *a Tree with small size.* The corresponding quantity value history is shown in Figure 4.6. As expected the *Size* of the tree increases from *small* to *medium* to *large,* as does the *Shade.* At the same time the *Growth rate* keeps increasing within its *plus* interval.

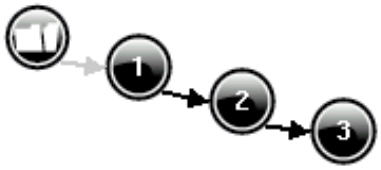


Figure 4.5: The state graph generated by simulating the tree with small size scenario.

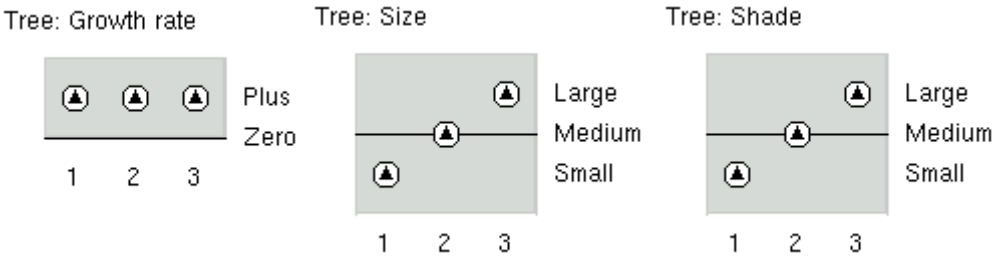


Figure 4.6: The quantity value history belonging to the state graph shown in Figure 4.5.

In each of the states all the model fragments fire. As a result, the relations between the quantities are aggregated, as can be seen from the dependency screen in Figure 4.7.

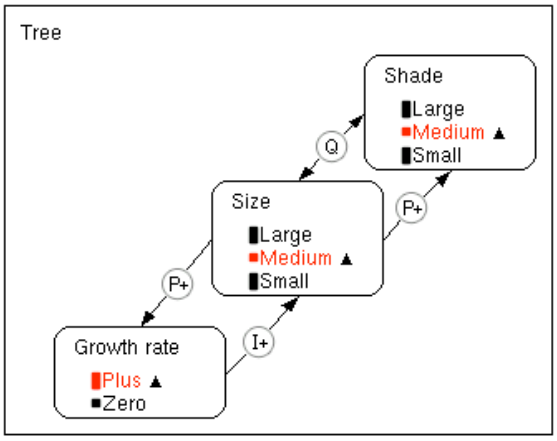


Figure 4.7: The dependencies in state 2 of the tree and shade model simulation.

4.1.7 Summary tables

Table 4.2: Entity summary

Entity	Super type	Description
Tree	Entity	A normal tree

Table 4.4: Quantity summary

Quantity	Entity/Agent	Quantity space	Description
Size	Tree	{small, medium, large}	The size of the tree.
Shade	Tree	{small, medium, large}	The shade cast by the tree.
Growth rate	Tree	{zero, plus}	The speed with which the tree grows.

Table 4.5: Scenario summary

Scenario name	A tree with small size
Initial values	Size(tree) = small, shade(tree) = small
Initial equations	None.
Description	The scenario models a small tree casting a small shade.

Table 4.6: Model fragment summary

Model fragment name	Tree with shade
Model fragment type	Static fragment
Model fragment parent	None
Description	The model fragment contains an entity tree as a condition, which introduces the quantities size and shade. There is a positive proportionality from size to shade, and a quantity space correspondence between the quantity spaces of their magnitudes.

Model fragment name	Growth of tree
Model fragment type	Process fragment
Model fragment parent	None
Description	The growth of tree model fragment has a conditional tree instance, which introduces the quantities size and growth rate. There is a positive proportionality from size to growth, and a positive influence from growth to size. Furthermore the magnitude of the growth rate is set to plus.

Table 4.7: Simulation summary

Scenario	A tree with small Size
Full simulation	3
Begin state(s)	[1]
End state(s)	[3]
Behaviour 1	[1] → [2] → [3]
Behaviour description	The growth rate, size, and shade all keep increasing. Size and shade move from small, to medium, to large.
Overall description	The behaviour is as expected.

4.2 General models of population behaviour

This chapter describes five examples of models about the behaviour of a single population. The main goal of these five models is to familiarise the reader with the different types of building blocks involved. First, model 1 is described, in which a *Birth* process causes an existing population of green frogs to grow. In the models 2-5, additional (competing) processes are added, as well as alternative representations, agents, and feedback. For population model 1, all model ingredients are explained. For models 2-5, only the differences with respect to the previous model(s) are discussed.

4.2.1 Population model 1: Basic population growth

Population model 1 describes the growth of a green frog population in a basic manner.

4.2.1.1 Entity and agent hierarchy

The entity hierarchy for the basic population model contains only one type of entity: *Population* (Figure 4.8). There are no agents defined in this model.

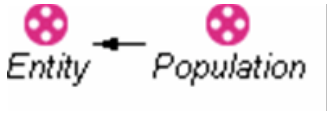


Figure 4.8: The entity hierarchy for the basic population model.

4.2.1.2 Assumption hierarchy

There are no explicit assumptions represented in this model.

4.2.1.3 Quantities and quantity spaces

There are three quantities in this model:

- *Number of:* The number of individuals in the population;
- *Biomass:* The amount of biomass involved in the population; and,
- *Birth:* A measure of how many individuals are born.

The *Number of* individuals and *Biomass* have a quantity space of three values {*small*, *medium*, *large*} along which an existing population can grow (or decrease). For *Birth*, the only relevant distinction is whether it is *zero*, or *plus*, hence the quantity space {zero, *plus*}. The quantities and their quantity spaces are summarised in Table 4.8.

Table 4.8: The quantities in the basic population model and their quantity spaces

Quantity	***Quantity Space***
Biomass	*{small, medium, large}*
Birth	*{zero, plus}*
Number of	*{small, medium, large}*

4.2.1.4 Scenarios

There is only one scenario, named *Population behaviour*. It is shown in Figure 4.9. Because this scenario is intended to show growing behaviour, *Biomass* and *Number of* are set to *small*, at the bottom of their quantity space.

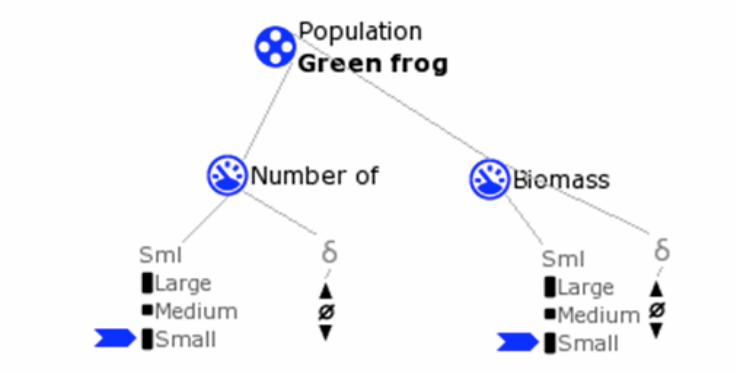


Figure 4.9: The scenario for a basic population.

4.2.1.5 Model fragments

There are two model fragments defined in this model: *Population* (static) and *Birth* (process). The model fragment *Population* (shown in Figure 4.10) models generic knowledge about any population, (*i.e.*, that when the *Number of* individuals changes, *Biomass* changes in the same direction), this knowledge is modelled by the positive proportionality (P+) from *Number of* to *Biomass*. To ensure that a value change in *Number of* corresponds to a similar value change in *Biomass*, also a quantity space correspondence (Q) is added.

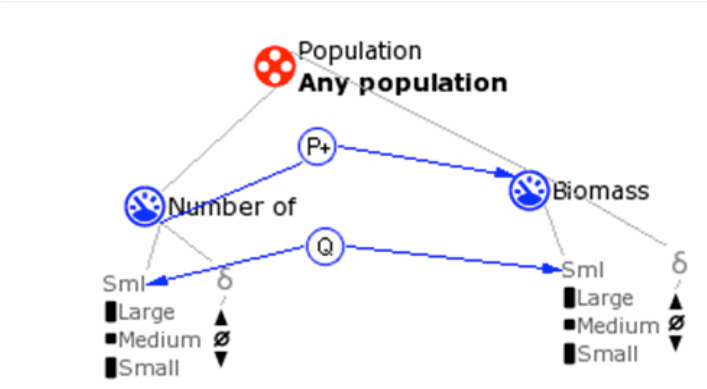


Figure 4.10: The model fragment for Population.

Figure 4.11 shows the model fragment for the *Birth* process. In this basic model, there are no conditions, except the existence of a population, so the *Birth* process automatically applies to any population. As a consequence, this model fragment introduces a quantity *Birth*, and a positive influence (I+) from *Birth* to *Number of*. The value of *Birth* is set to *plus*, so that there will indeed be an effect.

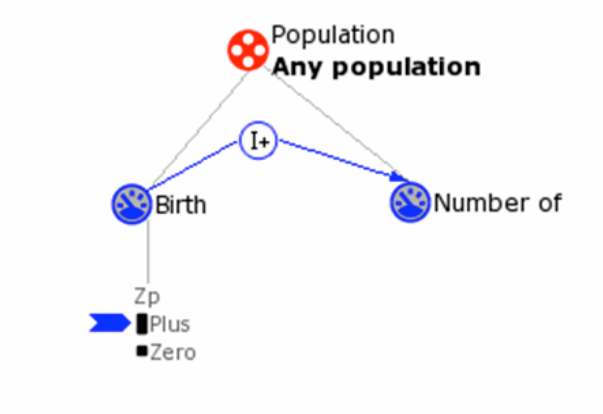


Figure 4.11: The model fragment for the *Birth* process.

4.2.1.6 Simulation results

The resulting state graph for the Population behaviour scenario consists of a sequence of 3 states: 1 → 2 → 3 (see Figure 4.12). There is one begin state (1) and one end state (3). There are no branching points.

Figure 4.12: The state graph for the population behaviour scenario.

The value histories for the three quantities are shown in Figure 4.13. As specified in the scenario, the value of *Biomass* and *Number of* are *Small* in state 1, and both increase via *medium* to *large* in end state 3. The value of *Birth* remains stable at *plus* throughout all states.

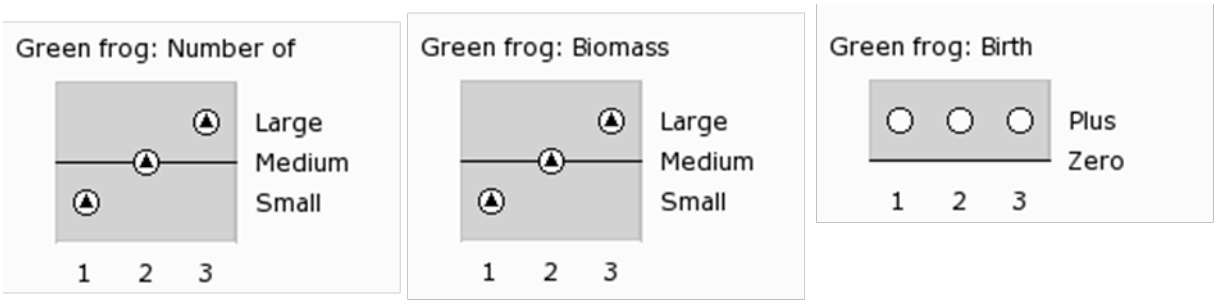


Figure 4.13: The value history for number of, biomass and birth.

The transition history (displayed in Figure 4.14) shows that both *Biomass* and *Number of* change simultaneously, first to a point (*medium*) above their start value, then to the interval (*large*) above that point.

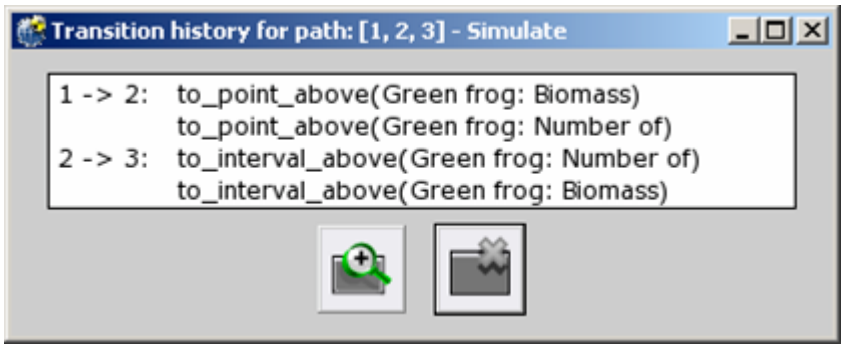


Figure 4.14: The transition history.

4.2.1.7 Summary tables

The contents of the complete model are summarised in tables below. There are no configuration definitions or attribute definitions present in this model.

Table 4.9: Entity summary

Entity	Super type	Description
Green frog	Population	The general concept of a population of individuals of some biological type

Table 4.10: Quantity summary

Quantity	Entity/Agent	Quantity space	Description
Biomass	Population	{small, medium, large}	The amount of biomass involved in the population
Birth	Population	{zero, plus}	A measure of how many individuals are born
Number of	Population	{small, medium, large}	The number of individuals involved in the population

Table 4.11: Scenario summary

Scenario name	Population Behaviour
Initial values	Biomass = small Birth = plus Number of = small
Initial equations	None
Description	Biomass and number of should increase to large due to the influence of birth

Table 4.12: Model fragment summary

Model fragment name	Population
Type/Parent	Static
Description	Specifies that biomass is positively proportional (P+) to number of, and that there is a quantity correspondence (Q) between these two quantities.

Model fragment name	Birth
Type/Parent	Process
Description	The birth process introduces a positive flow (I+) from birth (which is plus) to number of.

Table 4.13: Simulation summary

Scenario	1. Population behaviour
Full simulation	3 states
Begin state(s)	[1]
End state(s)	[3]
Behaviour 1	[1 → 2 → 3]
Behaviour description	The values of *Biomass* and *Number of* are *Small* in state 1, and both increase via *Medium* to *Large* in end state 3. The value of *Birth* remains *Plus* throughout all states.
Overall description	The elementary scenario results in a simple simulation, due to two reasons: the influencing quantity remains stable, and there is a quantity correspondence between *Biomass* and *Number of*.

4.2.2 Population Model 2: Competing Processes

4.2.2.1 Model Ingredients and Simulation Results

In population model 2, the *Death* process is added as a model fragment (see Figure 4.15). In this model fragment, a quantity *Death* is introduced, with a negative influence on *Number of*.

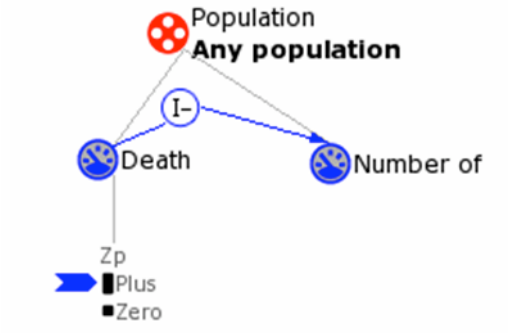


Figure 4.15: The model fragment for the death process.

The addition of *Death* leads to ambiguity, as *Birth* and *Death* are competing processes (see Figure 4.16). *Number of* may increase due to *Birth*, decrease due to *Death*, or the influences may balance each other out.

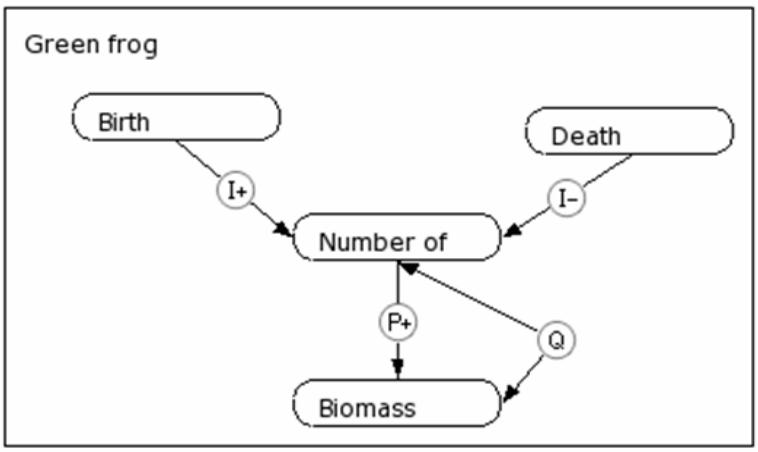


Figure 4.16: Competing influences from birth and death.

In the scenario, the relationship between *Birth* and *Death* is deliberately left unspecified, so the simulation should demonstrate all three possibilities. When the *Death* process is dominant, the population will decrease, and may become extinct. To allow this possibility to occur in the simulation, the value *zero* is added to the quantity space of the quantity *Number of* and *Biomass*, below the value *small*. In the scenario, the initial value of *Number of* is set to *small*, so that it is indeed possible for this quantity to increase, remain stable, or decrease.

The simulation that results is shown in Figure 4.17. It contains 8 states and 5 behaviours, starting from state 1, 2 and 3, respectively. Figure 4.18 shows an overview of the values for *Birth*, *Death*, and *Number of* in this simulation. The values for *Biomass* are not shown, as they correspond fully to those for *Number of*. When inspecting Figure 4.17 and 18 together, the overall behaviour becomes clear. In state 1, *Number of* decreases, in state 2, it remains stable, and in state 3, it increases. After decreasing in state 1, it reaches *zero* in state 6, as expected. After increasing in state 3, there are three possibilities: *Number of* can stabilize at *medium* (state 4), it can stabilize at *large* (state 7), or it can continue to grow at *large* (state 8).

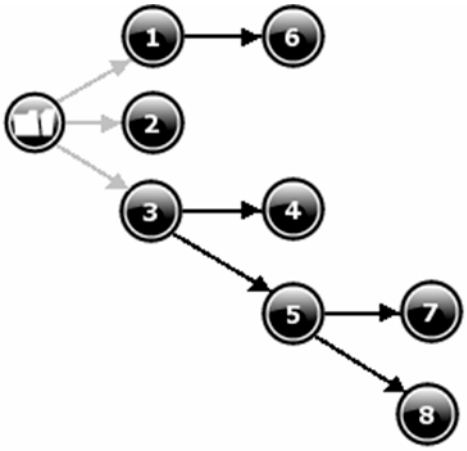


Figure 4.17: The state-transition graph for model 2.

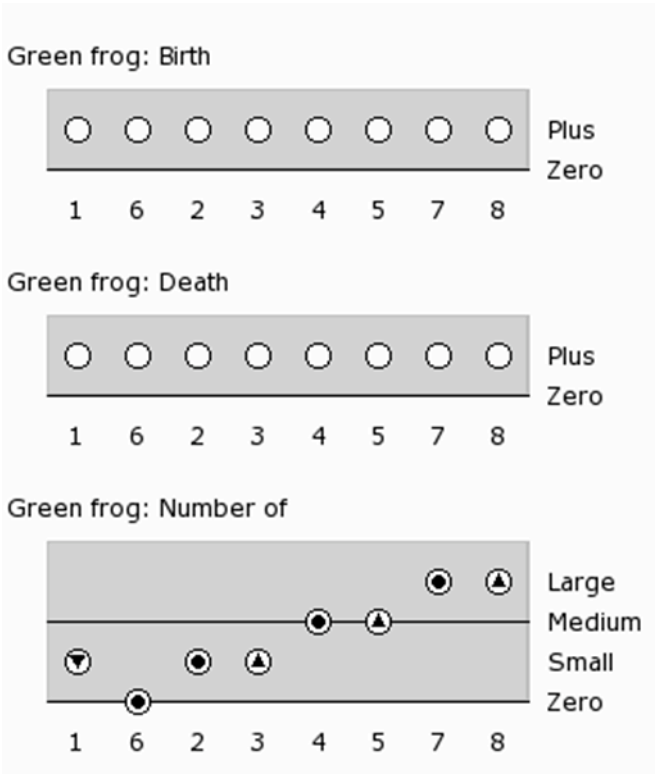


Figure 4.18: An overview of all values for the simulation of model 2.

4.2.2.2 Summary tables

A summary of all changes with respect to this second model is presented in the tables below.

Table 4.14: Quantity summary

Quantity	Entity/Agent	Quantity space	Description
Death	Population	{zero, plus}	A measure of how many individuals are born

Table 4.15: Scenario summary

Scenario name	Population Behaviour
Initial values	Number of = small
Initial equations	None
Description	Number of may decrease, remain stable, or increase due to the influences of death and birth processes

Table 4.16: Model fragment summary

Model fragment name	Death
Type/Parent	Process
Description	The death process introduces a negative flow from death (which is plus) to number of.

Table 4.17: Simulation summary

Scenario	1. Population behaviour
Full simulation	8 states
Begin state(s)	[1, 2, 3]
End state(s)	[2, 4, 6, 7, 8]
Behaviour 1	[1 → 6]
Behaviour description	The quantity number of decreases from small to zero, i.e., the population becomes extinct.

Behaviour 2	[2]
Behaviour description	The quantity number of remains stable at small, indicating that birth and death are in balance.
Behaviour 3	[3 → 4]
Behaviour description	The quantity number of increases from small to medium, after which birth and death reach a balance.
Behaviour 4	[3 → 5 → 7]
Behaviour description	The quantity number of increases from small to large, after which birth and death reach a balance.
Behaviour 5	[3 → 5 → 8]
Behaviour description	The quantity number of increases from small to large, and keeps increasing.
Overall description	The variety of behaviours arises due to the competing influences of birth and death.

4.2.3 Population model 3: An Alternative representation

4.2.3.1 Model Ingredients and Simulation Results

Model 3 is an alternative representation of model 2, also modelling a population with *Birth* and *Death* processes. Model 3 is based on the following principles:

1. Introduce the quantities of interest as early as possible.
2. There is feedback from the state quantity (*Number of*) to the process rates.
3. Flow rates should be conditional.
4. Processes should disappear when the population disappears.

To address point 1, all quantities are included in the *Population* model fragment (see Figure 4.19). This model fragment also implements point 2, by modelling positive feedback effects (P+) from *Number of* to the process rate quantities *Birth* and *Death,* (for example, when *Number of* increases, there will also be more *Birth* and *Death)*.

Point 3 (flow rates should be conditional) is addressed by adding a conditional value statement to the process model fragments. As is shown in Figure 4.20 for the *Birth* process, the processes should only apply when *Number of* > Zero (the same holds for *Death*).

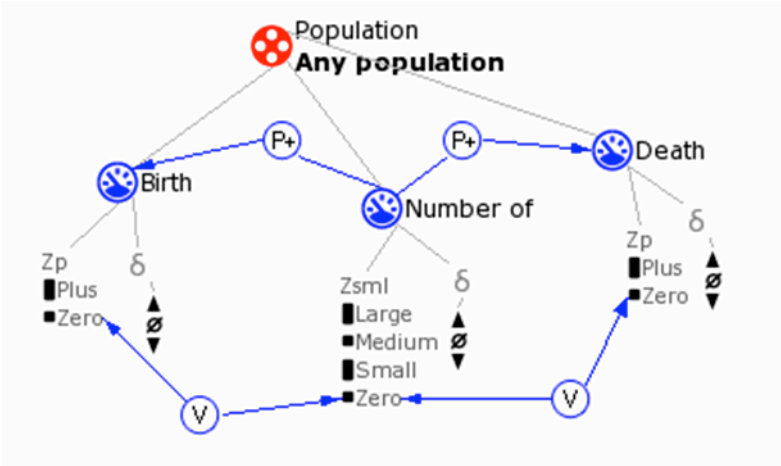


Figure 4.19: The revised model fragment for Population.

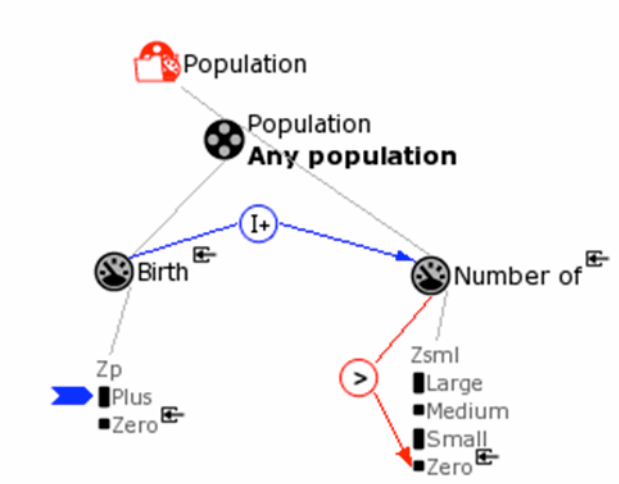


Figure 4.20: The revised model fragment for the *Birth* process.

Point 4 (processes should disappear when the population disappears) is related to point 3, and further addressed by the value correspondences (V) that were also visible in Figure 4.19.

Also, three extra model fragments are added (as subtypes of the *Population* model fragment) to model the different situations which may arise: *Birth* > *Death*, *Birth* = *Death*, and *Birth* < *Death*.

The scenario has not changed with respect to the previous model.

The simulation for this model yields three different behaviours, as shown by the state graph in Figure 4.21, and the value histories in Figure 4.22. *Number of* can increase (state 1), decrease (state 2), or remain stable (state 3). Compared to the previous model, an important difference is that the behaviour of *Birth* and *Death* follows that of *Number of*. For example, in state 4, the value of *Birth* and *Death* is *zero*, just as *Number of* is *zero*. Another difference is that there are less states and behaviours than in model 2. This is due to the extra model fragments with the ordinal relationships between *Birth* and *Death*. For example, once *Number of* increases, it does not stabilize anymore due to the model fragment *Birth greater than Death*.

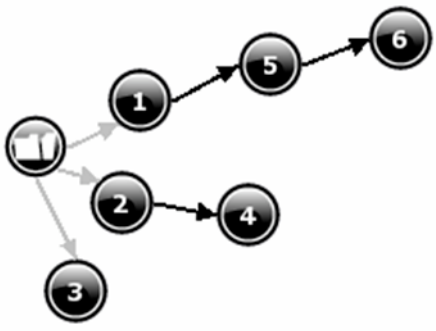


Figure 4.21: The state-transition graph for the model 3 simulation.

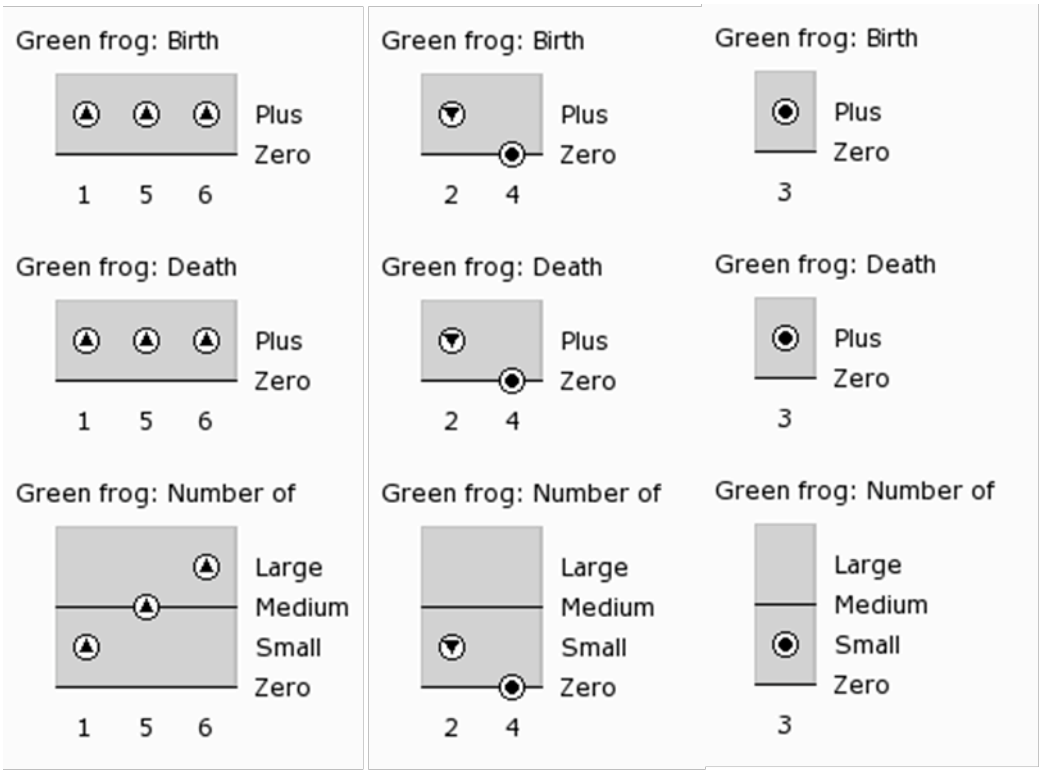


Figure 4.22: The value histories for the model 3 simulation.

Figure 4.23 presents an overview of the inequalities in the different states of the simulation, which shows the balance between the processes in state 4 and 3, and the equality *Number of* = Zero in state 4.

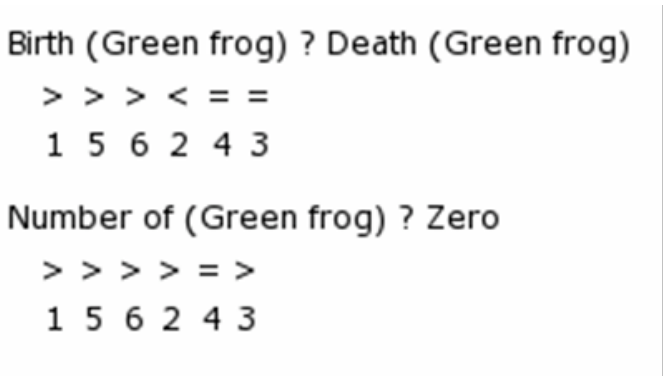


Figure 4.23: The inequalities for the model 3 simulation.

4.2.3.2 Summary tables

A summary of the changes to this third model is presented in tables below

Table 4.18: Scenario summary

Scenario name	Population Behaviour
Initial values	Number of = small
Initial equations	None
Description	Number of may decrease, remain stable, or increase due to the influences of death and birth processes

Table 4.19: Model fragment summary

Model fragment name	Population
Type/Parent	Static
Description	Consequences: Added value correspondences between zero (birth) and zero (number of) and between zero (death) and zero (number of). Positive proportionalities (P+) added from number of to birth and to death to model feedback.

Model fragment name	Birth equal to death
Type	Static
Parent	Population
Description	Condition: birth = death

Model fragment name	Birth greater than death
Type	Static
Parent	Population
Description	Condition: birth > death

Model fragment name	Birth Smaller than death
Type	Static
Parent	Population
Description	Condition: birth < death

Model fragment name	Birth
Type/Parent	Process
Description	Conditions: Number of > Zero. Consequences: the positive influence (I+), and value assignment birth = plus.

Model fragment name	Death
Type/Parent	Process
Description	Conditions: Number of > zero. Consequences: the negative influence (I-), and value assignment death = plus.

Table 4.20: Simulation summary

Scenario	1. Population behaviour
Full simulation	6 states
Begin state(s)	[1, 2, 3]
End state(s)	[3, 4, 6]
Behaviour 1	[1 → 5 → 6]
Behaviour description	The quantity number of increases from small to large, and keeps increasing. Birth and death also keep increasing, due to the feedback effects.
Behaviour 2	[2 → 4]
Behaviour description	The quantity number of decreases to zero; the population becomes extinct. Now, also birth and death decrease to Zero.
Behaviour 3	[3]
Behaviour description	The quantity number of remains stable at small, indicating that birth and death are in balance.
Overall description	Number of can increase, decrease, or remain stable.

4.2.4 Population model 4: An agent for colonisation and immigration

4.2.4.1 Model Ingredients and Simulation Results

The population in model 3 suffered from the problem of irreversible change. When the population size (*Number of*) was *zero*, there was no way for the population to recover again. To address this issue, colonisation and immigration are included in the model. This can be regarded as an external influence to the system. To be able to model immigration as a migration process, the entity hierarchy is expanded with *Habitat*, as a kind of *Ecosystem* (see Figure 4.24), and new configurations (*Lives In, Lives Outside*) are added to be able to specify who lives where.

Figure 4.24: The entity hierarchy for model 4.

The scenario (see Figure 4.25) specifies that our *Green Frog* population *Lives in* the *Pond*, while there exists also a neighbouring *Green Frog Reservoir*, which *Lives outside* the *Pond*. The value of *Number of* is set to *zero*, in order to check whether the colonisation is able to cause the population to start growing.

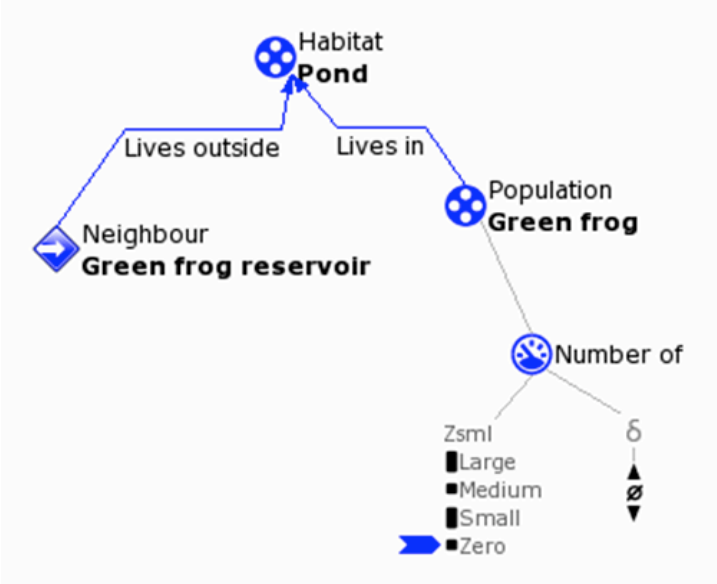


Figure 4.25: The scenario for model 4.

The *Green frog reservoir* is an *agent,* the source of the external influence. Furthermore, two agent model fragments are defined: *Colonisation*, which is active when a population does not (yet) exist (condition: *Number of* = zero), and *Immigration*, which is active only when a population exists (condition: *Number of* > zero). Apart from this difference in conditions, both model fragments are identical: a new quantity *Immigration* is introduced, with a positive influence on *Number of*. The *Colonisation* model fragment is shown in Figure 4.25.

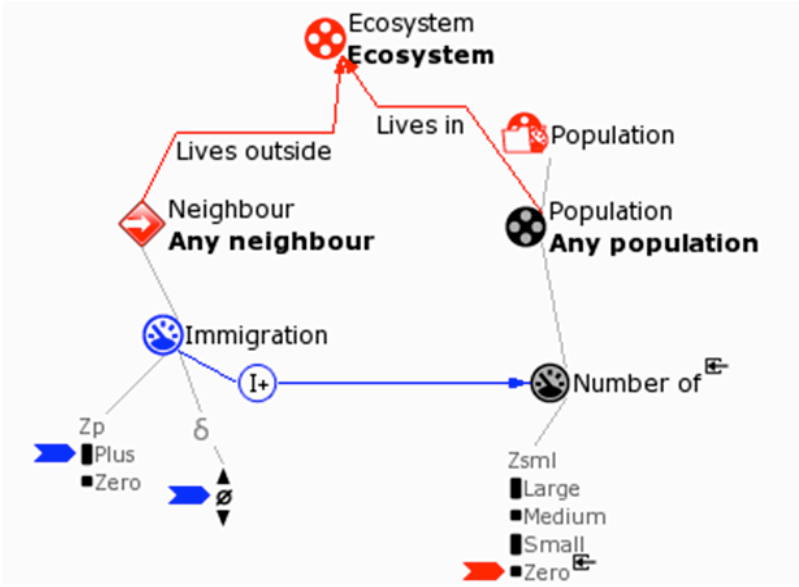
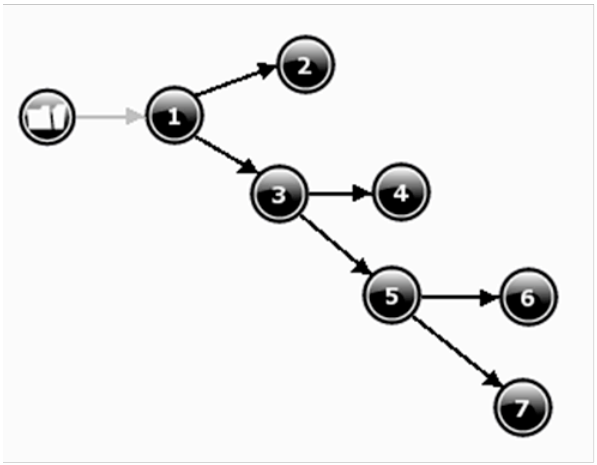


Figure 4.25: The agent model fragment for colonisation.

The simulation results are shown in Figure 4.26. The value history for the quantity *Birth* (not shown) is identical to that of *Death*. As expected, *Number of* increases due to *Colonisation*, and may increase further due to *Immigration* and *Birth*, or stabilise at one of its possible values.



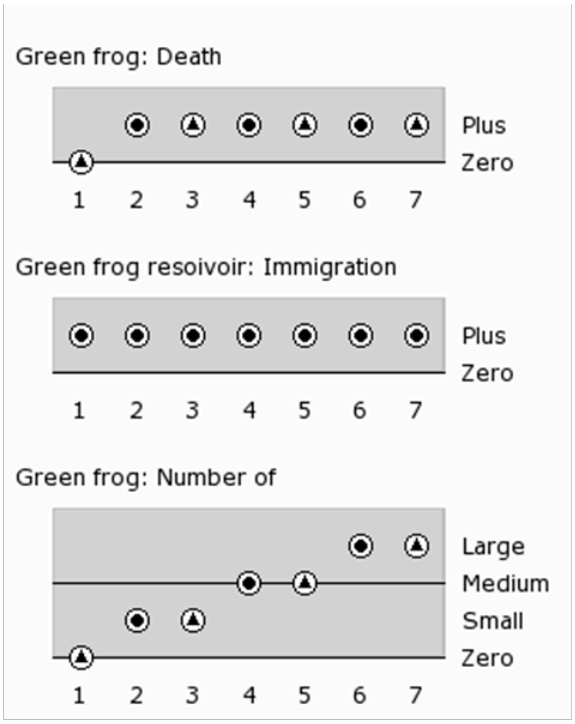


Figure 4.26: The state-transition graph and value history for the model 4 simulation.

4.2.4.2 Summary tables

A summary of changes with respect to model 4 is presented in table below.

Table 4.21: Entity summary

Entity	Super type	Description
Ecosystem	Entity	A biological community of interacting organisms and their physical environment

Habitat	Ecosystem	The natural home of an organism

Table 4.22: Configuration summary

Configuration	**Entity** *(from)*	**Entity** *(to)*	**Description**
Lives in	Green Frog	Pond	Specifies the frogs' habitat
Lives outside	Green Frog Reservoir	Pond	Specifies that there is another large population of green frogs elsewhere

Table 4.23: Quantity summary

Quantity	**Entity/Agent**	**Quantity space**	**Description**
Immigration	Green frog reservoir	{Zero, plus}	The amount of immigration coming from the green frog reservoir (to the pond)

Table 4.24: Agent summary

Agent	**Super type**	**Description**
Neighbour	Agent	A neighbouring agent which may cause immigration

Table 4.25: Scenario summary

Scenario name	Population behaviour
Initial values	Number of = zero
Initial equations	None
Description	The two configurations specified in the table above are added to the scenario.

Table 4.26: Model fragment summary

Model fragment name	Colonisation
Type/Parent	Agent
Description	Introduces a quantity immigration, with value plus and stable, and a positive influence (I+) from immigration to number of. Condition: Number of = zero.

Model fragment name	Immigration
Type/Parent	Agent
Description	Same as colonisation, but with condition: number of > zero

Table 4.27: Simulation summary

Scenario	Population Behaviour
Full simulation	7 states
Begin state(s)	[1]
End state(s)	[2,4,6,7]
Behaviour 1	[1 → 2]
Behaviour description	The population increases in size from zero to small
Behaviour 2	[1 → 3 → 4]
Behaviour description	The population increases in size from zero to medium
Behaviour 3	[1 → 3 → 5 → 6]
Behaviour description	The population increases in size from zero to large
Behaviour 4	[1 → 3 → 5 → 7]
Behaviour description	The population increases in size from zero to large and keeps increasing

Overall description	The green frog population living in the pond increases in size due to the positive external influence of the immigration from the neighbouring reservoir. The different behaviours arise from the possibilities of stabilizing at the different levels Low, medium, or large, or not stabilizing at all.

4.2.5 Population model 5: Negative, Neutral, and Positive growth

4.2.5.1 Model Ingredients and Simulation Results

Model 5 is intended to get the reader acquainted with the assumption mechanism in Garp3. To this end, model 3 is reformulated in slightly different terms. Only one rate is used, *Growth*, and the assumption mechanism of the Garp3 engine is used to generate alternative values for Growth. This is done by making a general model fragment called *Growth*, and creating three subtypes of this model fragment, for *Positive growth*, *Neutral growth* (i.e., no change), and *Negative growth*. As an example, the model fragments for growth and *Positive growth* are presented in Figure 4.27.

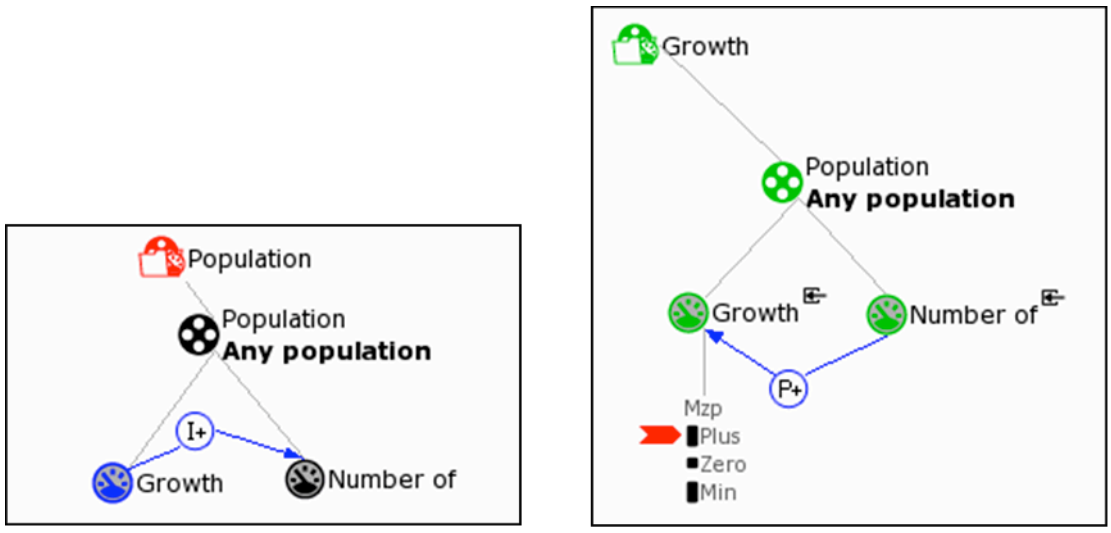


Figure 4.27: The model fragments for growth and positive growth.

In each of the three subtypes (positive, neutral and negative growth), the value of the *Growth* quantity is set to a specific value (*plus*, zero, and min, respectively). Also, feedback effects are added from *Number of* back to *Growth*. In the case of *Positive growth*, the feedback is positive (P+), *i.e.*, when *Number of* increases, Growth will increase too. In the case of *Negative growth*, the feedback is negative (P-), to ensure that when *Number of* decreases, the (negative!) growth will increase towards zero.

The scenario is the same as in model 3 (*Number of* = *small*).

When running the simulation, Garp3 will try to apply each of the model fragments. Because they are all possible, but mutually exclude each other, three different start states are generated. As shown in Figure 4.28, *Number of* may decrease from *small* (state 1) to zero (state 5), or stabilise already at *small* (state 3). *Number of* may also remain stable from the beginning (state 3 is also a begin state), or increase from *small* to *large* (from state 2 via 4 to 6). Thus, the simulation exhibits negative growth, neutral growth, and positive growth.

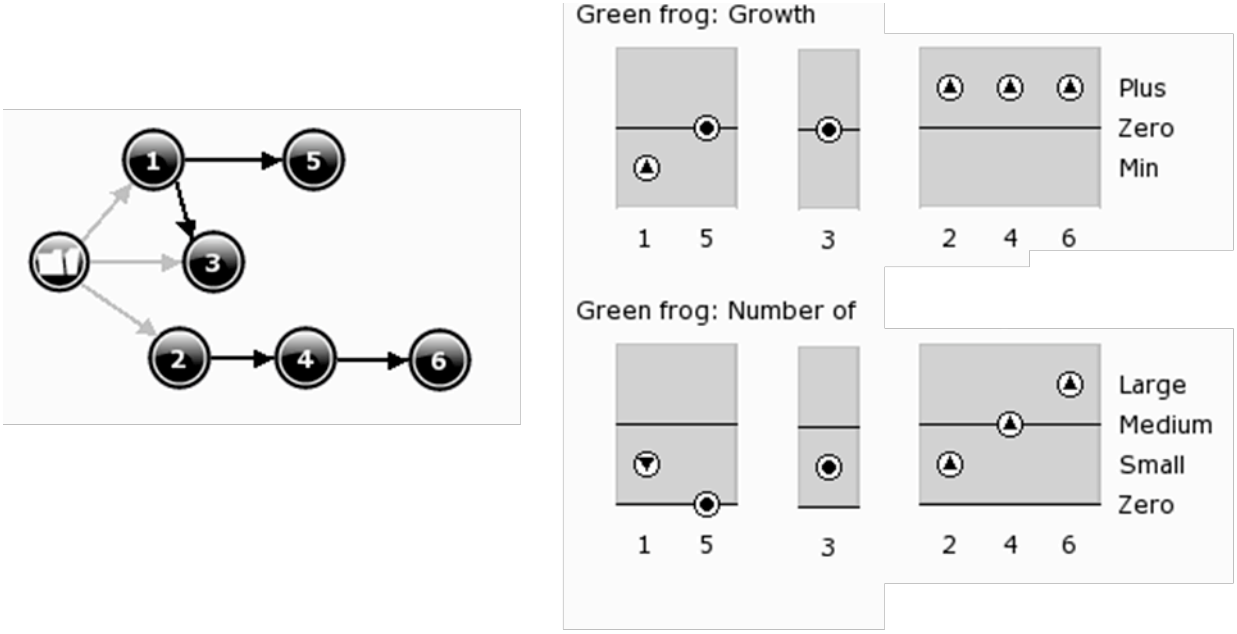


Figure 4.28: The state-transition graph and value histories for the model 5 simulation.

4.2.5.2 Summary tables

All changes to the model compared to the previous models are summarised in the tables below.

Table 4.28: Scenario summary

Scenario name	Population behaviour
Initial values	Number of = small
Initial equations	None
Description	*Number of* may increase, remain stable, or decrease

Table 4.29: Model fragment summary

Model fragment name	Growth
Type/Parent	Process
Description	Introduces a quantity growth, and a positive influence (I+) from growth to number of. The value of growth is deliberately left unspecified.

Model fragment name	Positive
Type	Process
Parent	Growth
Description	Specifies the value of growth as being plus. Also, a positive feedback proportionality (P+) is defined from number of to growth

Model fragment name	Neutral
Type	Process
Parent	Growth
Description	Specifies the value of growth as being zero. Also, a positive feedback proportionality (P+) is defined from number of to growth. Note that a negative proportionality would have the same effect here, because growth is Zero, according to this model fragment.

Model fragment name	Negative
Type	Process
Parent	Growth
Description	Specifies the value of growth as being min. Also, a negative feedback proportionality (P-) is defined from number of to growth.

Table 4.30: Simulation summary

Scenario	Population Behaviour
Full simulation	6 states
Begin state(s)	[1,2,3]
End state(s)	[3,5,6]
Behaviour 1	[1 → 5]
Behaviour description	The population decreases from small to zero
Behaviour 2	[1 → 3]
Behaviour description	The population decreases, and stabilises at small
Behaviour 3	[3]
Behaviour description	The population remains stable at small
Behaviour 4	[2 → 4 → 6]
Behaviour description	The population increases from small to large, and keeps increasing
Overall description	In this simulation, there is negative, neutral, and positive growth. The feedback loops allow the negative growth to stabilize to neutral, thereby creating behaviour 1 and 2. Neutral growth from the beginning is exemplified in behaviour 3, and positive growth is apparent in behaviour 4.

4.3 Communicating Vessels

The communication vessel system consists of a number of vertical containers, which are connected at the bottom by a pipe. The fluid has the same height everywhere, provided the containers contain the same liquid, and the bottoms of the containers are on the same height. If one of the containers is filled, a flow will level the fluid in each of the containers. Therefore, increasing or decreasing the amount of fluid in one of the containers will affect the height in the other containers too.

This phenomenon occurs, because the fluids, which are acted upon by gravity, cause equilibrium of the pressures of fluids. These pressure magnitudes depend only on how far the fluid surfaces are from the bottom of each container. This means that the pressure is not affected by the width or the shape of the container, but depends entirely on the height of the fluid column; in effect, the connected fluid containers represent a single vessel in which the heights of the fluid will always become equal.

4.3.1 Entity and agent hierarchy

The communicating vessel system consists of containers and pipes, which are both objects. The containers can contain liquids, which are substances, such as water and oil. These entity types are defined in the entity hierarchy shown in Figure 4.29. There are no agents defined in the communicating vessels model.

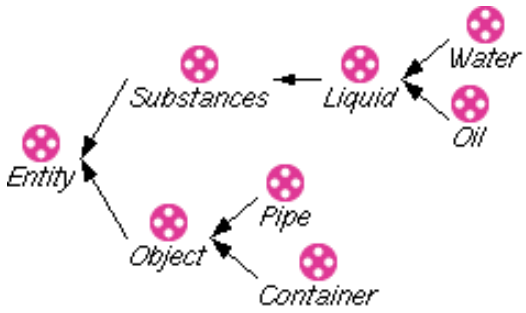


Figure 4.29: The entity hierarchy of the communicating vessel system.

4.3.2 Assumption hierarchy

The communicating vessel model does not contain any explicit assumptions.

4.3.3 Quantities and quantity spaces

From a qualitative point of view the height of the liquid in a container could be zero (empty), or have some positive value, or be full. Therefore, the quantity space chosen for *Height* is *{zero, plus, max}*. Analogous to *Height*, the *Pressure* and *Amount* of liquid can be zero, or have some value, or be maximal. For that reason, the same quantity spaces are chosen for *Amount* and *Pressure*. Finally, a quantity space has to be chosen for *Flow*. There can either be no flow, negative flow (from right to left), or positive flow (from left to right). So the quantity space chosen for *Flow* is *{min, zero, plus}*. A summery of the quantities and their quantity spaces can be seen in Table 4.31.

Table 4.31: The quantities in the tree and shade model and their quantity spaces

Quantity	***Quantity Space***
Amount	*{zero, plus, max}*
Height	*{zero, plus, max}*
Pressure	*{zero, plus, max}*
Flow	*{min, zero, plus}*

4.3.4 Scenarios

In the communicating vessel model three scenarios are defined. Each of these scenarios encompasses two containers which each contain either *Oil* or *Water*. The container is connected to a pipe using a *From* relation, and the pipe is connected to the other container using a *To* relation. These different configurations are used to indicate what the positive flow direction is. The *Water* or *Oil* instances each have a quantity *Height*, of which each has been given the value *plus*. The difference between the three scenarios is that there is a different inequality between the two *Height* quantities. In the first the height of the *Oil left* is greater than the *Oil right*, in the second the heights are equal, while in the third *Oil right* is greater than the *Oil left*. The first scenario can be seen in Figure 4.30.

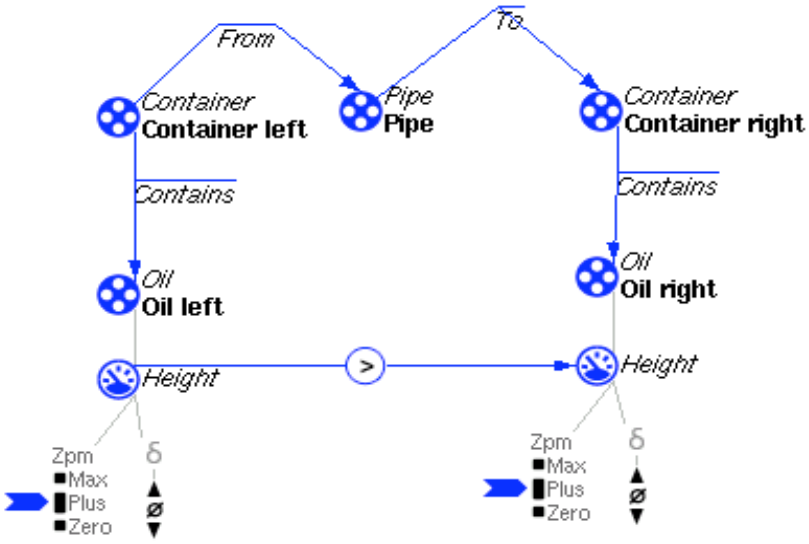


Figure 4.30: The scenario in which the height on the left is bigger than on the right.

4.3.5 Model fragments

The communicating vessel model has two model fragments that describe the structure and behaviour of the system. The first one: *Contained liquid*, models a contained liquid (Figure 4.31). It has a *Container* that contains *Liquid* as conditional elements, and introduces the quantities *Amount, Height,* and *Pressure.* There are positive proportionalities from *Amount* to *Height,* and from *Height* to *Pressure.* This is because if *Amount* changes, *Height* changes in the same direction, and *Height* is stable if *amount* is stable. The same is true for *Height* and *Pressure.* Since the *Amount*, *Height* and *Pressure* are all *max* at the same time, and *zero* at same time, they also have to be in the interval *plus* at the same time. This is modelled using quantity space correspondences between the quantity spaces of the magnitudes of these quantities (between *Amount* and *Height,* and between *Height* and *Pressure*). Finally there is an equality relation between *Height* and *pressure*.

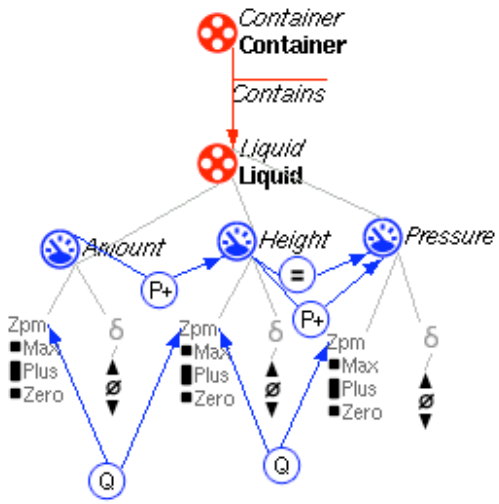


Figure 4.31: The contained liquid model fragment.

The second model fragment is *Liquid flow* and is shown in Figure 4.32, which shows only the model ingredients that are relevant in this context (being able to hide model

ingredient is a feature of the Garp3 workbench). The model fragment reuses the *Contained liquid* model fragment; once for the 'left container', and once for the 'right container'. The left *Container* is connected via a *From* relation to a conditional *Pipe* entity instance, and the *Pipe* is related to the right *Container* via a *To* relation. The *Pipe* entity introduces the quantity *Flow*, which, if it has a positive value, increases the *Amount* of fluid on the right side, and decreases the *Amount* of fluid on the left side. This is modelled using a positive influence from *Flow* to the *Amount* on the right side, and a negative influence from *Flow* to the *Amount* on the left side. The *Flow* is calculated by subtracting the *Pressure* on the left side from the *Pressure* on the right side. The result is assigned to the *Flow* using an equality relationship. In order to let the simulator calculate the derivative of the *Flow*, the change of the *Flow* has to be modelled. When the *Pressure* on the left side increases, the *Flow* increases, and when the *Pressure* on the right sided increases, the *Flow* decreases. This is modelled using a positive proportionality from the left *Pressure* to *Flow*, and a negative proportionality from the right *Pressure* to *Flow.*

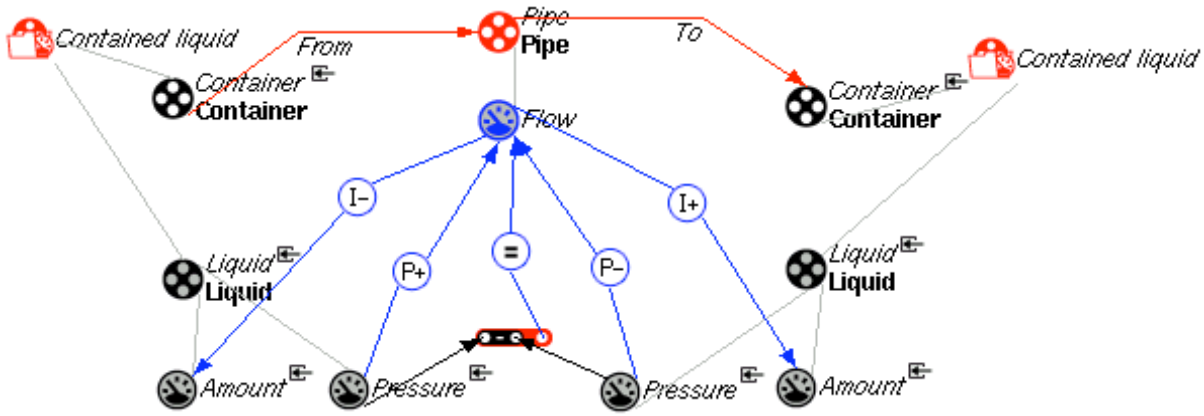


Figure 4.32: The liquid flow model fragment.

4.3.6 Simulation results

The state graph generated by simulating the scenario where the *Height* on the left is higher than on the right is shown in Figure 4.33. Some of the underlying details are shown in the quantity value history in Figure 4.34. The first state in derived from the scenario. There is a positive decreasing *Flow*, which decreases the *Amount* (and therefore *Height*) of *Oil* on the left side, and increases the *Amount* of *Oil* on the right side. The transition to state three happens when the *Container* on the right does not become full. The *Pressures* become equal, which causes the *Flow* to stop, and the changes in *Amount* will become stable.

Another possibility is that the *Height* on the right side becomes maximal (because the *Height* of the right container is smaller than the *Height* of the left one). In that case either the *Height* of the *Fluid* in the right *Container* becomes maximal and starts overflowing until the *Pressure* become equal (path [1,4,2]), or the *Height* on the right side becomes maximal and is instantly equal to the *Height* on the left side (path [1,2]).

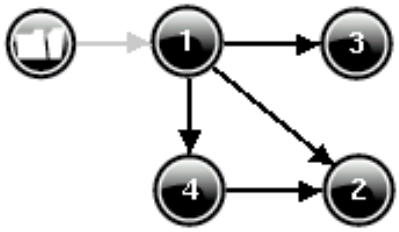


Figure 4.33: Communicating vessels simulation of scenario left higher than right.

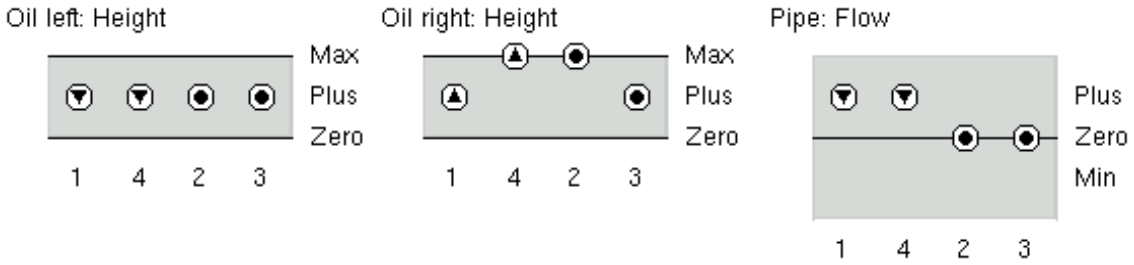


Figure 4.34: Communicating vessels value history of scenario left higher than right.

In all the states each of the model fragments fire. In figure 4.35 the situation of the system in state 4 is shown with the dependencies between them.

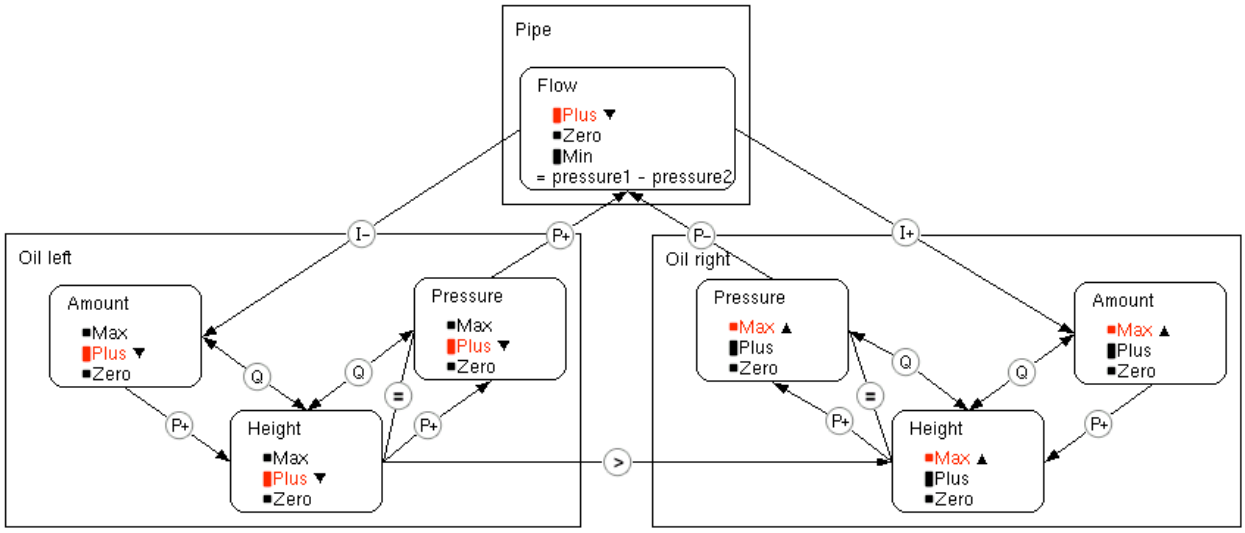


Figure 4.35: The dependencies in state 4 of the communicating vessels model.

4.3.7 Summary tables

Table 4.32: Entity summary

Entity	Super type	Description
Substance	Entity	Something that has mass and occupies space, i.e. matter.
Liquid	Substance	A substance which is in the fluid phase.
Water	Liquid	H_20 in the liquid phase.
Oil	Liquid	Oil in the liquid phase.

Object	Entity	A thing which is not purely a substance.
Pipe	Object	The object connecting two fluid containers.
Container	Object	The object which can contains fluid.

Table 4.33: Configuration summary

Configuration	**Entity** *(from)*	**Entity** *(to)*	**Description**
Connected	Container	Pipe	An undirected connection from the containers to the pipe.
Contains	Container	Fluid	A container can have fluid within it.
From	Container	Pipe	A directional connection from the container to the pipe.
To	Pipe	Container	A directional connection from the pipe to the container.

Table 4.34: Quantity summary

Quantity	**Entity/Agent**	**Quantity space**	**Description**
Amount	Fluid	{zero, plus max}	Indicated how much fluid is in the container
Height	Fluid	{zero, plus max}	Indicates how high the fluid is in the container
Pressure	Fluid	{zero, plus max}	Indicates the pressure at the bottom of the container.
Flow	Pipe	{min, zero, plus}	Indicates if there is a flow, and the direction.

Table 4.35: Scenario summary

Scenario name	Both tanks partially filled but left is higher
Initial values	height{left fluid} = plus, height{right fluid} = plus
Initial equations	Left height greater than right height.
Description	There are two containers (left and right) filled with oil. Each of the oil instances has a quantity height, of which the magnitude is set to plus. The left container is connected to the pipe via a from relation. The pipe is connected to the right container using a to relation. The left height is set to be greater than the right height.

Table 4.36: Scenario summary

Scenario name	Both tanks partially filled with equal heights
Initial values	height{left fluid} = plus, height{right fluid} = plus
Initial equations	Left height greater than right height.
Description	There are two containers (left and right) filled with oil. Each of the oil instances has a quantity height, of which the magnitude is set to plus. The left container is connected to the pipe via a from relation. The pipe is connected to the container using a to relation. Finally the left height is set to be equal to the right height.

Table 4.37: Scenario summary

Scenario name	Both tanks partially filled but right is higher
Initial values	height{left fluid} = plus, height{right fluid} = plus
Initial equations	Left height greater than right height.
Description	There are two containers (left and right) filled with oil. Each of the oil instances has a quantity height, of which the magnitude

	is set to plus. The left container is connected to the pipe via a from relation. The pipe is connected to the container using a to relation. Finally the right height is set to be greater than the left height.

Table 4.38: Model fragment summary

Model fragment name	Contained Liquid
Model fragment type	Static fragment
Model fragment parent	None
Description	A container containing liquid is modelled using conditional model ingredients. The quantities amount, height, and pressure are introduced. There are positive proportionalities between amount and height, and between height and pressure. Furthermore, there are quantity space correspondences between the quantity spaces of the magnitudes of amount and height, and height and pressure. Finally the magnitude of height is set to be equal to the magnitude of pressure.

Model fragment name	Liquid Flow
Model fragment type	Process fragment
Model fragment parent	None
Description	Two instances of the contained liquid model fragment are imported in liquid flow. The left container is connected using a from relation to the pipe. The pipe is connected using to relation to the right container. The relations and the pipe are all conditional elements. The pipe introduces a quantity flow, which is calculated by subtracting the right pressure from the left pressure (and assigning the result to flow). Flow has a positive influence on the right amount and a negative influence on the left amount. There is a positive proportionality from the left pressure to flow, and a negative proportionality from the right pressure to flow.

Table 4.40: Simulation summary

Scenario	Both tanks partially filled but left is higher
Full simulation	4
Begin state(s)	[1]
End state(s)	[3], [2]
Behaviour 1	[1 → 3]
Behaviour 1 description	The height in the left container is greater than the right height. There is a positive flow which is decreasing. The left height is decreasing, while the right is increasing. In the next state the heights become equal and the quantities become stable.
Behaviour 2	[1 → 2]
Behaviour 2 description	The height in the left container is greater than the right height. There is a positive flow which is decreasing. The left height is decreasing, while the right is increasing. In the next state the right height becomes maximal, and all quantities become stable, because the heights become equal.
Behaviour 3	[1 → 4 → 2]

Behaviour 3 description	The height in the left container is greater than the right height. There is a positive flow which is decreasing. The left height is decreasing, while the right is increasing. In the next state the right height becomes maximal, but the quantities keep changing. In the final state all the quantities are stable, because the pressures become equal.
Overall description	All the behaviours show a flow from left to right equalising the pressures.

Scenario	Both tanks partially filled but right is higher
Total nr. of states	4
Begin state(s)	[1]
End state(s)	[3], [2]
Behaviour 1	[1 → 3]
Behaviour 1 description	The height in the right container is greater than the left height. There is a negative flow which is increasing. The left height is increasing, while the right is decreasing. In the next state the heights become equal and the quantities become stable.
Behaviour 2	[1 → 2]
Behaviour 2 description	The height in the right container is greater than the left height. There is a negative flow which is increasing. The left height is increasing, while the right is decreasing. In the next state the left height becomes maximal, and all quantities become stable, because the heights become equal.
Behaviour 3	[1 → 4 → 2]
Behaviour 3 description	The height in the right container is greater than the left height. There is a negative flow which is increasing. The left height is increasing, while the right is decreasing. In the next state the left height becomes maximal, but the quantities keep changing. In the final state all the quantities are stable, because the pressures become equal.
Overall description	All the behaviours show a flow from right to left equalising the pressures.

Scenario	Both tanks partially filled with equal heights
Total nr. of states	4
Begin state(s)	[1]
End state(s)	[1]
Behaviour 1	[1]
Behaviour 1 description	The state graph shows a stable situation in which the two heights are equal and there is no flow.

4.4 Communicating Vessels (Version 2)

In this chapter the communicating vessel model from the previous chapter is further advanced. The goal is to adapt the model in such a way that it generates a *full envisionment*. A full envisionment is a state-graph that contains all the possible situations (states) which are plausible for the system, and the possible transitions between them. This is done by creating a model fragment for each possible value of the *Height* quantity, and for each possible in/equality between the *Pressures* of the two liquid columns. Furthermore, in the scenarios the maximum heights of the liquids are

made equal. This makes it impossible for containers to overflow, thereby making the state graph less complex by removing states.

4.4.1 Entity, Agent and Assumption Hierarchy

Equivalent to the first communicating vessels model.

4.4.2 Quantities and quantity spaces

Equivalent to the first communicating vessels model.

4.4.3 Scenarios

In this second version of the communicating vessels system two new scenarios are added. In the first, *unknown inequalities* (Figure 4.36), the values of the *Heights* are set to *plus*, but no inequality is set between the two *Heights*. This means they could be equal, or one could be greater than the other. Furthermore, the *max* value of the *Heights* are indicated to be equal. As a result it becomes impossible for a container to overflow without liquid being added by an exogenous influence.

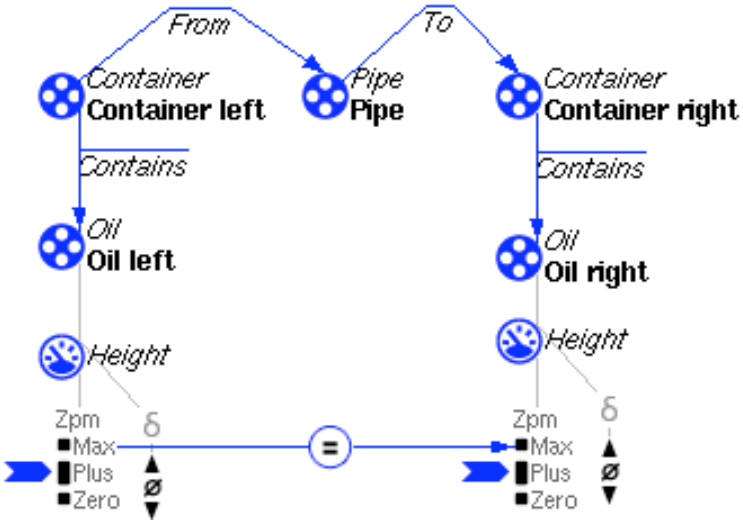


Figure 4.36: Scenario unknown inequalities in which the values are known, but the difference between the heights is not.

The second scenario, *unknown values* (Figure 4.37), allows more interpretations as the values of the Heights are not assigned. The rest of the scenario is the same as the *unknown inequalities* scenario.

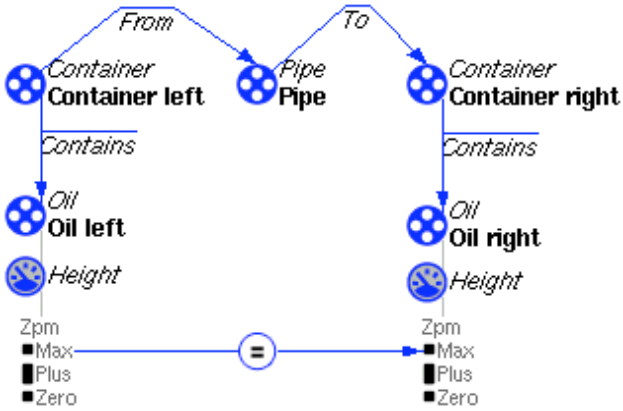


Figure 4.37: Scenario Unknown values in which both the values of the heights, and the difference between them, are unknown.

4.4.4 Model fragments

In order to have the simulator generate all the possible value combinations, a model fragment is needed for each possible value. In Figure 4.38 one of these model fragments is shown. The model fragment is a subtype of the contained liquid model fragment. The only information added, is the condition that the value of the *Height* should be *zero*. In the same way a model fragment is created in which the *Height* should be *plus (Partially filled)* and one in which the *Height* should be *max (Full)*. Conditional values and inequalities are assumed to be true by the qualitative simulator (when they are not inconsistent with each other).

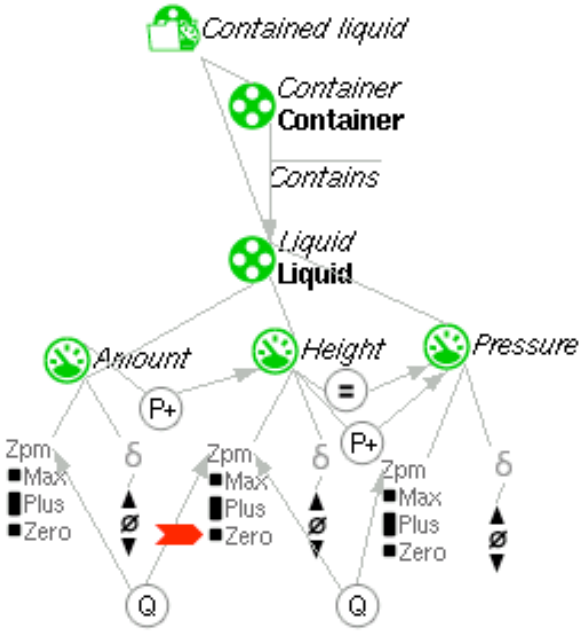


Figure 4.38: Empty container model fragment, a child of contained liquid, in which the height must be zero.

In order for the simulator to be able to derive the *Flow*, it should generate all the possible differences between the left and the right *pressure.* This is realised by creating three children of the *Liquid flow* model fragment. In each of these children model fragments a different conditional in/equality relation is specified between the *Pressures*. Figure 4.39 shows the model fragment *Left smaller than right*, which adds a smaller than relation from the left *Pressure* to the right *Pressure.* There are two more children: *Left equal to right,* in which the *pressures* are equal, and *Left greater than right*, in which the left *Pressure* is greater than the right Pressure.

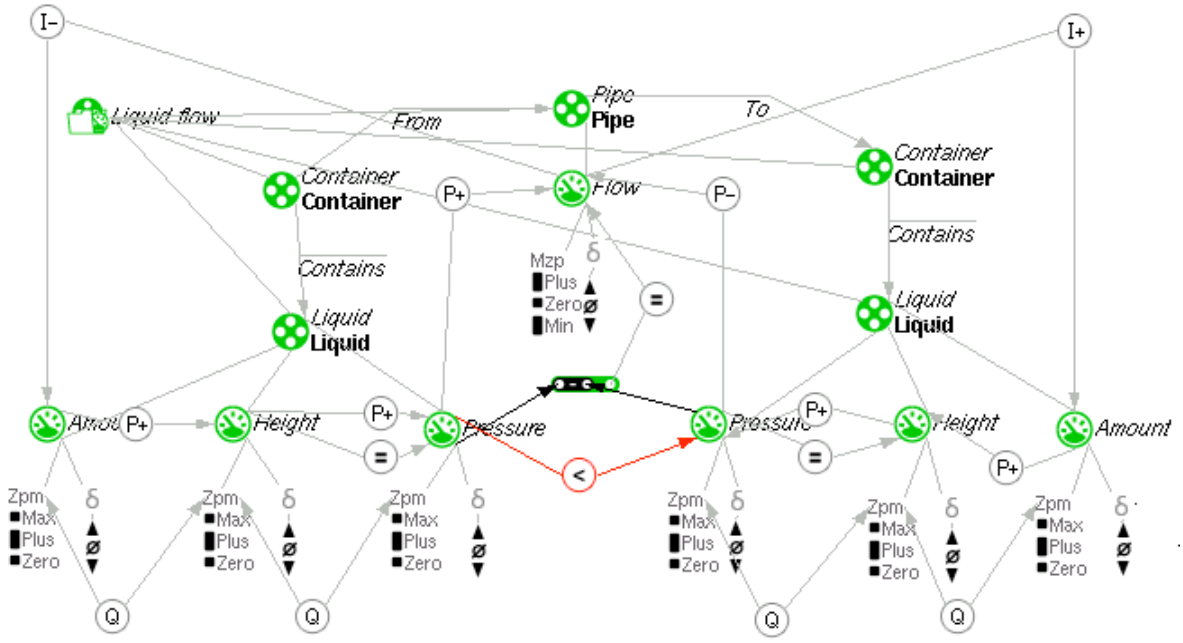


Figure 4.39: Left smaller than right: A child of the liquid flow model fragment in which the left pressure has to be smaller than the right pressure.

4.4.5 Simulation results

Simulating *Unknown inequalities* generates the state graph shown in Figure 4.40. The corresponding quantity value history is shown in Figure 4.41. In states 1 and 3 the Heights of the *Oil* are in the same interval, but in the former the left *Height* is higher, while in the latter the right *Height* is higher, as can be seen in the equation history (Figure 4.42). Both state 1 and 3 proceed to state 2, in which the pressures become equal and the flow becomes stable.

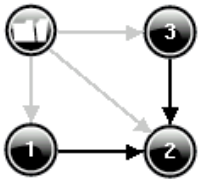


Figure 4.40: The state graph generated by simulating unknown inequalities.

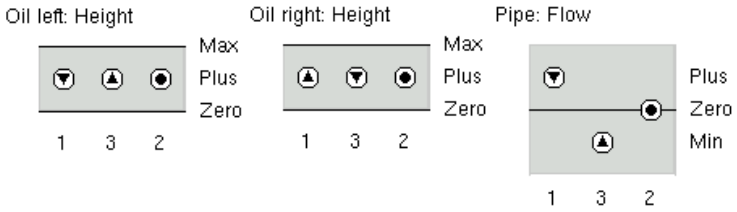


Figure 4.41: The quantity history belonging to the *Unknown inequalities* state graph.

Pressure (Oil left) ? Pressure (Oil right)

> < =

1 3 2

Figure 4.42: The equation history belonging to the unknown inequalities state graph.

The state graph resulting from simulating *Unknown values* is more complex than the *Unknown inequalities*, as the *Heights* can also be *zero* or *max* (See Figure 4.43). States 1, 6, and 11 are end states of which the values can be seen in Figure 4.44. In state 1 there is no *Oil* in either *Container*, in state 11 both the containers are completely filled, and in state 6 there both containers are partially filled and have an equal *Height*.

Similar to the *Unknown inequalities* simulation, every state is a possible start state. States 1 and 11 are special, since they are both start and end states, and not connected to the other states. In the former both *Heights* are *zero*, while in the latter both *Heights* are *max*. In states 2, 3, and 8 the right *Height* is greater than the left *Height*, and they have a different qualitative value. In state 7 the qualitative values become equal, but the right *Height* remains higher. Finally, in state 6 the *Heights* become equal. In states 4, 9, and 10 the left *Height* is higher, and there is a difference in qualitative values with the right *Height*. In state 5 the qualitative values become the same, but the *Height* difference remains. Finally, in state 6 the *Heights* become equal.

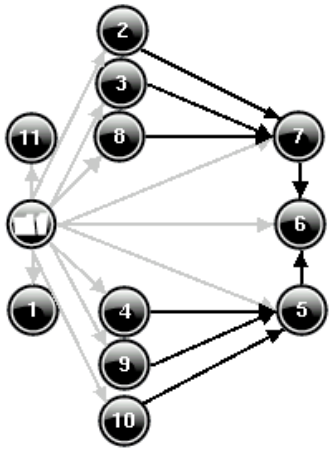


Figure 4.43: The state graph generated by simulating *unknown values.*

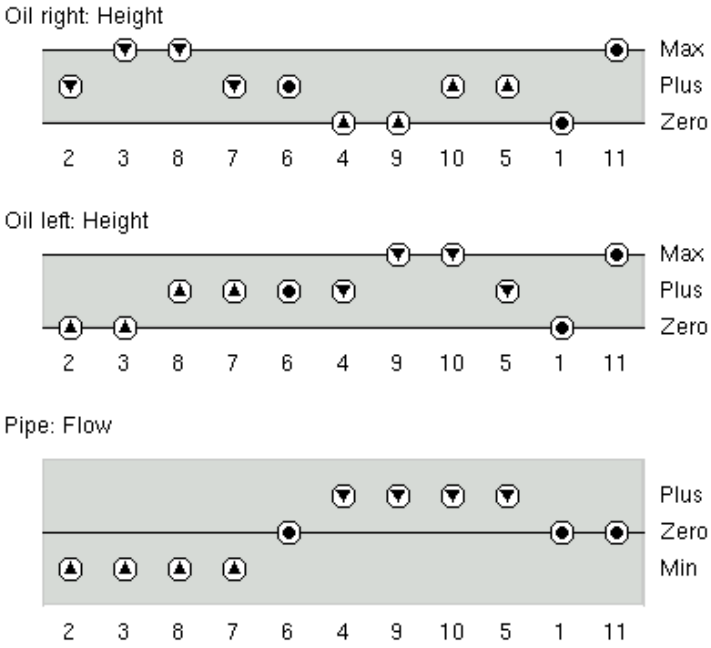


Figure 4.44: The quantity value history belonging to the *unknown values* state graph.

(Note that for both simulations the dependencies remain the same).

4.4.6 Summary tables

The entity, agent, assumption, attribute, configuration, quantity and quantity space definitions remain exactly the same as the first communicating vessel system.

Table 4.43: Scenario summary

Scenario name	Unknown inequalities
Initial values	Height{left oil} = plus, height{right oil} = plus
Initial equations	None.
Description	The scenario is the almost the same as in the first communicating vessel models. The difference is that there is no inequality specified between the heights. The maximum heights of the containers are equal.

Table 4.44: Scenario summary

Scenario name	Unknown values
Initial values	None.
Initial equations	None.
Description	The scenario is the almost the same as the scenarios in the first communicating vessel models. The difference is that there is no inequality specified between the heights, and no values are assigned to the heights. The maximum heights of the containers are equal.

Table 4.45: Model fragment summary

Model fragment name	Empty / Partially Full / Full
Model fragment type	Static fragment
Model fragment parent	None
Description	These are three model fragments which are children of *Contained liquid*. In the first, the height has to be *zero,* in the second *plus*, and in the final one *max*.

Model fragment name	Left greater than right / Left equal to right / Left smaller than right.
Model fragment type	Process fragment
Model fragment parent	None
Description	These three model fragments are children of the liquid flow model fragment. In the first the pressure left has to be greater than the pressure left, in the second they have to be equal, and in the final on the left pressure has to be smaller than the right pressure.

Table 4.46: Simulation summary

Scenario	Unknown inequalities
Full simulation	3
Begin state(s)	[1], [2], [3]
End state(s)	[2]
Behaviour 1	[1 → 2]
Behaviour 1 description	Both heights are plus, but the left one is higher. In the second state the heights become equal.
Behaviour 2	[3 → 2]
Behaviour 2 description	Both heights are plus, but the right one is higher. In the second state the heights become equal.
Overall description	There is a height difference which disappears due to a liquid flow, or the heights are equal.

Scenario	Unknown values
Full simulation	11
Begin state(s)	[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]
End state(s)	[1], [6], [11]
Behaviour 1	[2,3,6 → 7 → 6]
Behaviour 1 description	In the first set of states the right height is greater than the left one and they have a different qualitative value. In state 7 the heights both have the value plus, but the height difference

	remains. In state 6 the heights become equal.
Behaviour 2	[4,9,10 → 5 → 6]
Behaviour 2 description	In the first set of states the left height is greater than the right one and they have a different qualitative value. In state 5 the heights both have the value plus, but the height difference remains. In state 6 the heights become equal.
Behaviour 3	[1] or [11]
Behaviour 3 description	Both values are either max, or zero. Since the heights are equal, there is no flow.
Overall description	There is a height difference which disappears due to a liquid flow, or the heights are equal. Note that states 5, 6 and 7 correspond to the states in the Unknown inequalities simulation.

4.5 Heating & Boiling

The heating and boiling model describes the heating of a container with a liquid by a stove. The stove increases the heat of the substance, which in turn increases its temperature. When the liquid reaches its boiling point, the boiling process becomes active, causing the liquid too evaporate into gas. When the entire substance has evaporated, its temperature can increase above the boiling point.

4.5.1 Entity, Agent and Assumption Hierarchy

The key entities are the *Substance* that is heated, and the *Container* in which it is contained (Figure 4.45). The *Container* is a subtype of the object *Entity*. The possible *Energy sources*, a *Stove* and the *Sun*, are modelled as *Agents* (Figure 4.46). This heating and boiling model has no assumptions.

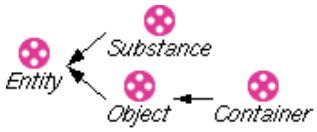


Figure 4.45: The entity hierarchy of the heating and boiling model.

Figure 4.46: The agent hierarchy of the heating and boiling model.

4.5.2 Quantities and quantity spaces

The quantities and quantity spaces of the heating and boiling model are shown in Table 4.46. The *Amount of liquid* within a *Container* is either *zero*, meaning it is empty, *max,* indicating that it is completely filled, or somewhere in between, *plus*. Given this quantity space for the *Amount of liquid*, it is good to choose the same quantity space for the quantity *Amount of gas*. The value *max* indicates the *Amount of gas* given the maximum *Amount of liquid* that fits in the container. There is a *Heat flow* between the *Energy source* and the *Container*, which in principle could be in either direction. For this reason the quantity space *{min, zero, plus}* is chosen, where *min* indicates a flow from the *Container* to the *Energy source*, and *plus* a flow in the other direction. The quantity *Heat* indicates the amount of energy of the substance. This could either be no energy (if the substance does not exist): *zero*, or some positive amount of energy: *plus*. Finally, a quantity space has to be chosen for the *Temperature* of a substance. *Temperature* has relatively quite a lot of landmarks. The lowest value is the *absolute nil*, in which case particles do not move anymore. The second is the *freeze melt* point, in which case a solid substance transforms into a liquid substance. In between these two points is the *solid phase*. The third landmark is the *condense boil* point in which a liquid transforms

into a gas. Between *freeze melt* point and the *condense boil* point is the *liquid phase*. Finally, above the *condense boil* point is the *gas phase*, which is theoretically unbound.

Table 4.46: The quantities and quantity spaces for the heating and boiling model

Quantity	***Quantity Space***
Amount of gas	*{zero, plus, max}*
Amount of liquid	*{zero, plus, max}*
Flow	*{min, zero, plus}*
Heat	*{zero, plus}*
Temperature	*{Absnil, Solid phase, Freeze melt, Liquid phase, Condense boil, Gas phase}*

4.5.3 Scenarios

The scenario *Boiling water* (Figure 4.47) models a *Container*, which has the value *open* for the attribute *openness*, and contains an *H_20* substance. H_20 has the quantities: *Amount of gas*, *Amount of liquid*, and *Temperature*, which have the initial values: *zero, max,* and *liquid phase* consecutively. Furthermore, the container is *on top of* a *Stove*, which has the value *on* for the attribute *heater status.*

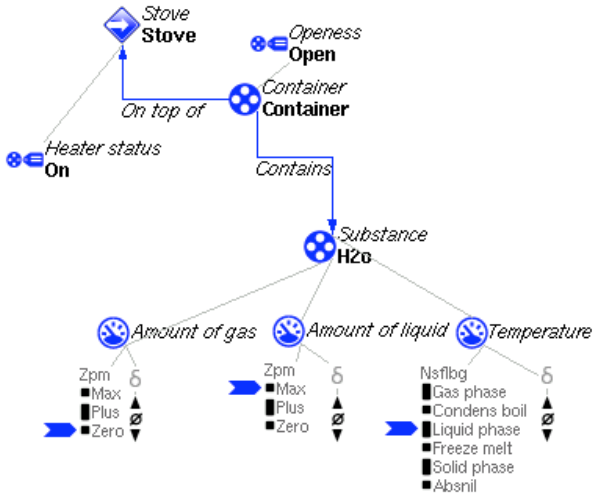


Figure 4.47: Heated water describes water in the liquid phase being heated by a stove.

4.5.4 Model fragments

The *Substance* model fragment is shown in Figure 4.48. The conditional *Substance* entity introduces the quantities: *Amount of gas*, *Amount of liquid*, *Heat*, and *Temperature*. There is a positive proportionality from *Heat* to *Temperature* - when the *Temperature* increases, the Heat energy increases, and if the Heat remains stable, the *Temperature* is stable too.

Figure 4.48: The substance model fragment.

The *Substance* model fragment has three children: one for the *Gas phase*, one for *Liquid phase,* and one for *Solid*. These model fragments are variations on a theme, therefore only the *liquid phase* fragment in shown in Figure 4.49. For the *Substance* to be in the *liquid phase*, the *Amount of liquid* has to be greater than *zero*, and the *Temperature* has to be greater or equal to the *freeze melt* point and *smaller or equal* to the *condense boil* point. For the *Substance* to be in the *gas phase*, the *Amount of gas* has to be greater than *zero*, and the *Temperature* has to be greater or equal to the *condense boil* point. For the *Substance* to be in the solid phase, the *Temperature* has to be smaller or equal to the *freeze melt* point.

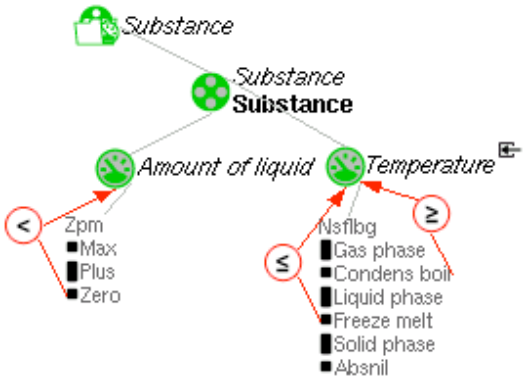


Figure 4.49: The liquid phase model fragment, which is a child of the substance model fragment.

The *container with substance* model fragment (Figure 4.50) imports the *substance* model fragment. It models a *Container* with *contains* the *Substance*.

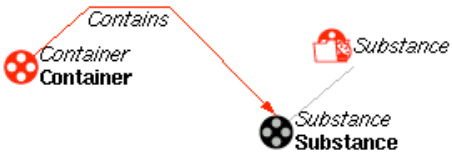


Figure 4.50: The container with substance model fragment.

The *container with substance* has four children. The first child models an *Empty container*, the second a *Container with a mixture of liquid and gas*, the third a *Container with only gas*, and finally the fourth a *Container with only liquid*. Again these model fragments are variations on a theme, therefore only the *Container with liquid and gas mixture* is shown (Figure 4.51). For the container to contain a mixture the *Amount of gas,* and the *Amount of liquid* have to be greater than *zero*. As a consequence, the *Heat* has to be greater than *zero* too, as substances always have energy. There is a value correspondence between the plus values of *Amount of gas* and *Amount of liquid,* as the maximum gas value represents the amount of gas when all the liquid has evaporated. Therefore if either gas or liquid is in the *plus* interval, the other must be in that interval too.

For the container to be empty, both the *Amount of gas*, and the *Amount of liquid* have to be *zero.* There is a value correspondence between the value *zero* of the *Amount of liquid* and the *zero* of the *Heat,* as non-existent substances have no energy. For the container to contain only gas, *the Amount of gas* has to be greater than *zero*, and the

Amount of liquid has to be zero. As a consequence the *Heat* has to be greater than *zero,* as a substance always has energy. For the container to contain only liquid, the *Amount of gas* has to be *zero*, and the *Amount of liquid* has to be greater than *zero*. As a consequence the *Heat* has to be greater than *zero,* as a substance always has energy.

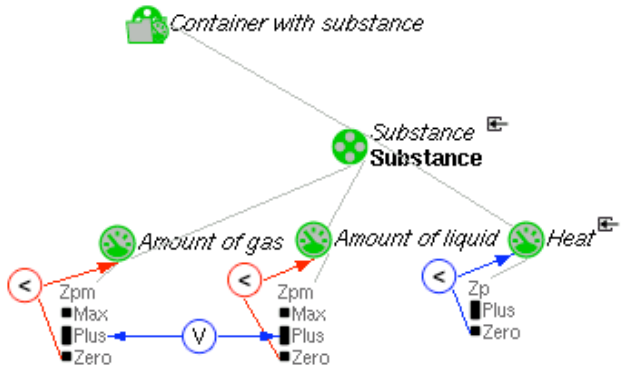


Figure 4.51: The container with liquid and gas mixture model fragment. This model fragment is a child of the container with substance model fragment.

The *Steady heater* model fragment (Figure 4.52) imports the *Container with substance* model fragment. It models a *Stove*, which has the value *on* for the *heater status* attribute, indicating that it is active. Furthermore, it models that the *Container* is *on top of* the *Stove*. As a consequence, it introduces the quantity *Flow* (related to the stove), which positively influences the *Heat* of the *Substance*. This indicates that the stove increases the *Heat* of the *Substance*. The magnitude of the *Flow* is set to *plus,* indicating that there is always a heat flow from the *Stove* to the *Substance,* and the derivative is set to *stable* (both as consequences), indicating that the flow does not change.

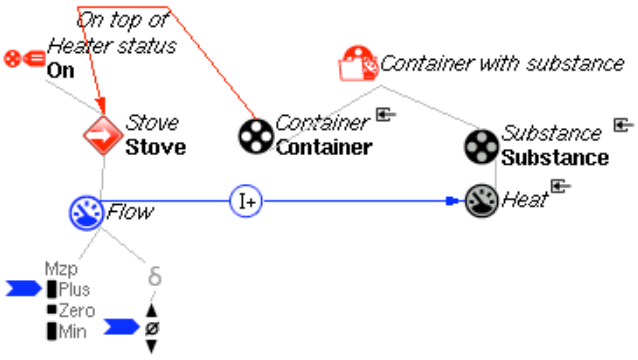


Figure 4.52: The steady heater agent model fragment.

The *Boiling* model fragment (Figure 4.53) imports the *Steady heater* model fragment. If the *Amount of liquid* is greater than *zero*, and the *Temperature* is on the *condense point*, the model fragment applies. It introduces a positive proportionality from *Amount of liquid* to *Temperature* (condensation causes the *Temperature* to decrease); a negative proportionality from *Amount of gas* to *Temperature* (evaporation causes the *Temperature* to decrease)*;* a positive proportionality from *Heat* to *Amount of gas* (boiling causes gas to appear)*;* and a negative proportionality from *Heat* to *Amount of liquid* (boiling causes liquid to disappear). Finally it sets the *Temperature* to stable (the *Temperature* remains at *condense point* while boiling).

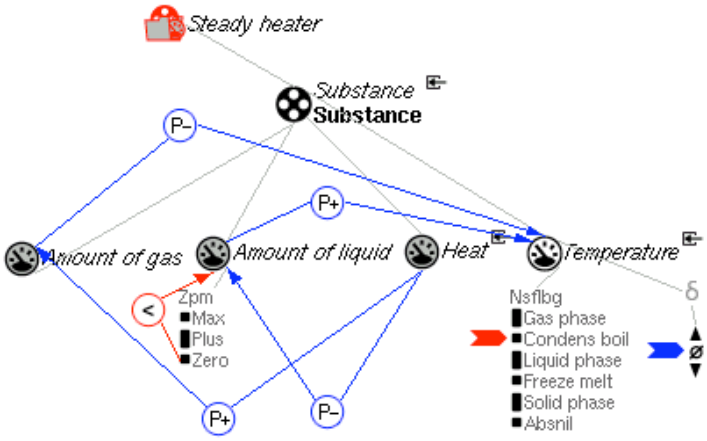


Figure 4.53: The *boiling* process fragment, importing the *steady heater* model fragment.

Another (slightly more complex) approach would be to include pressure as an intermediate quantity between heat and temperature and have two heat quantities, one for gas and one for liquid. In that case the energy of the liquid would decrease during evaporation and the heat of the gas increase. The individual heats would affect the substance temperature, which would become ambiguous, but be kept stable while boiling.

4.5.5 Simulation results

The state graph, shown in Figure 4.54, shows the result of simulating the scenario *heated water*. The corresponding value history is shown in Figure 4.55. As expected, the Temperature increases to the boiling point (state 2), and causes the liquid to evaporate. This decreases the *Amount of liquid*, and increases the *Amount of gas* (state 2 and 3), until all the water has completely evaporated (state 4), and the Temperature can increase beyond the *condense boil* point (state 5).

Figure 4.54: The state graph generated by simulating boiling water.

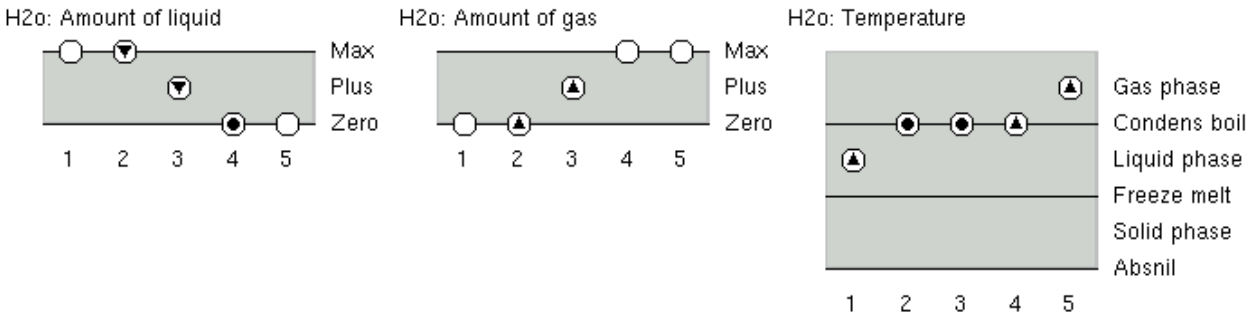


Figure 4.55: The quantity value history generated by simulating boiling water.

4.5.6 Summary tables

Table 4.47: Entity summary

Entity	Super type	Description
Substance	Entity	Something that has mass and occupies space, i.e. matter.
Object	Entity	A thing that is not purely a substance.
Container	Object	The object that can contains fluid.

Table 4.48: Agent summary

Entity	Super type	Description
Energy source	Agent	Something that provides energy.
Stove	Energy source	An apparatus in which electricity or a fuel is used to generate heat, as for cooking or warmth.
Sun	Energy source	A star that is the centre of our planetary system.

Table 4.49: Attribute summary

Attribute	Entity/Agent	Values	Description
Openness	Container	Open/Closed	A container can either be open or closed.
Heater status	Stove	On/Off	The energy source is either active or inactive.

Table 4.50: Configuration summary

Configuration	Entity *(from)*	Entity *(to)*	Description
Contains	Container	Substance	A container can have a substance in it.
On top of	Container	Stove	The container can be placed on top of the stove.

Table 4.51: Quantity summary

Quantity	Entity/Agent	Quantity space	Description
Amount of gas	Substance	{zero, plus max}	Indicates how much of the substance is in the gas phase.
Amount of liquid	Substance	{zero, plus max}	Indicates how much of the substance is in the liquid phase.
Flow	Heat source	{min, zero, plus}	Indicates if there is a flow, and the direction.
Heat	Substance/Stove	{zero, plus}	A form of energy associated with the motion of atoms or molecules.
Temperature	Substance	{absnil, solid phase, freeze melt, liquid phase, condens boil, gas phase}	The degree of hotness or coldness of a substance.

Table 4.52: Scenario summary

Scenario name	Boiling water
Initial values	Amount of gas{H_20} = zero, amount of liquid{H_20} = max, temperature{H_20} = liquid phase
Initial equations	None
Description	A container, which has the value open for the attribute openness,

	contains a H_20 substance. H_20 introduces the quantities amount of gas, amount of liquid, and temperature, which have the initial values mentioned above. Furthermore, the container is on top of a stove, which has the value on for the attribute heater status.

Table 4.53: Model fragment summary

Model fragment name	Substance
Model fragment type	Static fragment
Model fragment parent	None
Description	The conditional substance entity introduces the quantities amount of gas, amount of liquid, heat, and temperature. There is a positive proportionality from heat to temperature.

Model fragment name	Gas phase
Model fragment type	Static fragment
Model fragment parent	Substance
Description	For the substance to be in the gas phase, the amount of gas has to be greater than zero, and the temperature has to be greater or equal to the condense boil point.

Model fragment name	Liquid phase
Model fragment type	Static fragment
Model fragment parent	Substance
Description	For the substance to be in the liquid phase, the amount of liquid has to be greater than zero, and the temperature has to be greater or equal to the freeze melt point and smaller or equal to the condense boil point.

Model fragment name	Solid phase
Model fragment type	Static fragment
Model fragment parent	Substance
Description	For the substance to be in the solid phase, the temperature has to be smaller or equal to the freeze melt point.

Model fragment name	Container with substance
Model fragment type	Static fragment
Model fragment parent	None.
Description	

Model fragment name	Container empty
Model fragment type	Static fragment
Model fragment parent	Container with substance
Description	For the container to be empty, both the amount of gas, and the amount of liquid have to be zero. There is a value correspondence between the value zero of the amount of liquid and the zero of the heat.

Model fragment name	Container with liquid and gas mixture
Model fragment type	Static fragment

Model fragment parent	Container with substance
Description	For the container to contain a mixture of gas and liquid, the amount of gas, and the amount of liquid have to be greater than zero. As a consequence, the heat has to be greater than zero too. There is value correspondence between the plus values of amount of gas and amount of liquid.

Model fragment name	Container with only gas
Model fragment type	Static fragment
Model fragment parent	Container with substance
Description	For the container to contain only gas, the amount of gas has to be greater than zero, and the amount of liquid has to be zero. As a consequence the amount of heat has to be greater than zero, as a substance always has energy.

Model fragment name	Container with only liquid
Model fragment type	Static fragment
Model fragment parent	Container with substance
Description	For the container to contain only liquid, the amount of gas has to be zero, and the amount of liquid has to be greater than zero. As a consequence the heat has to be greater than zero.

Model fragment name	Steady heater
Model fragment type	Agent fragment
Model fragment parent	None
Description	The steady heater model fragment imports the container with substance model fragment. It models a stove, which has the attribute value on for the heater status attribute. Furthermore it models that the container is on top of the stove. As consequence, it introduces the quantity flow (related to the stove), which positively influences the heat of the substance. The magnitude of the flow is set to plus, and the derivative to stable (both as consequences).

Model fragment name	Boiling
Model fragment type	Process fragment
Model fragment parent	None
Description	The boiling model fragment imports the steady heater model fragment. If the amount of liquid is greater than zero, and the temperature is on the condense point, the model fragment applies. It introduces a positive proportionality from amount of liquid to temperature; a negative proportionality from amount of gas to temperature; a positive proportionality from heat to amount of gas; and a negative proportionality from heat to amount of liquid. Finally it sets the temperature to stable.

Table 4.64: Simulation summary

Scenario	Heated water
Full simulation	5

Begin state(s)	[1]
End state(s)	[5]
Behaviour 1	[1 →2 → 3 →4 → 5]
Behaviour 1 description	In state 1, the temperature starts increasing. In state 2, it reaches the boiling point, and start decreasing the amount of liquid, and increasing the amount of gas. In state 3, the amount of liquid, and amount of gas both become plus. In state 4, the amount of gas becomes max, and the amount of liquid becomes zero. Finally, in state 5, the temperature becomes greater than the boiling point.

4.6 Heating & Boiling (Version 2)

In this section the heating and boiling model from the previous chapter is modelled differently. Instead of viewing the heat source as an exogenous influence (modelled as an agent) the heater is now part of the system and may exchange heat with other objects when it is hotter than those objects. So, in this model the heat flow depends on the heat difference between the stove and the container.

4.6.1 Entity and agent hierarchy

In the previous version of the model we consider the stove to be an integral part of the system. Therefore the stove and the heat source are integrated in the entity hierarchy. The model has no agents.

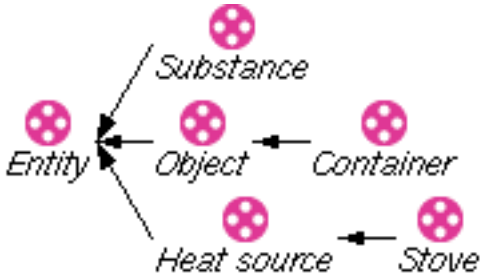


Figure 4.56: The entity hierarchy including the heat source and the stove.

4.6.2 Assumption hierarchy

The previous version of the model there was always a heat flow from the stove to the contained substance. In this model we use an assumption: *extreme heater*. This assumption will be used to model that the temperature of the stove is always greater than the temperature of the substance.

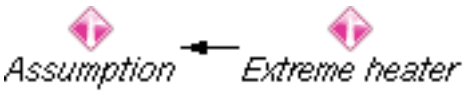


Figure 4.57: The assumption hierarchy.

4.6.3 Quantities and quantity spaces

Equivalent to the first heating and boiling model.

4.6.4 Scenarios

This model has two scenarios. *Boiling water* and *boiling water with extreme heater.* The model, presented in Figure 4.58, shows a *Container*, which has the value *open* for the

attribute *openness*, and *contains* a *H_20* substance. *H_20* has the quantities *Amount of gas* (value *gas phase*), *Amount of liquid* (value *max*), and *Temperature* (value *liquid phase*). Furthermore, the *Container* is on top of a *Stove*, which has the value *on* for the attribute *heater status.* The stove introduces a quantity *Temperature* which has the value *gas phase*. Furthermore, the *Temperature* of the *Stove* is greater than the *Temperature* of the *H_20*. Finally, this scenario has the assumption *Extreme heater*. The scenario *Boiling water* differs because this assumption is not included.

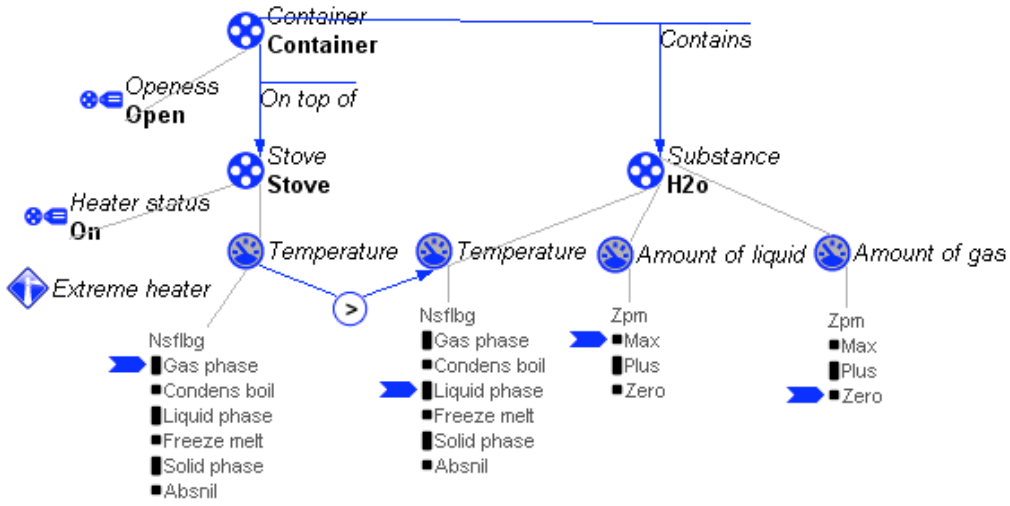


Figure 4.58: The *heated water with extreme heater* scenario.

4.6.5 Model fragments

The model fragment *Heater with steady value* is shown in Figure 4.59. It indicates that a turned on heater always has the *Temperature gas phase*. The heat source, which has to have a *heater status on*, introduces the quantities *Heat* and *Temperature.* There is a positive proportionality between *Heat* and *Temperature,* since the *Temperature* increases as the *Heat* increases. The *Temperature* has the value *gas phase* as a consequence.

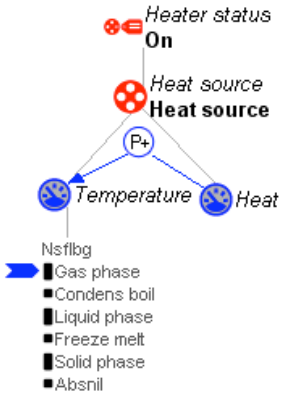


Figure 4.59: The *heated water with extreme heater* scenario.

The *Heat flow* model fragment (Figure 4.60) imports the *Heater with steady value*, and *Container with substance* model fragments. The *Container* has to be *on top of* the *Heat source*. Subtracting the *Temperature* of the *Heat source* from the *Temperature* of the *Substance* equals the *Flow*. The *Flow* positively influences the *Heat* of the Substance,

and negatively influences the *Heat* of the *Heat source*, because the heat *Flow* decreases the energy of the *Heat source*, and increases that of the *substance*. There is a positive proportionality from the *Temperature* of the *Heat source* to the *flow*, and a negative proportionality from the *Temperature* of the *Substance* to the *flow.* These proportionalities model the facts that the *Flow* decreases if the *Temperature* of the heat source decreases, and that the Flow decreases if the *Temperature* of the substance increases. Note that the model fragment *Steady heater* is replaced by the *Heat flow* model fragment.

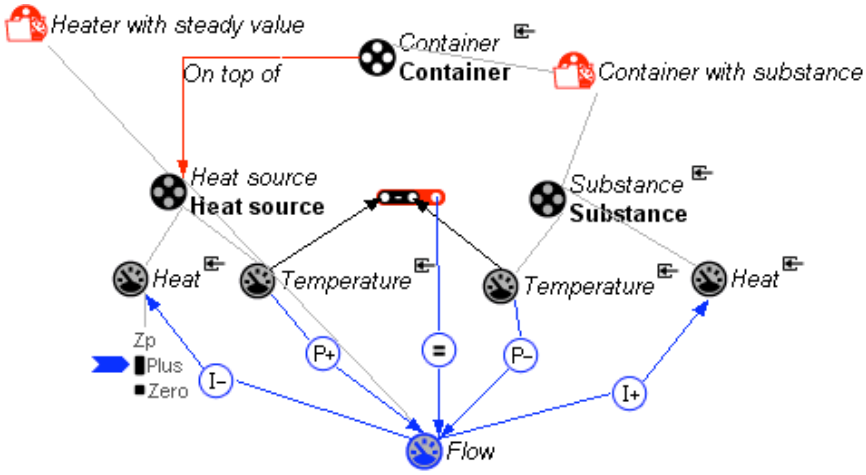


Figure 4.60: The heat flow model fragment.

The *Boiling* is the same as in the previous model, except it now imports the *Heat flow* model fragment.

The *Assume extreme heater* model fragment (Figure 4.61) imports the *Container with substance* and the *Heater with steady value* model fragments. If the assumption *Extreme heater* is true, it indicates that the *Temperature* of the *Heat source* is greater than the *Temperature* of the *Substance.* Notice that this model fragment only applies when the assumption is active, that is, has been specified in the scenario.

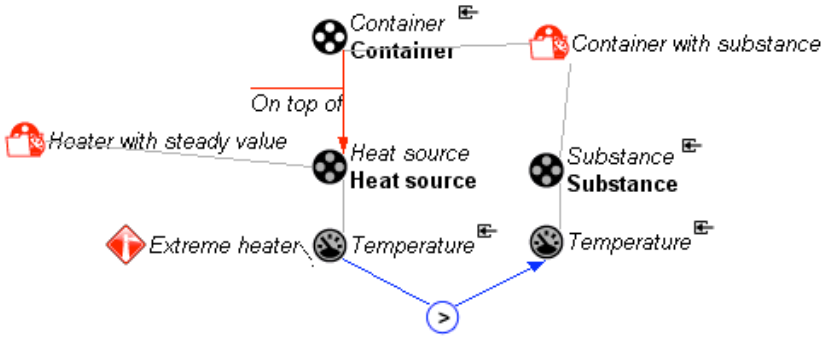


Figure 4.61: The assume extreme heater model fragment.

4.6.6 Simulation results

The simulation result of the *Boiling water with extreme heater* is equivalent to the simulation of the *Boiling water* scenario with the previous model, only the approach is different (see Section 6.2). The *Boiling water scenario* in this model gives other results,

as can be seen in Figures 62 and 63. If the *Temperature* of the *Heat source* remains greater than the *Temperature* of the *H_20*, such as in states [1], [4], [5], [8] and [9] (see Figure 4.64), the same behaviour happens as with the *Heated water with extreme heater* scenario. However, there is an exception: The two *Temperatures* become equal in state 10, stabilizing the heat flow. The difference compared to the *Heated water with extreme heater* is that it is unknown whether the *Temperature* of the *Stove* remains greater than the *Temperature* of the *Substance* (as indicated in the scenario). This happens because the values in the quantity space of the magnitude of *Temperature* of the *Stove* are not related to the values in the quantity space of the magnitude of the *Temperature* of the substance. As a result, it is possible that the *condense boil* point of the substance is equal to the *freeze melt* point of the stove. So the *Temperatures* could become equal when the *Temperature* of the substance is within the *liquid phase* (state 3), in the *condense boil* phase before any *Substance* has evaporated (state 2), in the *condense boil* phase after some *Substance* has evaporated (state 7), or in de *condense boil* phase after all the *Substance* has evaporated (state 6).

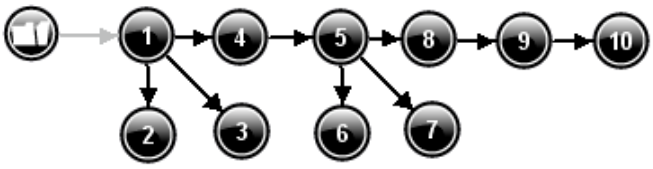


Figure 4.62: The state graph generated by simulating boiling water.

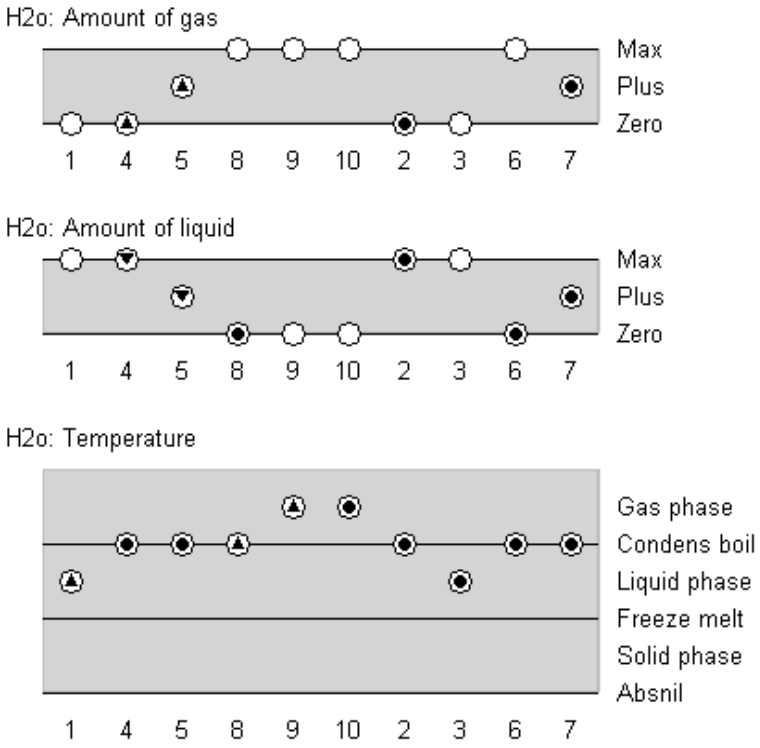


Figure 4.63: The quantity value history generated by simulating boiling water.

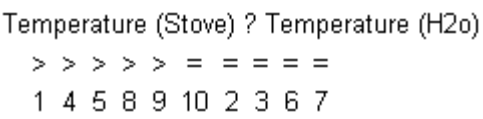


Figure 4.64: The transition history generated by simulating boiling water.

4.6.7 Summary tables

Table 4.65: Scenario summary

Scenario name	Heated water
Initial values	Amount of gas{H_20} = zero, amount of liquid{H_20} = max, temperature{H_20} = liquid phase, temperature{Stove} = gas phase
Initial equations	None
Description	A container, which has the value open for the attribute openness, contains a H_20 substance. H_20 introduces the quantities amount of gas, amount of liquid, and temperature, which have the initial values mentioned above. Furthermore, the container is on top of a stove, which has the value on for the attribute heater status. The stove introduces a quantity temperature that has the value gas phase. Furthermore, the temperature of the stove is greater than the temperature of the H_20.

Scenario name	Heated water with extreme heater
Initial values	Same as in Table 4.65.
Initial equations	None
Description	Same as in Table 4.65, except it has the assumption extreme heater.

Table 4.67: Model fragment summary

Model fragment name	Heater with steady value
Model fragment type	Static fragment
Model fragment parent	None
Description	The heat source, which has to have a heater status on, introduces the quantities heat and temperature. There is a positive proportionality between heat and temperature. The temperature has the value gas phase as a consequence.

Model fragment name	Heat flow
Model fragment type	Process fragment
Model fragment parent	None
Description	The heat flow model fragment imports the heater with steady value, and container with substance model fragments. The container has to be on top of the heat source. Subtracting the temperature of the heat source from the temperature of the substance equals the flow. The flow positively influences the heat of the substance, and negatively influence the heat of the heat source. There is a positive proportionality from the temperature of the heat source to the flow, and a negative proportionality from the temperature of the substance to the flow.

Model fragment name	Boiling
Model fragment type	Process fragment
Model fragment parent	None
Description	Boiling is the same as in Table 4.63, except it now imports heat flow instead of steady heater.

Model fragment name	Assume extreme heater
Model fragment type	Static fragment
Model fragment parent	None
Description	The assume extreme heater model fragment imports the container with substance and the heater with steady value model fragments. If the assumption extreme heater is true, it indicates that the temperature of the heat source is greater than the temperature of the substance.

Table 4.71: Simulation summary

Scenario	Heated water
Full simulation	10
Begin state(s)	[1]
End state(s)	[2], [3], [6], [7], [10]
Behaviour 1	[1 →4 → 5 → 8 → 9 → 10]
Behaviour 1 description	If the temperature of the heat source is remains greater than the temperature of the H_20, like in this behaviour, the same behaviour happens as with the heated water with extreme heater scenario. There is one exception: The two temperatures become equal in state 10, stabilizing the heat flow.
Overall description	The difference compared to the heated water in the first heating and boiling model is that it is unknown whether the temperature of the stove remains greater than the temperature of the liquid (as indicated in the scenario). This mean that the temperatures could become equal (stopping the process) when within the liquid phase of the substance (state 3), in the condense boil phase before any water has evaporated (state 2), in the condense boil phase after some water has evaporated (state 7), or in the condense boil phase after all the water has evaporated (state 6).

5 Examples of QR-based Ecological Modelling

This chapter presents applications of QR techniques to various ecological problems. It is not the intention to be exhaustive, but to present an overview of the possibilities QR has to offer for ecological modelling. In fact, formalizing qualitative ecological knowledge in qualitative terms is a longstanding problem in ecological modelling. May (1973) undertook a qualitative analysis of the results produced by differential equation models regarding the interactions between populations to study the relationship between complexity and stability in biological communities. May used only the signs {+, 0, –} and showed that a less complex community met the conditions for stability, while the more complex was not stable. Therefore, the 'common-sense wisdom' that more complexity means increased stability may not be true.

An approach for building qualitative models about the dynamics of communities subject to recurrent disturbance (such as fire) was proposed by Noble and Slatyer (1980). This approach is based on a small number of attributes of the plant's life history (vital attributes) that can be used to characterise the potentially dominant species in a particular community. Simulations produce a replacement sequence that depicts the shifts in composition and dominance following a disturbance. Further developments describe a simulation model that is also based on the vital attributes but is now combined with quantitative knowledge about the abundance of the populations and their survival according to the availability of environmental resources (Moore and Noble, 1990, 1993).

Câmara et al. (1987) describe SLIN, a program that supports qualitative simulations using values expressed in linguistic terms (such as low, medium, high) manipulated by a set of logical rules. SLIN was used in studies about the management of water resources of a hydropower plant and assessment of oil dispersion in the sea after a tanker accident (Antunes et al., 1987). Recently McIntosh (2003) describes a modelling language for dealing with partial and imprecise ecological knowledge. Borrowing some concepts from QR, such as the representation of quantities (including the distinction between amount and derivative, both having two value components, magnitude and sign, and a set of possible qualitative values), the author implements his ideas using a rule-based approach and presents an example about vegetation dynamics.

5.1 Population and Community Dynamics

Following a principled approach to QR Salles and Bredeweg (1997) have developed a library of model fragments that can be used to construct models and automate reasoning about the behaviour of populations (the examples presented in Chapter 4 are based on this work). This library was used to construct a model of the Cerrado Succession Hypothesis (CSH) (Salles and Bredeweg, 2003, in press). The Cerrado is the second largest Brazilian biome, a kind of savannah type of vegetation with a wide range of natural physiognomies, spanning from open grasslands to closed forests. Fire is one of the most influential determinants of this physiognomy and its influence is expressed by the follow hypothesis: if the fire-frequency increases (for example, because of human actions), then the vegetation becomes less dense, with reduction of trees and shrub populations so that grass may dominate. If, on the contrary, fire-frequency decreases, the vegetation becomes denser, with more trees and shrub and less grass. A set of model fragments defines the different physiognomies according to the proportion of three populations, tree, shrub and grass. For example, the *cerradão* is a forest defined by the maximum size of tree population and no grass. The *campo limpo* is open grassland defined by the maximum size of grass population and no trees and

shrubs. Between these two extremes, other physiognomies may have more or less of the three populations. According to the literature and Brazilian researchers, it is 'commonsense' that fire destroys the litter and, under this condition, temperature and light increase and humidity decreases. These are negative influences for the germination of trees and shrub seeds, and positive influence for the germination of grass seeds. These ideas are the basis for the causal model captured in the CSH. Simulations with the CSH model produce the behaviour predicted by the hypotheses mentioned above. Notice that, the CSH is a typical situation in which a mathematical approach is not adequate, because the ecological system is complex and numerical data about the *whole* phenomena does not exist. There is 'only' a conceptual model, which is the expert's commonsense understanding and hypothesis to explain the final result.

The work on the CSH has been the inspiration for a number of additional research efforts, among which the interactions between populations of different species. Such interactions are important for understanding the behaviour of larger communities. Salles et al. (2003a) present a set of models about interactions such as predation/parasitism, commensalism, cooperation/mutualism, amensalism, and competition. Each model produces simulation results that are characteristic for the interaction type it models. For example in the case of predation the state-graph shows four behaviours: only the prey reaching maximum size, both species stabilising at a corresponding size, both disappearing, and the predator disappearing while the prey grows to its maximum size.

The ants' garden is an interesting example of interacting species. This system, a well-known symbiosis between ants (*Formicidae*) and a fungus (*Lepiotaceae*), is more complex than initially understood. A third species, the specialized garden parasite fungi (*Escovopsis*), is often present and may destroy the system by attacking the cultivated fungi. However, it almost never happens because ants carry on their body colonies of bacteria (*Streptomyces*) that produce antibiotics specifically targeted to suppress the growth of *Escovopsis*. Traditional modelling approaches, based on differential or difference equations, are not adequate to handle this complex balance of interactions, but qualitative models can and have been made. Using the set of interacting population models Salles et al. (2003b) describe the ants' garden as follows: ants and *Lepiotaceae* fungi as mutualism; *Escovopsis* and *Lepiotaceae* fungi as parasitism; ants and bacteria as commensalism; and bacteria and *Escovopsis* fungi as amensalism. One of the typical simulations with this model produces the following four behaviours: coexistence of all the involved species, complete extinction of the garden, coexistence with *Escovopsis* but the ants and *Lepiotaceae* fungi reaching their maximum size, and the elimination of the parasite, with the garden reaching its maximum size. As the authors argue, this is another example of how QR models formalises conceptual knowledge, in this case representing alternative hypotheses of systems' behaviour.

Nuttle et al. (2004) describe models to support learning and research on food chains and the trophic cascades. They present an evaluation of three alternative mechanisms for implementing the basic trophic interaction, and discuss their potential to serve as basic building blocks for building more complex representations of food chains and food webs.

5.2 Water related Models

Aquatic systems offer the integration of physical, chemical and biological aspects that might be combined with social, cultural and economic aspects. Salles et al. (2003c) describe a model developed for understanding stream ecosystems, to predict values of

variables and to combine such understanding with restoration and proactive actions of management. The models show the effects of good and bad management practices on the effects of pollution by organic matter and the consequences for the amount of dissolved oxygen and fish stocks. Problems found during the modelling effort and implemented solutions are discussed, including the explicit representation of assumptions and the role of ambiguities in the outcomes of the models (Salles et al. 2003c).

A model for supporting stakeholders and decision makers to address problems related to nutrient cycle in stream ecosystems is presented by Neumann and Bredeweg (2004). The model explores the concept of the spiralling of resources in segments of a river from the perspective of processes within the nutrient cycle represented by the uptake rate (from nutrients to autotrophs), retention rate (from autotrophs to detritus), and release rate (from detritus to nutrients). Each segment of the river can be characterized with the definition of attributes and the influences coming from the catchments area.

Benthic macro-invertebrate communities, which have distinct responses to physical, chemical, and biological disturbances, are particularly interesting for assessing impacts of conversion of natural landscapes to urban and agricultural uses. However, modelling is difficult in this context because information relating anthropogenic activities to benthic communities is fragmented and temporally inconclusive. Tullos et al. (2004) present models that describe the impacts of watershed development and riparian deforestation activities on benthic macro-invertebrate communities based on a comprehensive understanding of the underlying processes that control these communities.

It is known that changes during the salmon development depend upon the moving sum of average daily water temperatures. Guerrin and Dumas (2001a,b) describe models for assessing the impact of the environment on salmon population dynamics. The models are implemented in QSIM and represent the functioning of spawning areas of salmon (salmon reeds) and the impact on mortality rates at early stages. The model consists of two sub-models that are quite complex, combining processes that occur at different time scales (fast and slow). A qualitative autonomous clock allows for the accumulation of degree-days from average water temperatures. The two sub-models are coupled via some shared variables and by means of transition states, in order to make alternative simulations of both. The model shows, for example, that when rain increases, the flow of water on the river also increases, increasing suspended solids and sediments and reducing the dissolved oxygen. These factors increase fish mortality, as expected from experts and the literature.

5.3 Management and Sustainability

Sustainable development is hampered by limitations on the available knowledge about important interactions and by difficulties to integrate the broad variety of regional problems into typical patterns of global change. Eisenack and Petschel-Held (2002) describe a QSIM model for understanding the interactions between nature and society. Their QR model helps to identify scenarios under which regional land-use changes due to the agricultural practices of small-holders in developing countries following the ‘impoverishment-degradation’ spiral. The outcome depends on how the small holders achieve their daily income and how this relates to environmental conditions around them. Eisenack (2003) addresses two threads of the debate on sustainable fisheries: participatory management frameworks and ‘ichtyocentric’ control strategies. A model of management framework is set up, composed of economic, ecological and political aspects, upon which viability criteria are posed. Then the author investigates how

different management strategies change the structure of the resulting state transition graph to conclude that a qualitative viability analysis can be a helpful first step for the design of controllers or the assessment of management frameworks.

5.4 Details in Qualitative Algebra

Guerrin (1991; 1992) developed a system (SIMAO) for simulating the interpretation of measurements, observations and analyses, commonly done on aquatic ecosystems for management purposes. His approach includes directed causal graphs and a qualitative algebra used to combine heterogeneous knowledge obtained by measurements (numerical), observations (linguistic) and calculations. With the support of causal graphs, SIMAO is able to reason with causal relations such as “an increase in photosynthesis decreases the CO2 concentration in water, which in turn (…) hence a risk of decrease of fish production” and to calculate the values of variables using the qualitative algebra. This qualitative algebra was also applied to other biological problems, including photosynthesis (Hunt and Cooke, 1994) and the life cycle of a plant population (Salles et al., 1996). Guerrin proposes that this approach could be an option to be used in controlled ecological life support systems (CELSS), modelling, simulation, and control (Guerrin et al., 1994).

5.5 Details in Automated Model Building

Applying QR to ecological systems pose new challenges for automatic model building. Rickel and Porter (1997) describe in the domain of plant physiology an approach for answering predictive questions. Depending on the question their approach automatically finds a model with the simplest level of detail adequate for answering that question. A particular feature of their approach is the ability to move between different timescales. Keppens and Shen (2002) address the problem of user preferences in the case of incomplete knowledge. They introduce an order of magnitude preference calculus to handle reasoning with preferences. Their models describe how the Mediterranean vegetation is being affected by various climate related factors, managed and accidental fires, and cattle farming.

5.6 Diagnosis

Diagnosis (finding the cause of undesired behaviour) is a promising area for applications for model-based reasoning in ecology (Struss and Heller, 1998). Heller and Struss (2002) use model-based technology to support the tasks of situation assessment (determining the actual state of the system) and therapy recognition (determining what can be done to recover from the undesired behaviour). Their work concerns rivers and water treatment plants.

6 Conclusion

Qualitative Reasoning provides means to build conceptual models and can be used to make the qualitative knowledge that people have explicit, organized, and manageable by means of symbolic computing. This document is developed for practitioners to help them to develop their expertise in Qualitative Reasoning and Modelling. The presented material can be processed following alternative paths, addressing different kinds of users. This document is part of a series of documents that together address different key aspects of Qualitative Reasoning and Modelling:

- D4.1 – Single-user QR model building and simulation workbench (software): refers to the software that is available for capturing and simulating qualitative models.
- D4.2.1 – User-manual for single-user version of QR workbench (document): is the user-manual that explains how to use the software.
- D6.9 – Curriculum for learning about QR modelling (*this* document): presents a curriculum that modellers can follow in order to learn about essentials of Qualitative Reasoning and Modelling, particularly focussing on the technical details required to actually build qualitative models.
- D6.1 – Framework for conceptual QR description of case studies (document): presents a structured methodology on how to capture qualitative knowledge, particularly focussing on the trajectory of developing a detailed model from a general idea.

In a collaborative effort with Sustainable Development workers, particularly in the construction of reusable knowledge libraries, it is possible to foresee a wider range of applications and better ways of dealing with the complexity of environmental systems. “Many questions of interest in ecology can be answered in terms of ‘better or worse’, ‘more or less’, ‘sooner or later’, etc.” (Rykiel, 1989), and when quantitative methods are inadequate or lacking, it is still possible to make estimates, predictions, and decisions with scientific support.

7 Literature

Addanki, S., Cremonini, R. and Penberthy, J.S. (1991) Graphs of models, *Artificial Intelligence*, volume 51, number 1-3, pages 145-177.

Amador, F., Finkelstein, A. and Weld, D. (1993) Real-time self-explanatory simulation. *Proceedings of the 11th International Conference on Artificial Intelligence*, AAAI'93, Menlo Park, California, pages 562-567.

Antunes, M.P., Seixas, M.J., Câmara, A.S. and Pinheiro, M. (1987) A New Method for Qualitative Simulation of Water Resource Systems - 2 Applications. *Water Resources Research*, volume 23, pages 2019-2022.

Bessa Machado, V. and Bredeweg, B. (2003) Building Qualitative Models with HOMER: A Study in Usability and Support. *Proceedings of the 17th International Workshop on Qualitative Reasoning (QR'03)*, Salles, P. and Bredeweg, B. (Eds.), pages 39-46, Brasilia, Brazil, August 20-22.

Biswas, G., Schwartz, D. and Bransford, J. (2001) Technology Support for Complex Problem Solving: From SAD Environments to AI. In: Forbus, K. and Feltovich, P. (Eds.) *Smart Machines in Education: The coming revolution in educational technology*, AAAI Press.

Bobrow, D.G. (Ed.) (1984) *Qualitative reasoning about physical systems*. Elsevier Science, Amsterdam, The Netherlands (reprint from the Journal *Artificial Intelligence*, volume 24, 1984)

Bossel, H. (1986) *Ecological Systems Analysis: an Introduction to Modelling and Simulation*. University of Kassel, Germany, Environmental Research Group, Technical Report.

Bouwer, A. and Bredeweg, B. (2001) VisiGarp: Graphical Representation of Qualitative Simulation Models. In Moore, J.D., Redfield G.L. and Johnson, J.L. (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, pages 294-305, IOS-Press/Ohmsha, Osaka, Japan.

Bouwer, A., Liem J. and Bredeweg, B. (2005) User Manual for Single-User Version of QR Workbench, Deliverable D4.2.1, NaturNet-Redime, EU STREP, project number 004074.

Bredeweg, B. (1992) *Expertise in Qualitative Prediction of Behaviour*. PhD thesis, University of Amsterdam, Amsterdam.

Bredeweg, B. and Winkels, R. (1998) Qualitative models in interactive learning environments: an introduction. *Interactive Learning Environments*, volume 5, issue 1-2, pages 1-18.

Bredeweg, B., and Salles, P. (2005) The Ants' Garden: Complex interactions between populations and the scalability of qualitative models. *AI communications*, 18(4):305-317.

Brown, J.S., Burton, R.R. and Kleer, J. de (1982) Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In Sleeman, D. and Brown, J.S. (Eds.), *Intelligent Tutoring Systems*, pages 227-282, Academic Press, New York USA.

Câmara, A.S., Antunes, P.C., Pinheiro, M.D. and Seixas, M.J. (1987) Linguistic dynamic simulation - A new approach. *Simulation*, volume 49, number 5, pages 208-212.

Collins, J. and Forbus, K. (1989) *Building qualitative models of thermodynamic processes.* ILS Technical report, Computer Science Department, Northwestern University, Evanston, USA.

Eisenack, K. and Petschel-Held, G. (2002) Graph Theoretical Analysis of Qualitative Models in Sustainability Science. *Proceedings of the International Workshop on Qualitative Reasoning*, (QR'02), Agell, N. and Ortega, J.A. (Eds.), pages 53-60, Sitges/Barcelona, Spain, June 10-12, 2002.

Eisenack, K. (2003) Qualitative Viablility Analysis of a Bio-Socio-Economic System. *Proceedings of the17th International Workshop on Qualitative Reasoning, (QR'03)*, Salles, P. and Bredeweg, B. (Eds.), pages 63-70, Brasília, Brazil, August 20-22, 2003.

Falkenhainer, B. & Forbus, K. (1991) Compositional modeling: Finding the right model for the Job. *Artificial Intelligence*, volume 51, number 1-3, pages 95-143.

Forbus, K.D. (1984) Qualitative process theory. *Artificial Intelligence*, volume 24, number 1-3, pages 85-168.

Forbus, K.D. (1986) *The Qualitative Process Engine*. University of Illinois, Department of Computer Science Technical Report. Reprinted in Weld, D.S. and Kleer, J. de (Eds.) (1990) *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, San Mateo, California, USA.

Forbus, K.D. (1988) Qualitative physics: past, present and future. In: H.E. Shrobe (Ed.), *Exploring artificial intelligence*, pages 239-296, Morgan Kaufmann, San Mateo. USA.

Forbus, K.D., Whalley, P., Everett, J., Ureel, L., Brokowski, M., Baher, J. and Kuehne, S. (1999) CyclePad: An articulate virtual laboratory for engineering thermodynamics, *Artificial Intelligence*, volume 114, number 1-2, pages 297-347.

Forbus, K.D, Carney, K., Harris, R. and Sherin, B.L. (2001) A qualitative modeling environment for middle-school students: A progress report. In: Biswas, G. (Ed.), *Proceedings of the 15th International Workshop on Qualitative Reasoning (QR'01)*, pages 65-72, St. Mary's University, San Antonio, Texas.

Fromherz, M.P.J., Bobrow, D.G., and Kleer, J. de (2003) Model-based Computing For Design and Control of Reconfigurable Systems, *AI Magazine*, volume 24, number 4, pages 102-130.

Guerrin, F. (1991) Qualitative Reasoning about an Ecological Process: Interpretation in Hydroecology. *Ecological Modelling*, volume 59, pages 165-201.

Guerrin, F. (1992) Model-based Interpretation of Measurements, Analysis and Observations of an Ecological Process. *AI Applications*, volume 6, number 3, pages 89-101.

Guerrin, F., Bousson, K., Steyer, J.Ph. and Travé-Massuyès, L. (1994) Qualitative Reasoning Methods for CELSS Modeling. *Advances in Space Research*, volume 14, number 11, pages 307-312.

Guerrin, F. and Dumas, J. (2001a) Knowledge representation and qualitative simulation of salmon reed functioning. Part I: qualitative modelling and simulation. *BioSystems*, Volume 59, pages 75-84.

Guerrin, F. and Dumas, J. (2001b) Knowledge representation and qualitative simulation of salmon reed functioning. Part II: qualitative model of redds. *BioSystems*, Volume 59, pages 85-108.

Heller, U. and Struss, P. (2002) Consistency-based Problem Solving for Environmental Decision Support. *Computer - Aided Civil and Infrastructure Engineering*, volume 17, pages 79-92.

Hollan, J.D., Hutchins, E.L., and Weitzman, L. (1984) STEAMER: An interactive inspectable, simulation based training systems. *AI Magazine*, volume 5, pages 15-27.

Hunt, J.E. and Cooke, D.E. (1994). Qualitative modeling photosynthesis. *Applied Artificial Intelligence*, volume 8, pages 307-332.

Iwasaki, Y. and Simon, H.A. (1986) Causality in device behaviour. *Artificial Intelligence*, volume 29, pages 3-32, 1986.

Keppens, J. and Shen, Q. (2002) On Supporting Dynamic Constraint Satisfaction with Order of Magnitude Preferences. *Proceedings of the 16th International workshop on Qualitative Reasoning, (QR'02)*, Agell, N. and Ortega, J.A. (Eds.), pages 143-150, Sitges/Barcelona, Spain, June 10-12, 2002.

Kim, H. (1993) *Qualitative reasoning about fluids and mechanics*. Ph.D. thesis, Computer Science Department, Northwestern University, Evanston, USA.

Kleer, J. de and Brown, J.S. (1984) A qualitative physics based on confluences. *Artificial Intelligence*, volume 24, number 1-3, pages 7-83.

Kleer, J. de and Brown, J.S. (1986) Theories of causal ordering. *Artificial Intelligence*, volume 29, pages 33-61.

Kuipers, B. (1986) Qualitative simulation. *Artificial Intelligence*, volume 29, pages 289-388, 1986.

Kuipers, B. (1994) *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT Press, Cambridge, Massachusetts.

May, R.M. (1973) Qualitative stability in model ecosystems. *Ecology*, volume 54, number 3, pages 638-641.

McIntosh, B.S. (2003) Qualitative modelling with imprecise ecological knowledge: a framework for simulation. *Environmental Modelling and Software*, volume18, pages 295-307.

Moore, A.D. and Noble, I.R. (1990) An Individualistic Model of Vegetation Stand Dynamics. *Journal of Environment Management*, volume 31, pages 61-81.

Moore, A.D. and Noble, I.R. (1993) Automatic Model Simplification: the Generation of Replacement Sequences and their Use in Vegetation Modelling. *Ecological Modelling*, volume 70, pages 137-157.

Neumann, M. and Bredeweg, B. (2004) A qualitative model of the nutrient spiraling in lotic ecosystems to support decision makers for river management. *Proceedings of the 18th International Workshop on Qualitative Reasoning (QR'04)*, de Kleer, J. and Forbus, K.D. (Eds.), pages 159-164, Evanston, USA, August 2-4, 2004.

Noble, I.R. and Slatyer, R.O. (1980) The Use of Vital Attributes to Predict Successional Changes in Plant Communities Subject to Recurrent Disturbances. *Vegetatio*, volume 43, pages 5-21.

Nuttle, T., Bredeweg, B. and Salles, P. (2004) Qualitative Reasoning about Food Webs: Exploring Alternative Representations. *Proceedings of the 18th International Workshop on Qualitative Reasoning (QR'04)*, de Kleer, J. and Forbus, K.D. (Eds.), pages 89-96, Evanston, USA, August 2-4, 2004.

Price, C. and Peter Struss, P. (2003) Model-based Systems in the automotive Industry, *AI Magazine*, volume 24, number 4, pages 17-43.

Raiman, O. (1986) Order of magnitude reasoning. *Proceedings of the AAAI*, pages 100-104, San Mateo, California, Morgan Kaufmann.

Rickel, J. and Porter, B. (1997) Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence*, volume 93, pages 201-260.

Rykiel, E.J. (1989) Artificial Intelligence and Expert Systems in Ecology and Natural Resource Management. *Ecological Modelling*, volume 46, pages 3-8.

Salles, P. (1997) *Qualitative models in ecology and their use in learning environments*. Ph.D. thesis, University of Edinburgh, Edinburgh, Scotland, UK.

Salles, P. and Bredeweg, B. (1997) Building Qualitative Models in Ecology. In: Ironi, L. (Ed.) *Proceedings of the 11th International Workshop on Qualitative Reasoning (QR'97)*. Instituto di Analisi Numerica C.N.R., Pubblicazioni no. 1036 , Pavia, Italy.

Salles, P. and Bredeweg, B. (2003) Qualitative Reasoning about Population and Community Ecology. *AI Magazine*, volume 24, number 4, pages 77-90.

Salles, P. and Bredeweg, B. (in press) Modelling Population and Community Dynamics with Qualitative Reasoning. *Ecological Modelling*.

Salles, P., Bredeweg, B., Araujo, S. and Neto, W. (2003a) Qualitative models of interactions between two populations. *AI Communications*, volume 16, issue 4, pages 291-308.

Salles, P., Bredeweg, B. and Bensusan, N. (2003b). The Ant's Garden: Qualitative Models of Complex Interactions between Populations. *Proceedings of the 17th International Workshop on Qualitative Reasoning (QR'03)*, Salles, P. and Bredeweg, B. (Eds.), pages 163-170, Brasilia, Brazil, August 20-22.

Salles, P., Bredeweg, B. and Araujo, S. (2003c). Qualitative Models of Stream Ecosystem Recovery: Exploratory Studies. *Proceedings of the 17th International Workshop on Qualitative Reasoning (QR'03)*, Salles, P. and Bredeweg, B. (Eds.), pages 155-162, Brasilia, Brazil, August 20-22.

Salles, P.S.B.A., Muetzelfeldt, R.I. and Pain, H. (1996) Qualitative Models in Ecology and their Use in Intelligent Tutoring Systems. *Proceedings of 10th International Workshop on Qualitative Reasoning (QR'96)*, Iwasaki, Y. and Farquhar, A. (Eds.). AAAI Technical Report WS-96-01.

Struss, P. and Heller, U. (1998) Process-oriented Modelling and Diagnosis - Revising and Extending the Theory of Diagnosis from First Principles. *Working Notes of the Ninth International Workshop on Principles on Diagnosis (DX-98)*, Sea Crest, Cape Cod, USA.

Tullos, D.D., Neumann, M. and Sanchez, J.J.A. (2004) Development of a Qualitative Model for Investigating Benthic Community Response to Anthropogenic Activities. *Proceedings of the 18th International Workshop on Qualitative Reasoning (QR'04)*, de Kleer J. and Forbus, K.D. (Eds.), pages 179-185, Evanston, USA, August 2-4, 2004.

Weld, D.S. (1988) Comparative analysis. *Artificial Intelligence*, volume 36, pages 333-374.

Weld, D.S. and Kleer, J. de (Eds.) (1990) *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, San Mateo, California, USA.

Williams, B.C., Ingham, M., Chung, S., Elliott, P., and Hofbaur, M. (2003) Model-based Programming of Fault-Aware Systems, *AI Magazine*, volume 24, number 4, pages 61-76.

Appendix A: Qualitative Reasoning Vocabulary

Learning about qualitative reasoning requires the acquisition of a certain vocabulary. This appendix provides a short definition for each item in the vocabulary and can be used as a reference. An overview of the available modelling ingredients is shown in Figure A.1.

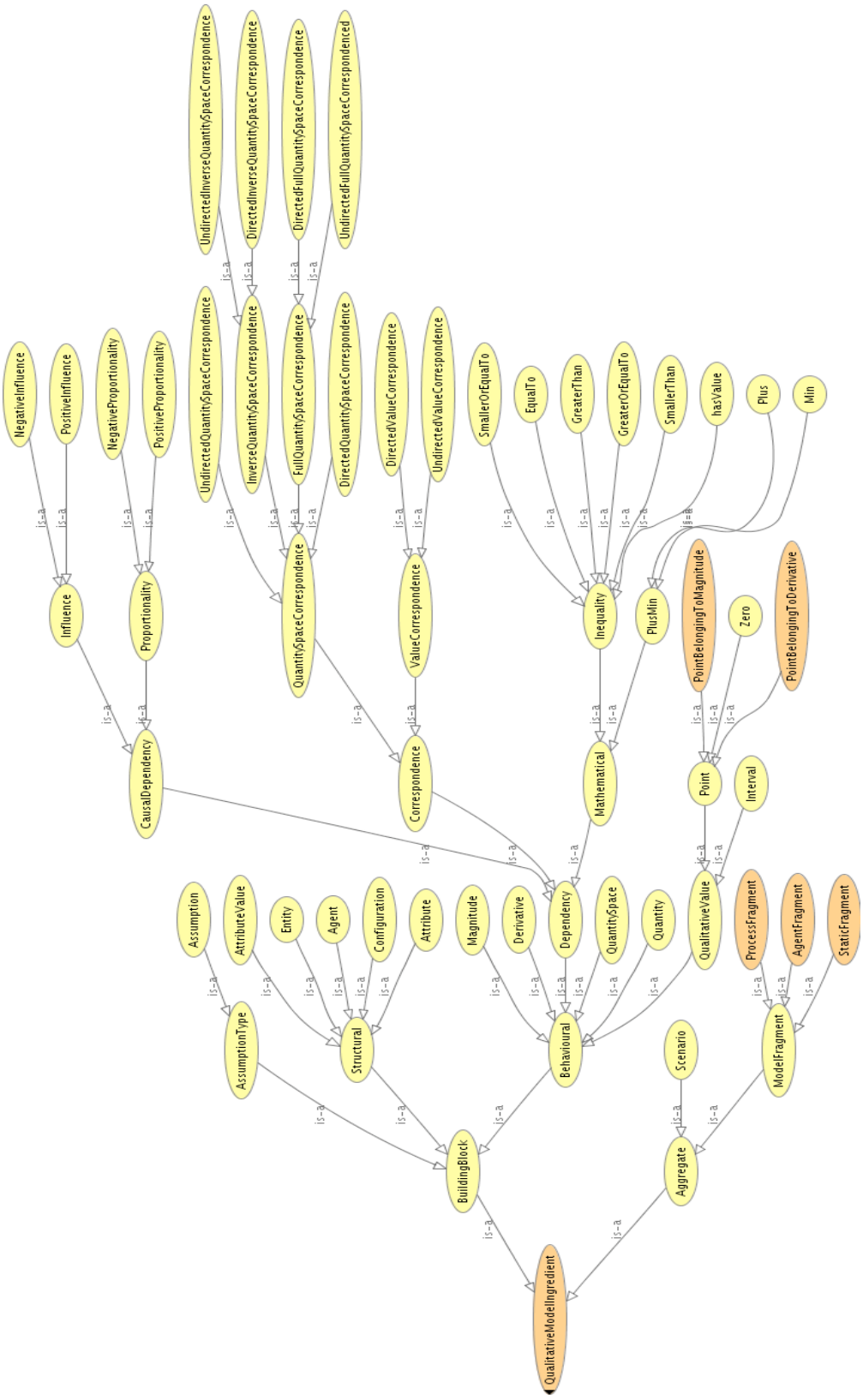


Figure A.1: QR modelling ingredients hierarchy

A.1: The notion of Model

A central concept to Qualitative Reasoning and Modelling is the 'model': A model is an abstract representation of a system that enables users to make testable predictions about what happens to a system in different situations.

A.2: Structural Building Blocks

Structural modelling ingredients describe the organisation of the concepts within a system and the static features of those concepts.

A.2.1 Entities

Entities are the physical objects or abstract concepts that play a role within the system. These entities are arranged in a subtype hierarchy.

Examples:

- An Animal is an Entity
- A Vertebrate is an Animal
- A Mammal is a Vertebrate
- A Zebra is a Mammal

Remarks:
The organisation of entities in a subtype hierarchy allows more general entities higher in the hierarchy to be used in model fragments instead of more specific entities below them. This allows for more efficient modelling, as there is no need to - for example - create a model fragment describing flow for every occurrence of an entity. Instead, just one model fragment is needed to describe the characteristics of the whole class.

A.2.2 Configurations

Configurations are used to model relations between instances of entities and agents; they are sometimes referred to as structural relations.

Examples:

- Lion preys on Zebra.
- Container contains Water.
- Container is connected to Tube.

Remarks:
Entities and agents cannot exist within a model fragment without a configuration connecting them; they have to be structurally related.

A.2.3 Attributes

Attributes are properties of entities that remain static during simulation (i.e. do not change). They have an associated set of attribute values, which are the possible values of the attributes.

Examples:

- The colour of the animal is black.
- The openness attribute of a pan filled with boiling water can be either open or closed.
- The status of a lock can be either unlocked of locked.
- The status of a device can be turned on or turned off.

Remarks:
Attributes are often used as conditions in model fragments to indicate that a specific process is only active when certain attributes have certain values.

A.2.4 Agents

Agents are used to model entities outside of the modelled system. Agents can have quantities influencing the rest of the system, which are sometimes called exogenous or external influences.

Examples:
- James fishing in a pond, decreasing the number of fish within that ecosystem.
- John filling one of the tubes in the communicating vessels system
- A manager setting a Brazilian Cerrado forest on fire.

A.2.5 Assumptions

Assumptions are labels that are used to indicate that certain conditions are presumed to be true. They are often used to constrain the possible behaviour of a model. Because they describe neither structural nor behavioural aspects of a system, they belong to neither the structural building blocks nor the behavioural building blocks categories.

Examples:
- The height of water in each of the fluid containers in the communicating vessels system is greater than zero.
- The populations within an ecological system are closed (no migration).
- Adhesion and cohesion are assumed to have no effect on the height of a fluid in a container.

Remarks:
Assumptions can only be used as a condition and are often combined with an inequality relation as a consequence.

A.3: Behavioural Building Blocks: Features

The behavioural modelling ingredients are further distinguished into features and dependencies. The behavioural features describe the variable aspects of entities of a system.

A.3.1 Quantities

Quantities represent changeable features of entities and agents. Each quantity has two associated quantity spaces: a definable one for the magnitude, and the default quantity space {Min, Zero, Plus} for the derivative of the quantity.

Examples:
- A contained liquid has the quantities of: volume, height and pressure.
- The temperature of an object.
- The size of a population.

A.3.2 Magnitude

The magnitude indicates the current value of a quantity.

Example:
- The magnitude of the temperature of the water is its boiling-point.

A.3.3 Derivative

The derivative indicates the current value of the first derivative of a quantity, and indicates whether the magnitude is increasing, decreasing or stable.

Example:

- The derivative of the temperature of the water is steady.

A.3.4 Quantity Spaces

A quantity space specifies a range of qualitative values a quantity magnitude or derivative can have. The qualitative values in a quantity space form a total order. Each qualitative value is either a point or an interval, and within the quantity spaces these two types consecutively alternate.

Examples:

- The quantity space for the height of contained liquid is {zero, positive, full}.
- The quantity space for the temperature of a material could be: {absolute zero, solid phase, freezing point, liquid phase, boiling point, gas phase}.
- The quantity space for the size of a population could be: {zero, positive} if there is no maximum size for a population. Otherwise, {zero, positive, maximum} could be used.

A.3.5 Qualitative Value

A qualitative value is either a point or interval that can become the current magnitude or current derivative of a quantity. Qualitative values are contained by quantity spaces.

Examples:

- Zero, minimum, negative, positive, high, full, maximum, medium.

Remarks:
Qualitative values having the same name do not necessarily represent the same value. For example, two contained liquids with their height quantities equal to the value positive do not necessarily have the same height, as one could be smaller than the other. The one exception is the value zero, which specifies the turning point between positive and negative. Zero is universally equal among quantity spaces.

A.3.6 Current Value and Quantity Value

The current value refers to either the value of the magnitude or the value of the derivative of a quantity in a specific situation. The quantity value is the combination of the current value of the magnitude and the current value of the derivative of a quantity. Quantity values are often written down as: <magnitude, derivative>.

Examples:

- <full, decreasing>
- <zero, stable>
- <positive, increasing>

Remarks:
The changes in quantity values are visualised in the value history.

A.4: Behavioural Building Blocks: Causal Dependencies

Behavioural dependencies describe relations between behavioural features. There are three kinds of dependencies: causal dependencies, mathematical dependencies and correspondences. Causal dependencies are used to model how processes induce changes, either directly or indirectly. These dependencies are represented as influences and proportionalities.

A.4.1 Influences

Influences are directed relations between two quantities, and are either positive or negative. Influences are the cause of change within a model, and are therefore said to model processes. Depending on the magnitude of the source quantity and the type of influence, the derivative of the target quantity either increases or decreases. An influence I+(Q2,Q1) causes the quantity Q2 to increase if Q1 is positive, decrease if it is negative, and remain stable when it is zero (assuming there are no other causal dependencies on Q2). For an influence I- this is just the opposite. Influences are also referred to as direct influences.

Examples:

- I+(Size, Natality) – Population *Size* increases if population *Natality* is above zero, decreases if it is below zero, and remains stable if it is zero.
- I-(Size, Mortality) – Population *Size* decreases if population *Mortality* is above zero, increases if it is below zero, and remains stable if it is zero.
- I+(Height, Growth rate) – Plant *Height* increases if plant *Growth rate* is above zero, decreases if it is below zero, and remains stable if it is zero.

Remarks:
Influences should be used when the following relation has to be modelled: ‘If the source quantity has a non-zero value, the target quantity will change’.

A.4.2 Proportionalities

Proportionalities are directed relations between two quantities. They propagate the effects of a process, (i.e. they set the derivative of the target quantity depending on the derivative of the source quantity). For this reason, they are also referred to as indirect influences. Like influences, proportionalities are either positive or negative. A proportionality P+(Q2,Q1) causes Q2 to increase if Q1 increases, decrease if Q1 decreases, and remain stable if Q1 remains stable (given that there are no other causal influences on Q2). For a proportionality P- the opposite applies.

Examples:

- P+(Height,Amount) – Fluid *Height* increases if fluid *Amount* increases, decreases if fluid *Amount* decreases, and remains stable if fluid *Amount* remains stable.
- P-(Height,Width) – Fluid *Height* decreases if container *Width* increases, increases if container *Width* decreases, and remains stable if container *Width* remains stable.
- P+(Birth,Size) – Population *Birth* increases if population *Size* increases, decreases if population *Size* decreases, and remains stable if population *Size* remains stable.)

Remarks:
Proportionalities should be used when the following relation has to be modelled: If the source quantity changes, the target quantity will change too.

A.5: Behavioural Building Blocks: Mathematical Dependencies

Mathematical dependencies describe relations between behavioural features.

A.5.1 In/equalitities

In/equalities (<,≤,=,≥,>) specify an ordinal relation between two items, (i.e. that one item is different from, or equal to, the other item). Because inequalities specify an order between items, they are sometimes referred to as ordinal relations. There are eleven ways to use inequalities, depending on the type of the two items related by it. Table 1 shows the possible inequalities between magnitude items, while Table 2 shows them for derivative items. Note that the type of an inequality from A to B is considered the same type (i.e. has the same number) as an inequality from B to A.

1. From a magnitude to another magnitude.
2. From a point belonging to the quantity space of a magnitude, to a point belonging to the quantity space of another magnitude.
3. From a plus/min relation between magnitude items, to another plus/min relation between magnitude items.
4. From a magnitude to a point belonging to the quantity space of a magnitude. The reverse is impossible.
5. Between a magnitude and a plus/min relation between magnitude items.
6. Between a point belonging to the quantity space of a magnitude and a plus/min relation between magnitude items.
7. From a derivative to another derivative.
8. Between a plus/min relation between derivative items and another plus/min relation between derivative items.
9. From a derivative to a point belonging the quantity space of a derivative (i.e. zero). Note that the reverse is impossible.
10. Between a derivative and a plus/min relation between derivative items.
11. Between a point belonging to the quantity space of a derivative (i.e. zero) and a plus/min relation between derivative items.

Table 1: The possible inequalities between magnitude items

To From	Magnitude	Point (Magnitude)	Plus/Min (Magnitude)
Magnitude	1	4	5
Point (Magnitude)	Impossible	2	6
Plus/Min (Magnitude)	5	6	3

Table 2: The possible inequalities between derivative items

To From	Derivative	Point (Derivative)	Plus/Min (Derivative)
Derivative	7	9	10
Point (Derivative)	Impossible	Impossible	11
Plus/Min (Derivative)	10	11	8

Examples:

1. The magnitude of the number of predators is greater than the magnitude of the number of prey.
2. The value max of the height of the water container is equal to the value max of the height of the other water container.
3. The magnitude of the population natality minus the magnitude of the population natality is smaller than the difference between the magnitude of immigration and emigration.
4. The magnitude of the amount of fluid in the container is greater than zero.
5. The difference between the pressure in the left water container and the pressure in the right water container is equal to the flow.
6. *No example given*.
7. *No example given*.
8. *No example given*.
9. *No example given*.
10. *No example given*.
11. *No example given*.

A.5.2 Plus/Min relations

Using plus/min relations more complex expressions can be created than is possible with only inequalities. They are used to calculate the sum of difference between two items. Plus/min relations can be the target or source of an inequality relation. There are nine different ways plus/min relations can be used, depending on the type of the two items related by it. Table 3 shows the possible uses of plus/min relations between magnitude items, while Table 4 shows the possibilities between derivative items. Note that the type of a plus/min relation from A to B is considered the same type (i.e. has the same number) as an inequality from B to A.

1. From a magnitude to another magnitude.
2. From a point belonging to the quantity space of a magnitude, to a point belonging to the quantity space of another magnitude.
3. From a plus/min relation between magnitude items, to another plus/min relation between magnitude items.
4. From a magnitude to a point belonging to the quantity space of a magnitude.
5. Between a magnitude and a plus/min relation between magnitude items.
6. Between a point belonging to the quantity space of a magnitude and a plus/min relation between magnitude items.
7. From a derivative to another derivative.
8. Between a plus/min relation between derivative items and another plus/min relation between derivative items.
9. Between a derivative and a plus/min relation between derivative items.

Table 3: The possible plus/min relations between magnitude items

To From	Magnitude	Point (Magnitude)	Plus/Min (Magnitude)
Magnitude	1	4	5
Point (Magnitude)	4	2	6
Plus/Min (Magnitude)	5	6	3

Table 4: The possible plus/min relations between derivative items

To From	Derivative	Point (Derivative)	Plus/Min (Derivative)
Derivative	7	Impossible	9
Point (Derivative)	Impossible	Impossible	Impossible
Plus/Min (Derivative)	9	Impossible	8

Examples:

1. The magnitude of the predation of the hunters is equal to their food need minus the amount of prey.
2. *No example given*.
3. The sum of birth and immigration minus the sum of death and immigration is equal to the magnitude of population growth.
4. *No example given*.
5. The intensity of the light plus the concentration of carbon dioxide plus the surface area of the leaves is equal to the rate of photosynthesis.
6. *No example given*.
7. *No example given*.
8. *No example given*.
9. *No example given*.

A.6: Behavioural Building Blocks: Correspondences

Correspondences are used to indicate that qualitative values of different quantity spaces occur at the same time.

A.6.1 Value Correspondences

Value Correspondences are relations between qualitative values of quantity spaces belonging to different quantities, and can be either directed or undirected. Directed means that when value A of quantity space X corresponds to value B of quantity space Y, the simulator derives that quantity space Y has value B when quantity space X has value A. If the correspondence is undirected, it also derives the value A of quantity space X when quantity space Y has value B.

A.6.2 Quantity Space Correspondences

Quantity Space Correspondences exist between two quantity spaces, indicating that each of the values of the quantity spaces of those quantities correspond to each other. Like value correspondences these can also be either directed or undirected.

A.6.3 Inverse Quantity Space Correspondences

Quantity space correspondences can also be inversed, indicating that the first value of the first quantity space corresponds to the last value of the second quantity space; the second value corresponds to the penultimate value, etc. Inverse correspondences can also be either directed or undirected.

A.6.4 Full Quantity Space Correspondences

Full Quantity Space Correspondences indicate that there is both a quantity space correspondence between the quantity spaces of the magnitudes, as well as a quantity space correspondence between the quantity spaces of the derivatives. Full correspondences can be either directed or undirected.

A.7: Aggregates

There are two kinds of aggregates, namely model fragments and scenarios. Aggregates are complex model constituents, as they consist of multiple model ingredients.

A.7.1 Model Fragments

Model fragments describe part of the structure and behaviour of a system in a general way. They are partial models which are composed of multiple ingredients. Model fragments have the form of a rule. This means that model ingredients are incorporated as either conditions or consequences. Table 3 shows the abilities of ingredients to be used as a condition or consequence. As can be seen from the table, model fragments themselves can be reused within other model fragments as conditions. Furthermore, sub-classes of model fragments can be made, which augment the parent model fragment with new ingredients. The consequence ingredients of model fragments that match on the actual system situation, will be added to the scenario. In that case, the scenario fulfils the conditions specified in the model fragment (which describes a general situation). There are three different kinds of model fragments: static fragments, process fragments, and agent fragments.

Condition

The conditions in a model fragment indicate what things should be true within a scenario (or state) in order for the consequences to be true. The model ingredients that can be used as conditions within a model fragment are shown in Table 5.

Table 5: Condition and consequence abilities of QRM ingredients

Possible conditions	**Impossible conditions**
Entities Configurations Agents Attributes Quantities Inequalities Min/Plus Assumptions Model Fragments	Correspondences Proportionalities Influences
Possible consequences	**Impossible consequences**
Entities (except in static MF) Configurations (except in static MF) Attributes Quantities Inequalities Min/Plus Correspondences Proportionalities Influences	Agents Assumptions Model Fragments

Consequence

Consequences within a model fragment capture the information that is added to a scenario when the conditions of the model fragment are fulfilled. The model ingredients that can be used as consequences within a model fragment are shown in Table 5.

Model Fragment Types

Static Fragments
In static fragments all ingredients may occur except agents and influences. Static fragments are used to describe parts of the structure of the system, and the proportionalities that exist between the quantities.

Process Fragments
Process fragments contain at least one influence, but no agents. These model fragments are used to describe processes that take place within the system.

Agent Fragments
Agent fragments contain an agent and may contain one or more influences. Agent fragments are used to describe the influences agents (exogenous entities) have on the system.

A.7.2 Scenarios

Scenarios describe the actual state of a system, and can consist of all the ingredients that can be used as conditions in model fragments, except for other model fragments (see table 1). The use of ingredients in scenarios differs from their use in model fragments, as in scenarios ingredients are incorporated as facts instead of as conditions or consequences. Obviously, this allows model fragments to match with scenarios. Scenarios are used as input for the qualitative simulator and function as the starting state from which the rest of the behavioural graphs are generated.

A.7.3 Identity

Identity relations are used to specify that two model ingredients in different imported model fragments are the same. There are two possible applications. Firstly, they can be used to indicate that two entities in different imported model fragments are actually the same one. Secondly, they can be used to specialise entities in child model fragments. For example, the fish entity in a parent model fragment can be specialised to a salmon entity in the child model fragment.

A.8: Simulation Vocabulary

A.8.1 Simulation

The word simulation is ambiguous, as it is used to refer to two things. Firstly, it is used as a synonym for the output generated by the qualitative simulator in the simulation environment. Secondly, it is used to designate the process of qualitative reasoning itself, which is used to generate the simulation output. The simulation environment in the Garp3 software provides control to the reasoning process to be able to follow each step of the simulation, therefore distinguishing the concepts of full simulation, which refers to the complete simulation with the resulting state-graph, from that of partial simulation, which is an incomplete simulation and state-graph.

A.8.2 Simulation Output

Simulation output refers to the state-graph generated by the qualitative reasoning engine as a result of simulating a qualitative model by providing a scenario as input.

A.8.3 The notion of State

State

A state describes a particular situation of a modelled system. A state contains information about the structural organisation, the current values of quantities, inequalities, and the active model fragments. A state can be interpreted, terminated, ordered, or closed.

Interpreted State

An interpreted state is either a scenario, or a new state generated during the closing of a state, and all the applicable model fragments.

Terminated State

A state is referred to as terminated if all the possible ways a state can end, due to changes of magnitudes of derivatives of quantities, are identified.

Ordered State

A state is referred to as ordered, when transitions overruled by other transitions with higher priority have been pruned, and terminations that occur simultaneously due to correspondences have been merged.

Closed State

A state is referred to as closed when the successive states and transitions are generated from its pruned and merged terminations.

A.8.4 The notion of State-graph and Behaviour

Behaviour

Within the context of qualitative reasoning and modelling the behaviour of a system refers to the changes of quantity values (and in/equality statements) as the result of processes that are active within the modelled system.

Transition

A transition describes the change between two different states. The transition information describes the transition conditions that are fulfilled by the source state, and the results of the change (i.e. the next state).

State-graph

A state-graph is a set of states, and the possible transitions between those states, which represents the behaviour of modelled system. State-graphs are generated by simulating a qualitative model.

Behaviour-graph

See state-graph.

Behaviour Path

A behaviour path is a sequence of successive states within a state-graph. The path describes the evolution of the quantity values and ordinal relations as time progresses.

A.8.5 The notion of History

Value History

The value history describes how quantity values change through a sequence of states (usually a behaviour path).

Transition History

The transition history shows the transitions that cause the changes from one state to the other for a sequence of states.

Equation History

The equation history describes how the ordinal relations change through a sequence of states (usually a behaviour path).

A.9: Inequalities and Values as Conditions or Consequences

A.9.1 Value Assignment

Values are abbreviations for inequalities between a quantity, or a derivative, and qualitative values in its quantity space. The qualitative simulator converts these values to (possibly multiple) inequality relations. For example, a magnitude set to an interval between two points using a value is mapped to the inequality relations smaller than (with respect to the higher point), and small than (with respect to the lower point). A value assignment to a point is equivalent to an equality relation to that point.

Remarks:
When value assignments (and/or inequalities) are used as a condition, and the value (and/or inequality) cannot be derived, the reasoning engine tries to assume the value (and/or inequality). If the resulting state is not inconsistent, it will appear in the state-graph. Value assignments (and/or in/equalities) as consequences should be used conservatively, because if two model fragments with opposing consequence value assignments (and/or in/equalities) become active, the resulting state is inconsistent and will be removed. It is good practice to only use these types of ingredients in combination with an assumption.

Appendix B: Assignments

This appendix presents a number of assignments, which can be undertaken individually or by small groups, preferably pairs. The advantage of the latter is that discussion and negotiation with co-workers often leads to a more in depth exploration of the issues involved. Solutions to the assignments are given in Chapter 4. Notice that the assignments do not discuss details concerning the use of the Garp3 workbench. Learners are advised to consult the Garp3 user manual for such details (Bouwer et al, 2005). The assignments progress in complexity and are organised as follows:

- **Getting started**: The *Tree & Shade* assignment takes the learner stepwise through the whole enterprise of creating a simple qualitative model. Moreover, each modelling step has a reference to the Garp3 user manual (Bouwer et al, 2005). This supports novices in carrying out the steps using the Garp3 software.
- **Some experience required**: The *Population Behaviour* assignments have less specific instructions. The learner is expected to develop the crucial steps. In order to make that doable the models are relatively simple and the assignments gradually progress towards more complex issues.
- **Advanced**: The *Communicating Vessels* and *Heating & Boiling* assignments are complex. Learners are expected to develop the model given general instructions.

B.1: Tree & Shade

The goal of this assignment is to create a qualitative model that describes the growth of a tree in time and the effects this growth has on the shaded area that this tree produces. The model is kept a simple as possible, as the aims are to be familiarised with the model ingredients from which a qualitative model is built, and to learn to use the qualitative reasoning and modelling software Garp3. The following model ingredient types are covered in this assignment:

- Entity
- Quantity (with value: magnitude and, derivative)
- Quantity space
- Scenario
- Model fragment types: Static & Process
- Influence (= direct influence)
- Proportionality (= indirect influence)
- Correspondence

Consider a growing tree that casts a shadow on the ground. Assume that the tree always grows (ignoring the need for water, sunlight, air and minerals). The size of the shadow depends on the size of the tree. This is a dynamic feature: the shaded area increases when the tree becomes bigger. Indirectly, the growth of the tree causes the shaded area to increase.

Step 1: Entity Hierarchy

The model implementation start with the definition of the key objects within the system. In this case this is the tree. In order to keep the model simple, we ignore other possible objects in the system, like the sun, the forest the tree is located in, or the amount of minerals in the soil.

Do the following:

- Add an entity *Tree* to the entity hierarchy [D4.2.1, Section 4.1][8]

[8] See Bouwer et al. 2005

Step 2: Quantities and Quantity Spaces

Quantities describe the changeable features of entities. The possible values a quantity can have are described in a quantity space. The *Tree* in our system, which grows and produces shade, has three important quantities, namely:

- the size of the *Tree* (*Size*);
- the area of shade caused by the *Tree* (*Shade*); and,
- the rate at which the *Tree* grows (*Growth rate*).

Each of the above quantities has to have an associated quantity space. For the *Size* of the *Tree*, three qualitative values are possible: *Small*, *Medium* and *Large*. Since the *Shade* depends on the *Size* of the *Tree*, it is logical to choose the same quantity space for that quantity. Finally, there is either no growth, or some positive amount of growth. Therefore, the possible qualitative values for the *Growth rate* are: *Zero* and *Plus*. The quantities and their quantity spaces are summarised in Table 1.

Table 1: The quantities in the Tree & Shade model and their quantity spaces

Quantity	Quantity Space
Size	{Small, Medium, Large}
Shade	{Small, Medium, Large}
Growth rate	{Zero, Plus}

Do the following:

- Define a quantity space called 'sml' with the values {Small, Medium, Large} (*Small* and *Large* should be intervals, and *Medium* a point.) [D4.2.1, Section 4.5].
- Define the quantity *Size* and associate it with the quantity space 'sml' [D4.2.1, Section 4.4].
- Define the quantity *Shade* and associate it with the quantity space 'sml'.
- Define a quantity space called 'zp' with the values {Zero, Plus} (*Zero* should be a point).
- Define the quantity *Growth rate* and associate it with the quantity space 'zp'.

Step 3: Creating a Scenario

A scenario is the starting point for a simulation. It is a description of a (typical) situation to which the knowledge captured in the model will be applied when simulating. A model usually has multiple scenarios that can be used to run simulations. But in this assignment we consider only one scenario, namely: ‘a tree with a small size’.

Do the following:

- Define a new scenario: 'A small growing tree' [D4.2.1, Section 5].
- Open the scenario in the scenario editor.
- Add the entity *Tree* to the scenario.
- Add the quantity *Size* attached to the entity *Tree* to the scenario.
- Set the value of the *Size* quantity to *Small*.
- Add the quantity *Shade* attached to the entity *Tree* to the scenario.
- Set the value of the *Shade* quantity to *Small*.
- Close the scenario editor.

Step 4: Static Model Fragment with Knowledge about Size and Shade

In this step a model fragment will be created which represents the relationship between *Size* and *Shade*. Changes in the shaded area (*Shade*) are the consequence of changes in the *Size* of the *Tree*. In this case, when *Size* increases the *Shade* also increases, and when *Size* decreases the *Shade* also decreases. This relation can be modelled using a positive proportionality.

Do the following:

- Define a new static model fragment: 'Tree with shade' [D4.2.1, Section 6].
- Open the model fragment in the model fragment editor.
- Add the entity *Tree* to the model fragment as a condition.
- Add the quantity *Size* attached to the entity *Tree* to the model fragment as a consequence.
- Add the quantity *Shade* attached to the entity *Tree* to the model fragment as a consequence.
- Add a positive proportionality from *Size* to *Shade* to the model fragment.
- Close the model fragment editor.

Step 5: Process Model Fragment with Knowledge about Tree Growth

In this step the relationship between the *Growth rate* and the *Size* will be modelled. The *Growth rate* is assumed to be positive and stable and constantly causing the *Size* of the *Tree* to increase. This relation is represented using a positive influence between *Growth rate* and *Size*.

Do the following:

- Define a new process model fragment: 'Growth of tree'.
- Open the model fragment in the model fragment editor.
- Add the entity *Tree* as a condition.
- Add the quantity *Size* as a consequence associated to the *Tree.*
- Add the quantity *Growth rate* as a consequence.
- Set the value (as a consequence) of the *Growth rate* to *Plus* (to indicate that a small, medium or large tree always grows).
- Set the value of the derivative of the *Growth rate* to stable (to indicate that the growth does not change).
- Create a positive influence from the *Growth rate* to the *Size.*
- Close the model fragment editor.

Answer question:

- Why should the value of the Growth rate be set as a consequence, and not as a condition?

Step 6: Running and Inspecting the Model

When running the model, the goal is to have the simulator predict (simulate) the changes of the *Size* and the *Shade* of the Tree as time passes. Ideally, the resulting simulation consists of 3 states: a first state in which both *Size* and *Shade* are *Small*, a second state in which both *Size* and *Shade* are *Medium*, and a third state in which both *Size* and *Shade* are *Large*.

Do the following:

- Start the simulation of the model using the previously created scenario: 'A small growing tree' [D4.2.1, Section 7].
- Run a full simulation.

- Select a path from the start state to the end state (mind the order in which the states are selected) [D4.2.1, Section 8].
- Look at the value history view [D4.2.1, Section 9.2].
- Do the previous two steps for each of the possible paths (if multiple paths exist).

Step 7: Debugging Suggestions

Debugging questions:

- Are the values in the first state in accordance with what has been defined in the initial scenario?
- Does the *Growth rate* have a value?
- Do both the *Size* and the *Shade* have a value?
- Do *Size* and the *Shade* have the same qualitative value?
- Is the number of states similar to the expected amount mentioned in Step 6?
- Do all quantities have their expected values in all states?

Suggestions:

- If *Shade* does get values in each of the states, but if those values are different compared to the values generated for *Size*, maybe a correspondence should be defined between the two quantities in the model fragment 'Tree with shade'.
- If *Size* does not increase in the first state, check whether the influence between the *Growth rate* and the *Size* is defined appropriately. Also check whether the value assigned to the *Growth rate* is actually positive (that is: *Growth rate* >0).
- If *Shade* does not increase in the first state, check whether the proportionality between *Size* and *Shade* is defined appropriately. Also check whether *Size* is actually increasing, because it should before *Shade* can increase. Also check whether *Shade* has a value, because it should have a value defined in the scenario before it can increase.

Do the following:

- Discuss which of the debugging options is needed to improve the model.

Step 8: Fixing the Model

Do the following:

- Open the 'Tree with shade' model fragment in the model fragment editor.
- Create a 'directed quantity space correspondence' from the quantity space of *Size* to the quantity space of *Shade.* Notice that a quantity space correspondence is created between the names of the involved quantity spaces.
- Close the model fragment editor.

Step 9: Interpreting the Results, Debugging, and further Issues

In each of the following step, try to understand what is shown, and discuss this with your partner.

- Select the Path from the start state to the end state (mind the order).
- Look at the Value history View.
- Choose one of the states.
- Open the Entity-Relations View, and look at the structural model.
- Open the Model Fragments View and look at the active model fragments.
- Open the Dependencies View and make it show only the quantities and the causal relations (I's and P's).
- While being in the Dependencies View: make all 'the context' appear.

Answer question:

- What is the relationship between the contents of the model fragments in the build environment, and the contents of the Dependencies View mentioned above?

Step 10: Additional Issues

- It is possible to create alternative scenarios by specifying another initial value for the quantity *Size*. Try to run these scenarios, and compare the results.
- Remove the influence from the 'Growth of tree' model fragment, and simulate the result. Look at the Value History. What is the reason only one state is generated? What is wrong with the Value History? Restore the influence.
- Remove the proportionality from the 'Tree with shade' model fragment, and simulate the result. Explain why only one state is generated. Restore the proportionality.

B.2: Population Behaviour

The assignments presented below closely follow the models discussed in Chapter 4. The main goal of these assignments is to further familiarise the reader with the different kinds of building blocks that constitute a qualitative model. Assignment 1 deals with the initial organisation of this population model. Assignments 2-5 introduce additional details and alternative representations.

Basic Population Growth

The goal is to create a model that captures the idea of a population and its growth due to birth. Essentially three ideas should be represented in the model:

- Existing populations grow.
- Growing means that the number of individuals increases.
- When the number of individuals increases, the biomass increases.

The population is thus characterised by three quantities: *Number_of*, *Biomass* and *Birth*. Notice, the close resemblance between these three quantities and the quantities involves in the ‘Tree & Shade’ system discussed above. For both systems there is a central variable (*Size* and *Number_of*) that designates the typical situations in which the system can be. Such a quantity is sometimes called the *state variable*. Second, there is a *rate* that directly (in this case positively) influences the state variable (*Growth* and *Birth*). Finally, there is a dependent variable that follows the changes in the state variable (*Shade* en *Biomass*). In fact, the two systems can be represented following the same general principles. As a result, the ‘Basic population growth’ model can be obtained by ‘editing’ the names of the model ingredients in the ‘Tree & Shade’ model. The following changes are required to accomplish this:

- Entity hierarchy:
  - Entity type (1 change).
- Scenario:
  - Name (1 change), entity instance (1 change), quantities (2 changes).
- Model fragments:
  - Name (2 changes), entity type (2 changes), quantities (4 changes).

The model should consist of two model fragments: a static fragment that describes the relationship between *Number_of* and *Biomass*, and a process fragment implementing the influence of *Birth*. There can be multiple scenarios with varying initial values for the quantity *Number_of*. Simulating the model should produce results similar to those of the ‘Tree & Shade’ model. Notice that the exact number of states that will be generated depends upon the initial value set for the quantity *Number_of*.

Competing Processes

Populations not only grow. They may also shrink and become extinct due to dying. The goal of this assignment is to include 'Death' as a second process affecting the population. This process is kind of analogous to the 'Birth' process. A quantity *Death* is introduced, with a negative influence on *Number_of*.

Whilst running the simulation take notice the following:

- 'Death' and 'Birth' are competing processes resulting in ambiguity. Given a certain value for *Number_of* the population may increase, stay steady, or decrease.
- The population may become extinct. *Number_of* and *Biomass* should thus go to zero in one of the behaviour paths.
- How many states are generated? And how many behaviours are there?

An Alternative Representation

There are different ways in which 'the same' idea can be represented in a model. The goal of this assignment is to take an alternative representation, by applying the following principles:

- Introduce the quantities of interest as early as possible. Thus in addition to *Number_of* and *Biomass*, also introduce *Birth* and *Death* as quantities in the static model fragment describing population features.
- Make the processes conditional: the 'Birth' and 'Death' processes should only become active when:
    - the static model fragment describing the population features has been found *and*
    - the population exists, that is: *Number_of* > *zero*.

In addition to the alternative representation, also add feedback to this model. The idea is that the *Birth* and *Death* rates increase (or decrease) following an increase (or decrease) in the *Number_of*. This requires positive proportionalities between those quantities in the static model fragment describing the population features. One of the results of this approach is that the rates will change, and among others, *Birth* and *Death* will also go to zero when the *Number_of* goes to zero.

An Agent for Immigration

In the models created so far, a population cannot 'recover' from zero - from being extinct. The goal of this assignment is to add immigration. Consider the population discussed in the previous assignments as the system we are dealing with. Immigration can then be seen as an external influence affecting that system, namely, individuals from a different area entering the area in which this population lives. External influences are modelled using model fragments of type agent. Thus: create an agent that causes immigration to the system (population):

- Define an agent (external influence) using the 'agent hierarchy editor' (e.g. *Neighbour)*.
- Define the quantity *Immigration*.
- Adapt the structural details in the scenario. Introduce the idea of an area (e.g. an *Ecosystem*) in which a population lives while another population (e.g. the *Neighbour*) lives outside of this area.
- Create a model fragment (type: agent) in which (for the structural situation mentioned above) the quantity *Immigration* has a positive influence on the quantity *Number_of* and as such implements the idea of individuals immigrating

into the ecosystem. (Ignore the details of the neighbour population in terms it becoming smaller due to individuals leaving).

When simulating the model, run a simulation in which the *Number_of* starts at *Zero*. This should result in a state-graph in which this population starts growing from *Zero* and reaches it maximum size. Also run a simulation in which *Number_of* starts at a value higher than *Zero*. Does this situation lead to the population becoming extinct (thus *Number_of* going to *Zero*)? Why not? An additional assignment could be to adapt the model such that it does go to *Zero*, e.g. by including an emigration process.

Negative, Neutral and Positive growth

This assignment is intended to get the reader acquainted with the assumption mechanism in Garp3 (not to be confused with the notion of 'using assumptions', which refers to using additional labels to include or exclude model fragments). The assumption mechanism in Garp3 is always active. It is concerned with inequality reasoning. Particularly, in the situation that an inequality cannot be proven inconsistent, the engine 'assumes' that the inequality is true. This assumption stays active until counter evidence is found, in that case all inferences depending on the assumption are considered incorrect (which may lead to states not being generated).

The Garp3 engine assumption mechanism can be used to automatically try all possible values for quantities by creating model fragments with conditional value assignments for each value of such quantities. In this assignment this mechanism is used to generate all values for the quantity *Growth*. Proceed as follows:

- Use only one rate: the quantity *Growth* (and do not use *Birth*, *Death*, *Immigration* nor *Emigration*). *Growth* can take on three values: {min, zero, pos}, referring to decrease, steady and increase (notice that *zero* should be a point).
- Define a process (e.g. named 'Growth') in which the quantity *Growth* has a positive influence on *Number_of* (thus: when *Growth* has value *min*, *Number_of* will decrease, with value *zero* it will stay steady, and with value *plus* it will increase).
- Create three subtypes of this process model fragment, namely 'Negative growth', 'Neutral growth' (i.e., no change), and 'Positive growth'. Each of these fragments as a conditional value statement concerning *Growth*, namely *Growth<zero*, *Growth=zero*, and *Growth>zero*, respectively.

When simulating the model, run a simulation in which the *Number_of* starts at a value higher than *Zero*. Does *Number_of* go to *Zero* in this simulation? If not, why? Hint: is there feedback from *Number_of* on *Growth.* See also previous assignments.

B.3: Communicating Vessels

Construct a (simple) qualitative model that captures the typical behaviour of communicating vessels (also referred to as U-tube systems), using a QPT oriented approach.

Take the following details into account:
The height of the liquid column determines the pressure at the bottom of the container. The amount of liquid determines the height of the column. The latter also depends on the width of the column, but when the containers have the same shape this fact can be ignored (assume the containers are equal). When two containers are (partially) filled with liquid and connected by a pipe near the bottoms of the containers, the pressure difference (at the bottoms) between the two liquid columns determines the flow of liquid

between the two containers (via the pipe by which they are connected). The flow changes the amount of liquid in the containers. The container with the highest liquid column (and thus, the highest pressure), will loose liquid, whereas the amount of liquid in the other container will increase. The changes in the amount will eventually change the pressures at the bottoms of the two containers: they will become equal. That is, there is no pressure difference and thus no flow.

Proceed as follows

- Construct the model ingredients using paper + pencil.
- Define the expected states of behaviour and transitions. This is to illustrate how you think the simulator will simulate behaviour using your specifications.
- Implement the model using the Garp3 software. Your model should at least include:
    - A scenario with unequal liquid column heights.
    - A static model fragment describing the behaviour of a 'contained liquid' (the features of a container containing liquid).
    - A process model fragment describing a liquid flow process between two 'contained liquids'.
    - When simulating the model the unequal column heights should become equal in one of the successor states. Notice that the simplest model will only produce two states of behaviour, but more complex state-graphs are also possible.

B.4: Heating & Boiling

Construct a model of a heater that heats a container containing water. Different options exist:

- *Open or closed container:* The container can be open at the top allowing the water to evaporate without seriously affecting the pressure of the container. Or the container can be closed and heating causes the pressure inside the container to increase, which may lead to the container exploding.
- *Relative temperatures:* The temperature of the heater may differ with respect to the boiling point of the water. The temperature can be below, equal or higher compared to this boiling point.
- *Kind of heat source:* The heat source could be another object with a certain temperature and a certain amount of heat that exchanges energy with the container containing water. Alternatively, the heat source could be a kind ‘infinite’ resource (e.g. a furnace) that keeps on generating energy.

Sections 4.5 and 4.6 present both models with an open container. In Section 4.5 the heat source is considered an external and ‘endless’ source of energy with a temperature higher than the boiling point of the water. In Section 4.6 the heat source is considered an object with a finite amount of energy, while the relative temperatures are unknown (that is: all options are taken into account during simulation).