Student Modeling and Domain Models: Instruments in the Process of Acquiring Conceptual Knowledge

Uroš Milošević
Student No. 5758238

Master's thesis
MSc in Artificial Intelligence
Masterschool of Informatics
Faculty of Science
University of Amsterdam

Supervisor: Bert Bredeweg
Informatics Institute
Faculty of Science
University of Amsterdam

UNIVERSITEIT VAN AMSTERDAM

Acknowledgements

The research presented in this MSc Thesis is co-funded by the EC within FP7 (20092012) (project DynaLearn, 231526, http://www.DynaLearn.eu).

I would like to thank:
Bert Bredeweg for his guidance, patience and understanding.
Paulo Salles and Tim Nuttle for their time, effort, and all the advice and inspiration.
Jochem Liem, Floris Linnebank and Michael Wißner for their help and all the overtime spent debugging.
My family and my friends, for everything else I ever needed.

Abstract

The DynaLearn project is an effort to develop an Interactive Learning Environment (ILE) that relies on Qualitative Reasoning methods to assist students in constructing their conceptual system knowledge, either individually or in a collaborative setting. Such an environment must have the means to assess a student's knowledge state during this process. In order to do so, the system must be able to understand first how this knowledge is constructed and understood from the user's perspective. The goal of this research is to learn what it takes to get the DynaLearn ILE in the process of students understanding and building qualitative knowledge of system behavior. We first take the perspective of the learner and go through the process of model building, tackling the domain of global warming amplifiers. Then, we learn how to construct a model of the student's knowledge based on Bayesian networks. We provide solutions for some of the existing issues related to constructing Bayesian network based learner models in the context of Qualitative Reasoning, prove the applicability of the approach and propose future work.

Contents

1 Introduction **6**

1.1 Problem context 6

1.2 DynaLearn - learning by modeling 7

1.2.1 Global design 8

2 Qualitative Reasoning **10**

2.1 Qualitative Reasoning and Interactive Learning Environments 10

2.2 Representing qualitative knowledge 10

2.2.1 Values and changes 10

2.2.2 Causality 11

2.2.3 Simulations 11

2.2.4 Domains and libraries 11

2.3 Garp3 12

2.4 Basic model ingredients 12

2.5 Aggregates 12

2.6 A working example - Tree and shade 12

2.6.1 Entities, agents and assumptions 13

2.6.2 Quantities and quantity spaces 13

2.6.3 Model fragments 13

2.6.4 Scenarios 14

2.6.5 Simulation results 14

3 Qualitative Models of Global Warming Amplifiers **17**

3.1 Climate change 17
3.2 Global warming amplifiers 18
3.2.1 Planetary radiation balance and greenhouse effect 18
3.2.2 Modeling choice: entity hierarchies 19
3.3 Snow and ice cover 19
3.3.1 Entities and agents 20
3.3.2 Assumptions 20
3.3.3 Quantities and quantity spaces 20
3.3.4 Model fragments 21
3.3.5 Scenarios 22
3.3.6 Simulation results 23
3.4 Cooling aerosols 24
3.4.1 Entities and agents 25
3.4.2 Assumptions 26
3.4.3 Quantities and quantity spaces 26
3.4.4 Model fragments 26
3.4.5 Scenarios 26
3.4.6 Simulation results 27
3.5 Water vapor 27
3.5.1 Entities and agents 28
3.5.2 Assumptions 28
3.5.3 Quantities and quantity spaces 28
3.5.4 Model fragments 29
3.5.5 Scenarios 30
3.5.6 Simulation results 31
3.6 Warming aerosols 31
3.6.1 Entities and agents 32
3.6.2 Assumptions 32
3.6.3 Quantities and quantity spaces 33
3.6.4 Model fragments 33

3.6.5 Scenarios . . . 34
3.6.6 Simulation results . . . 34
3.7 Low and high clouds . . . 35
3.7.1 Entities and agents . . . 36
3.7.2 Assumptions . . . 37
3.7.3 Quantities and quantity spaces . . . 37
3.7.4 Model fragments . . . 38
3.7.5 Scenarios . . . 39
3.7.6 Simulation results . . . 39
3.8 Evaluation by experts . . . 40
3.8.1 Evaluation by expert 1 . . . 41
3.8.2 Evaluation by expert 2 . . . 44
3.9 Conclusion . . . 45

4 Bayesian Network based Learner Modeling 46
4.1 Probability theory . . . 47
4.1.1 Conditional and unconditional probability . . . 47
4.1.2 Sum rule and product rule . . . 48
4.1.3 Bayes rule . . . 48
4.2 Bayesian networks . . . 49
4.2.1 Dynamic Bayesian networks . . . 51
4.3 Bayesian networks and learner models . . . 52
4.3.1 Knowledge tracing . . . 52
4.3.2 Student modeling based on belief networks . . . 53
4.3.3 Bayesian knowledge tracing model issues . . . 60

5 From Garp3 Models to Bayesian Network Based Learner Models 62
5.1 Question generation . . . 62
5.2 A learner model based on a Bayesian network for Garp3 . . . 63
5.3 Global architecture . . . 65
5.4 Individual scenarios . . . 67

5.4.1 QUAGS, questions and answers . . . 68
5.4.2 Dealing with mathematical dependencies . . . 68
5.4.3 Augmenting the superstate . . . 69
5.4.4 Recurring substructures . . . 70
5.4.5 Options for updating knowledge . . . 71
5.5 Final Bayesian network structure and evaluation . . . 75
5.5.1 Communicating vessels model fragments . . . 76
5.5.2 Communicating vessels scenario . . . 77
5.5.3 Final Bayesian network structure . . . 78
5.5.4 Choice of parameters . . . 79
5.5.5 Evaluation . . . 80

6 Conclusion **84**
6.1 Discussion . . . 84
6.2 Future work . . . 85

Appendix:

A Sharing and Reusing Qualitative Reasoning Knowledge **90**
A.1 OWL . . . 90
A.2 Qualitative Reasoning models and simulations in OWL . . . 93
A.2.1 Representing models . . . 93
A.2.2 Representing simulations . . . 93

B Communicating Vessels: The Bayesian Network **96**

Chapter 1

Introduction

1.1 Problem context

Qualitative Reasoning originates from *Artificial Intelligence*(AI), and provides the tools for constructing conceptual models of systems. More specifically, it provides formal means to externalize thought on notions such as the physical system structure, causality, the start and end of processes, the assumptions and conditions under which facts are true, etc. (Bredeweg et al., 2006, 2008, 2009c). The *DynaLearn*[1] project is an effort to develop an *Interactive Learning Environment* (ILE) that relies on Qualitative Reasoning methods to aid students in learning (Bredeweg et al., 2009b). Learners will be given an opportunity to use diagrammatic representations to articulate, analyze and communicate ideas in an easy to use environment for constructing their conceptual knowledge. Learners working on similar ideas will be able to collaborate thanks to ontology mappings, which will allow for individualized and mutually benefiting learning opportunities. Interactions with the learning environment will be made more engaging and motivating through the use of Virtual characters.

However, in order for the DynaLearn learning environment to be able to help a student develop their knowledge, it must be able to assess their knowledge state during this process.. In order to assess a learner's state of knowledge, the system must understand first how this knowledge is constructed and understood from the user's perspective. The goal of this research is to learn what it takes to get the DynaLearn ILE in the process of students understanding and building qualitative knowledge of system behavior. Therefore, we will first take the perspective of the learner and go through the process of model building. Then, we will learn how to construct a model of the student's knowledge based on a probabilistic approach, namely, *Bayesian networks* (Mitchell, 1997; Bishop, 2009).

We will first familiarize with the domain of Qualitative Reasoning, that is, its essential ingredients, but also *Garp3*, an environment that facilitates building conceptual knowledge

[1] http://www.DynaLearn.eu

models (Bredeweg et al., 2009c). As there is growing interest from ecological experts to create qualitative models of phenomena for which numerical information is sparse or missing (Salles and Bredeweg, 2006; Cioaca et al., 2009)., in the first part of this thesis, we will shift to the Qualitative Reasoning paradigm by seeing how QR can be used to model systems in the field of *Environmental Science*, namely, the domain of *global warming* (Allaby, 2000; Philander, 2008; Lerner and Lerner, 2009). The second part of the research builds on the work of Brielmann (2009), where a learner model initially proposed by Corbett and Anderson (1995), and later formalized and implemented in terms of a Bayesian network by Reye (2004) is used to construct the system's view of a learner's state of knowledge about a QR model.

The DynaLearn project, its purpose, the overall approach and the architecture will be discussed below.

1.2 DynaLearn - learning by modeling

Constructing conceptual knowledge of system's behavior is of great importance for understanding and successfully interacting with the environment. Moreover, cognitive science research has shown that students learn about system behavior best when they have a clear mental picture/model of how the system functions, i.e. they can distinguish the system from the surrounding environment, identify its building parts, and predict and explain its behaviors (Bredeweg et al., 2006).

The past three decades have given us a number modeling environments, starting with Papert's Mindstorms (1980), and then moving forward with StarLogo (Resnick, 1994) (later NetLogo), Stella (Richmond and Peterson, 1992), and Model-It (Jackson et al., 1998). These innovative environments were among the first to offer students the possibility to construct their own simulations to solve different problems, allowing them to examine new levels of complexity, and look at previously hidden details (Bredeweg et al., 2009b). However, despite the value for learning, this technology for handling qualitative knowledge couldn't overcome a number of issues. Bredeweg and Winkels (1998) give a full list of arguments; we will mention only the most important ones. First and foremost, the fact that the underlying representation is quantitative, makes these systems useless when numerical information is not available. Moreover, dealing with numbers distracts learners from focusing on developing their conceptual understanding of how systems work. Second, (Forbus, 1984) argues that many crucial conceptual notions are not explicitly represented in such quantitative approaches, such as landmark values, causality, qualitative distinct states of behavior, processes, etc. This prevents the learners from using the appropriate language to develop their knowledge, thereby resulting in suboptimal learning. Also, when certain essential notions are not captured by the underlying representation, creating interactive tools that teach learners the key conceptual insights that explain system behavior is difficult. In other words, the automated feedback that can be provided is suboptimal.

The DynaLearn project is an effort to develop an individualized and engaging cognitive tool - an interactive learning environment that will allow the learners to acquire conceptual system knowledge, either individually or in a collaborative setting (Bredeweg et al., 2009a,b). The three main characteristics/goals of the workbench are:

- Accommodate the true nature of conceptual knowledge
- Be engaging by using personified agent technology
- React to the individual knowledge needs of learners

1.2.1 Global design

All of the above mentioned characteristics should be covered by the the DynaLearn software's main components: Semantic Technology (ST; allows for searching for and recommending related models), Conceptual Modeling (CM; facilitates capturing of conceptual knowledge about system behavior, and allows for using that knowledge for simulations) and Virtual Characters (VC; engage the learner by being interactive in a knowledgeable way and by providing support). (Figure 1.1)

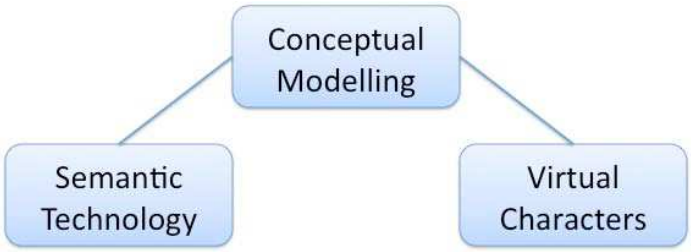


Figure 1.1: DynaLearn software main components (Bredeweg et al., 2009a)

Chapters 2 and 3 give an insight into the CM component. The former introduces *Garp3*, a conceptual modeling environment, whereas the latter covers a set of five qualitative models from the domain of Environmental Science, built using this workbench. The ST component is not thoroughly covered by this thesis; Chapter 3, only briefly discusses Cyc[2], a proposal for term grounding, and Appendix A describes OWL, a language for sharing and reusing qualitative knowledge, but mainly for the purpose of constructing learner models. Finally, chapters 4 and 5 discuss tracking and updating a student's state of knowledge during an examination session using a Bayesian network based learner model, thereby covering both the CM and VC components. A more detailed organization of the report is given below.

We will first explain the essentials of Qualitative Reasoning, and *Garp3*, a workbench for building, simulating, and inspecting qualitative models (Chapter 2). In Chapter 3, the author takes a qualitative view on the popular domain in Environmental Science of global warming. More specifically, we will see Qualitative Reasoning at work in an attempt to tackle the domain of *global warming amplifiers*. The models get reviewed by two field experts and conclusions are drawn at the end of the chapter. In Chapter 4, we get acquainted with Bayesian networks, and review the existing work in the field of *Learner Modeling*. This chapter also serves as an introduction to Chapter 5, where we describe our own approach

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

to modeling and updating the student's state of knowledge of domains described in Garp3, using a *Dynamic Bayesian network*. Finally, we sum up the results, discuss the outcome of the work, and suggest further improvements. Appendix A explains the low level details of Garp3 models, namely, their representation in *OWL*, the *Web Ontology Language* (Horrocks et al., 2003), and Appendix B shows the Bayesian network structure for *Communicating vessels*, a QR model we use to evaluate our approach in Chapter 5.

Chapter 2

Qualitative Reasoning

2.1 Qualitative Reasoning and Interactive Learning Environments

The means for capturing continuous aspects of the world, such as space, time, and quantity are provided by Qualitative Reasoning, an area of Artificial Intelligence that supports reasoning with very little information, i.e. without numerical data (Forbus, 1997). A reason to abandon quantitative data when modeling a system's behavior is that this kind of data can often be missing, which cancels out the applicability of a pure mathematical approach. What's more important is that these models can't capture causality or the model's structure properly (Salles and Bredeweg, 2006). This makes interpreting and explaining a model, or comparing alternative models difficult. In other words, the usability of the model in question is directly affected. Qualitative Reasoning, on the other hand, doesn't suffer from such deficiencies. A property of qualitative models that is crucial for DynaLearn, or education and training in general, is the fact that usable conceptual models make "knowledge communication" between the agents involved in the learning process possible.

The link between QR and education or, more specifically, ILEs starts with some of the earliest work on QR, which focuses on automatic explanation generation in the context of environments that facilitate interactive learning (Brown et al., 1982; Hollan et al., 1984). Also, some of the first approaches to computer based conceptual analysis of system behavior (Bobrow, 1984) gave rise to the idea of using *qualitative* models and simulations, also known as *articulate simulations* (Forbus, 1988; Bredeweg and Winkels, 1998).

2.2 Representing qualitative knowledge

2.2.1 Values and changes

The power of Qualitative Reasoning models lies in the ability to capture both system's structural and behavioral information. Quantitative information is abstracted and represented via ordered sets of qualitative values. Each set's elements are usually alternating points

and intervals (also known as *magnitudes*), forming the so called *quantity space*. The points represent landmarks that refer to situations in which system behavior changes significantly (e.g. a substance's boiling point - the substance will stop getting hotter and start boiling). Quantities can be assigned quantity space values, but this isn't enough to capture the behavior changes within a system. To overcome this obstacle, qualitative *derivatives* are used for each quantity, showing the direction of a change, i.e. whether a quantity is *steady*, *decreasing* or *increasing*.

2.2.2 Causality

As already mentioned earlier, QR models can also capture the notion of causality. Different (types of modeling) primitives have been suggested, each of them having a specific and formal meaning and calculus, so they can be implemented in computer programs. This work has laid down the foundation for further research and advancements in the field (Bredeweg et al., 2006).

Causal information within a system is represented via a set of dependencies that is defined so that it can both closely match human reasoning, and be grounded in mathematical formalisms that facilitate automated computation. The most typical examples are illustrated in the notions of direct and indirect influences. The *direct influences* represent changes initiated by processes, whereas the indirect ones, also known as *proportionalities*, show the causal propagation of these changes to other quantities. In quantitative terms, they represent the ordinary differential equations and monotonic functions, respectively (Bredeweg et al., 2009c).

2.2.3 Simulations

The behavior of a system over a period of time is shown in the results of qualitative *simulations*, where each state in the *simulation state graph* represents a qualitatively distinct moment in time. The states don't hold the information about their duration. Quantity magnitudes changes (increases or decreases) cause state transitions. In case of ambiguity, as in the case of competing influences, when the information on the magnitude of one influence with respect to the other influence is unknown, Qualitative Reasoning engines generate all possible solutions.

2.2.4 Domains and libraries

A fundamental aspect of using this technology is capturing knowledge from a certain domain in model fragments and combining them to build libraries. By using these libraries, QR engines automatically generate qualitative models of systems belonging to the domain. A lot of effort has already been put into covering the domains of physics and engineering. Some of the successful application areas include autonomous spacecraft support, failure analysis and on-board diagnosis of vehicle systems, automated generation of control software for photocopiers, and intelligent aids for learning about thermodynamic cycles as well as for learning about other systems and phenomena (Bredeweg et al., 2009c). Lately, however, the

domain of environmental science is being tackled more and more (Bredeweg et al., 2006; Salles and Bredeweg, 2006; Cioaca et al., 2009).

2.3 Garp3

In order to preserve the full expressiveness of the above mentioned Qualitative Reasoning formalisms, but also be able to address domain experts and support them in articulating and capturing their conceptual knowledge, Garp3 was developed. It is an easy to use workbench that allows for building, simulating, and inspecting qualitative knowledge (Bredeweg et al., 2009c). The software package allows the modelers to represent their conceptual knowledge of system behavior in a user-friendly graphical (diagrammatic) environment.

In Garp3, knowledge is built using model ingredients and aggregates that can be built using the basic ingredients.

2.4 Basic model ingredients

Entities, which can be arranged in a subtype hierarchy, are used to represent the physical objects or abstract concepts the system itself is made of. The relevant properties that can change under the influence of processes are represented as quantities.

Agents represent external entities or processes that may influence the system's behavior. The agent quantities influencing the rest of the system are called *exogenous quantities*.

To indicate that certain condition is presumed to be true, the modeler can use *Assumptions*. Their main use is to constrain the modeled system's behavior.

Finally, *Configurations* represent the structural relations between instances of entities and agents.

2.5 Aggregates

Scenarios are used to represent a specific system state the simulation should start from. They allow the user to inspect system behavior in various real-world or experimental situations.

Model fragments represent individual parts of the user's knowledge about the system. When running a simulation for a given scenario, for each simulation state, the QR engine searches the model fragment (MF) library in order to find the fragments that match the conditions given in that state. The end result is a state graph, which can be expected using multiple views.

2.6 A working example - Tree and shade

This section illustrates details relevant to Qualitative Reasoning modeling using a very simple model that has shown to be successful in teaching elementary QR courses in the past (Bredeweg et al., 2006).

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

Table 2.1: Tree and shade model quantities and relevant quantity spaces

The *Tree and shade* model is supposed to describe the relation between a growing tree and the shade that's being cast on the ground by the tree. In order to simplify the model and focus only on the relevant aspects of the domain, we assume that a tree always grows, and ignore the need for the basic necessities for the growth process, such as water, sunlight, air and minerals. In this dynamic process, as the tree grows bigger, so does the shade. We will go step by step in trying to explain the model essentials.

2.6.1 Entities, agents and assumptions

The entity hierarchy for the *Tree and shade* model contains only one type of entity: *Tree* (Figure 2.1). There are no agents defined in this model. Moreover, in order to make the model as simple a possible, no explicit assumptions are defined either.

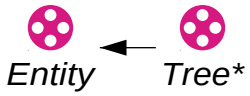


Figure 2.1: Tree and shade model entity hierarchy

2.6.2 Quantities and quantity spaces

The only quantities we need to know about are the size of the tree and the shade, and the rate at which the tree grows. In order to be able to visualize the growth of the tree and its shade, a quantity space with three consecutive values is chosen for both quantities: {small, medium, large}. As the tree either grows or doesn't, the growth rate can take the values of {zero, plus}. This is summarized in Table 2.1.

2.6.3 Model fragments

Our model has two model fragments, one static and one process. The static model fragment, named *Tree with shade*, represents the indirect influence (propagation of change) of *Size* on *Shade* for the Tree entity, as depicted in Figure 2.2. This means that changes in *Size* cause similar changes in *Shade*. Moreover, the sizes of the tree and its shade have corresponding magnitudes, meaning that these quantities always have the same magnitude value (e.g. a big tree indicates a big shade).

The second model fragment, *Growth of tree* is a process showing the direct positive influence of a stable *Growth rate* on the tree size (Figure 2.3). We model the assumption

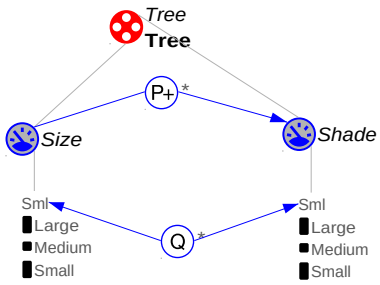


Figure 2.2: Tree and shade model: *Tree with shade* model fragment

that the tree always grows, by assigning the value plus to *Growth rate*. Another way to do this would be to explicitly define an assumption and include it as a condition in this model fragment.

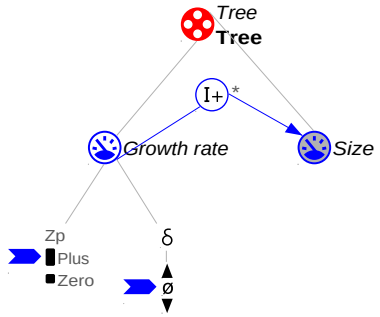


Figure 2.3: Tree and shade model: *Growth of tree* model fragment

2.6.4 Scenarios

A small growing tree scenario has only one quantity defined, namely, *Size*, and indicates that the initial value for the tree size is *small* (Figure 2.4[1]). Simulating this scenario should show a gradual growth of the tree and, as a consequence, its shade.

2.6.5 Simulation results

Finally, we take a look at the simulation results. Figure 2.5 shows the state graph, i.e. the results of simulating "A tree with small shade" scenario. The simulation produces a single path and 3 states.

The quantity value history depicting the quantity values in each of the states can be seen in Figure 2.6. We can see that as the size of the tree increases from *small* to *large*, so does its shade, as expected.

[1]The shaded *Size* icon means some quantity details are hidden. In this case, the derivative is not shown for the sake of simplicity, as it is not s et.

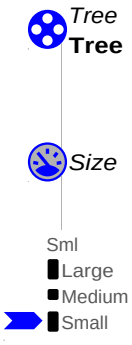


Figure 2.4: Tree and shade model: *A small growing tree* scenario

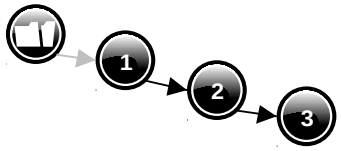


Figure 2.5: Tree and shade model simulation results

As both model fragments are applied in each of the states, we depict the dependencies only for state 3 in Figure 2.7.

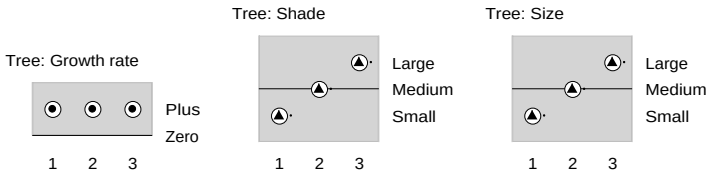


Figure 2.6: Tree and shade model quantity value history

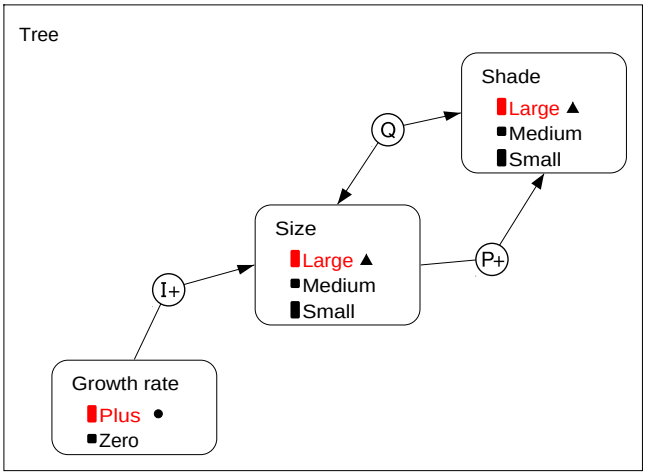


Figure 2.7: Tree and shade model dependencies in state 3

Chapter 3

Qualitative Models of Global Warming Amplifiers

The growing concern about global warming, as well as the impact it is expected it will have on people and the ecosystems on which they depend, have caused the field experts across the globe to turn to finding new methods for modeling, explaining and predicting the important phenomena within the domain of environmental science (Haupt et al., 2008; Philander, 2008; Lerner and Lerner, 2009). In this chapter we will show a number of successful applications of QR modeling techniques, more specifically, via Garp3, the expert tool described in the previous chapter, to the domain of global warming.

As mentioned earlier, in order to assess a student's state of knowledge, the system must understand the user's perspective, i.e. how such knowledge is constructed and understood by the learner. This chapter will give a deeper insight into the Conceptual Modeling component of DynaLearn. The final result is to be contributed to the DynaLearn model repository.

3.1 Climate change

Climate is defined as the average weather of a region over time (Lerner and Lerner, 2009). Some of the most important aspects of climate are temperature, winds, heat waves and cold snaps, rainfall, when seasons begin and end, and similar weather patterns and events. The global machinery of ocean currents, winds, forests, ice caps, mountain ranges, bacteria, planetary orbital motions, and many other factors is what shapes the Earth's weather.

Although the global climate has been changing since the creation of our planet, and even though Earth has experienced sudden ice ages and heat waves, that have happened in as little as a decade, the changes in the past half-century appear to be more drastic than ever before. The important difference is that these changes appear to be coming mostly from increased burning of fossil fuels for energy, industrial processes, and transportation, i.e. human activities. In 2007, the IPCC (Intergovernmental Panel on Climate Change), a large, international panel of experts on the Earth's climate, concluded that human activities, specifically those that cause an increase in the atmospheric concentration of carbon dioxide,

have started affecting the Earth's climate (Philander, 2008). Since 1906, the temperatures have already risen 0.77° C, with much of this warming occurring in just the last 3 decades. Moreover, the temperatures will likely rise at least another 1.1° C, and possibly more than 6.1° C, over the next 100 years (Lerner and Lerner, 2009; Staudt et al., 2009). This warming is expected to cause significant changes in sea level, ecosystems, and ice cover, among other impacts. The landscape and ecosystems are already changing rapidly in the Arctic, where temperatures have increased almost twice as much as the global average.

3.2 Global warming amplifiers

Factors that can amplify or reduce the effect of the causes of climate change are known as *feedbacks*. More specifically, those factors that amplify the effect of global warming are called *positive feedbacks*, whereas the ones that reduce it are known as *negative feedbacks*. They consist of interconnected processes in which a change in one feedback leads to a change in another, which ultimately leads to further changes in the first factor. The idea for modeling these factors came from a careful inspection of 28 models developed by the students at the University of Amsterdam for a course in Qualitative Reasoning by Bert Bredeweg, as well as a Marian Koshland Science Museum's (Washington, USA) exhibition titled *Global Warming Facts and Our Future*[1]. Four factors listed as the key feedbacks by the latter, will be the focus of this section:

1. Snow and ice cover
2. Water vapor
3. Cooling and warming aerosols
4. Low and high clouds

In the following sections, we will see how each of these amplifiers affects the global climate, and then see if we can abstract a general mechanism for each of the feedback types.

The models provide abstract explanations of the above outlined phenomena, and, as such, are not intended for field experts. Instead, a possible end-user could be a student looking to better understand the field of environmental science and the domain of global warming. It is also worth noting that, in approaching the domains from the QR perspective, a modeling framework proposed by Bredeweg et al. (2008) was followed.

3.2.1 Planetary radiation balance and greenhouse effect

The Earth's radiative energy balance depends on the balance between the incoming (short-wave) solar radiation and its absorption by the planet, and subsequent outgoing (longwave)

[1] An online version of the exhibit can be seen at www.koshland-science-museum.org (Last accessed: January 11, 2010)

radiation from the Earth to outer space (Philander, 2008). The Earth emits the absorbed radiation from the Sun back to outer space to maintain its heat energy balance. The greenhouse gases, however, prevent the outgoing longwave radiation from leaving the atmosphere by absorbing it, and then emitting it back toward the surface as shortwave radiation. This "surplus" in the planet's energy "budget" is what leads to global warming (Allaby, 2000). This mechanism plays an important role in our positive feedback models.

3.2.2 Modeling choice: entity hierarchies

It is worth noting, that the entity hierarchies for all of our models contain a number of higher levels that may appear redundant, as well as a naming style that may seem confusing at first. This is because the entities are named after their corresponding concepts within Cyc[2], the world's largest multi-contextual knowledge base and inference engine developed by Cycorp. The entire Cyc ontology, whose domain is all of human consensus reality, contains hundreds of thousands of terms, along with millions of assertions relating the terms to each other.

Cyc concepts are linked to both corresponding DBpedia[3] entries and WordNet[4] synsets. Therefore, enforcing a strong correspondence (i.e. grounding) between the concepts in our QR models and relevant Cyc terms, would pave the way for linking DynaLearn to the existing and ongoing researches, and make it part of a much larger community. Furthermore, the approach should also make model comparison and recommendation mechanisms easier to implement later on and, hence, prove to be useful from a practical point of view as well.

3.3 Snow and ice cover

Freshly fallen snow reflects as much as 80 – 90% of the light falling on it, whereas grass can reflect only 18 – 25 percent (Allaby, 2000). As Allaby (2000) states, "this is why you need dark glasses when crossing snow: it may be almost as bright as the Sun itself."

The *reflection coefficient*, or more usually *albedo* of a surface, is the proportion of light reflected by that surface. It is usually expressed as a fraction or a percentage, as shown above in our snow and grass examples. Albedo varies widely from one surface to another. As reflected radiation does not warm the Earth's surface, the surfaces with a high reflection coefficient, such as the snow and ice cover, have an important climatic effect. To be more precise, snow and ice have a cooling effect on the Earth. This also means that if the global warming reduces the global snow and ice cover, the warming will be enhanced because more solar energy will be absorbed. In relation to our previous discussion, we can also say that the snow and ice cover of the Earth's surface provide a negative feedback mechanism, one that can be easily affected by human intervention.

[2]http://www.cyc.com
[3]http://www.dbpedia.org
[4]http://wordnet.princeton.edu

3.3.1 Entities and agents

As outlined in section 3.2.2, the entity hierarchy shown in Figure 3.1 follows closely the Cyc concept hierarchy for the entities describing our domain. The reader may focus only on the "leaf" nodes, namely, *Theearthsatmosphere*, *Thesun*, *Planetearth* and *Snow and ice cover*. *Theearthsatmosphere* represents two concepts here – that of the lower part of the atmosphere and the Earth's surface. The concept is left general enough to remain clear for all end user levels. *Thesun* is the Earth's Sun, and the source of incoming solar radiation, *Planetearth* is our planet, and *Snow and ice cover*[5] is the portion of the planetary surface covered by snow and ice.

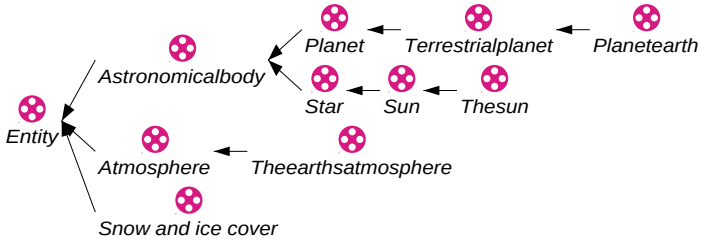


Figure 3.1: Snow and ice cover entity hierarchy

We also have a single agent (Figure 3.2), *Person*, which is the closest Cyc match for human being.

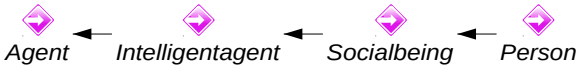


Figure 3.2: Snow and ice cover agent hierarchy

3.3.2 Assumptions

Our first global warming amplifier model, *Snow and ice cover*, contains no explicit assumptions.

3.3.3 Quantities and quantity spaces

The important quantities we need to take into consideration when discussing the domain of snow and ice albedo are given in Table 3.1. We use *Amount* to denote the amount of snow and ice covering the Earth's surface, *Temperature* for the Earth's temperature, and *Solar radiation* to represent the incoming radiation from the Sun. To visualize the changes within these quantities, we use a quantity space with three values ranging from low to high. *Effective radiation* represents the total amount of radiation reaching the Earth's surface, i.e.

[5] At the time of writing this thesis, Cyc was lacking an entry that would correspond to the planetary snow and ice cover. Therefore, a generic name was used.

Quantity	Quantity space
Amount	{low, medium, high}
Temperature	{low, medium, high}
Solar radiation	{low, medium, high}
Effective radiation	{min, zero, plus}
Greenhouse gases	{min, zero, plus}

Table 3.1: Snow and ice cover quantities and relevant quantity spaces

the portion of incoming shortwave radiation that doesn't get reflected back to outer space. A quantity space with a negative value (*min*) is used to be able to show opposite effects on global temperature. Finally, we use a quantity named *Greenhouse gases* for the amount of greenhouse gases emitted by the humans into the atmosphere. As with the effective radiation quantity, we use the {*min, zero, plus*} quantity space, so we can simulate the two-way effect on temperature (and, indirectly, snow and ice cover).

3.3.4 Model fragments

Our first global warming amplifier model contains one static, one process and one agent model fragment. The static model fragment is named *Snow and ice* (Figure 3.3) and shows only that the snow and ice cover amount is indirectly influenced by the Earth's temperature.

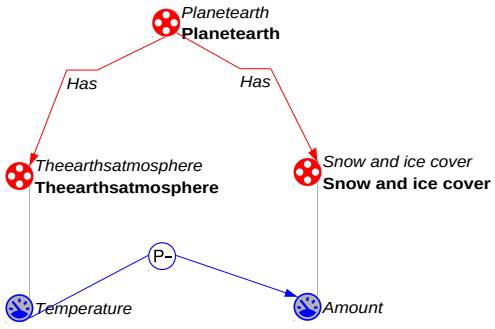


Figure 3.3: Snow and ice cover: Snow and ice model fragment

The process model fragment, *Solar radiation reflectance* defines the incoming shortwave radiation reflectance by snow and ice mechanism.

3.3.4.1 Radiation balance

Although snow and ice work in the opposite way when compared to greenhouse gases, the idea of the Earth's radiative energy balance getting disturbed (as presented in Section 3.2.1) is still present.

One could expect that the above idea implies we could model the planet as a partially closed system affected merely by external factors providing incoming and outgoing radiation. Such an approach would leave out the real idea of balance. That is, it is exactly the process

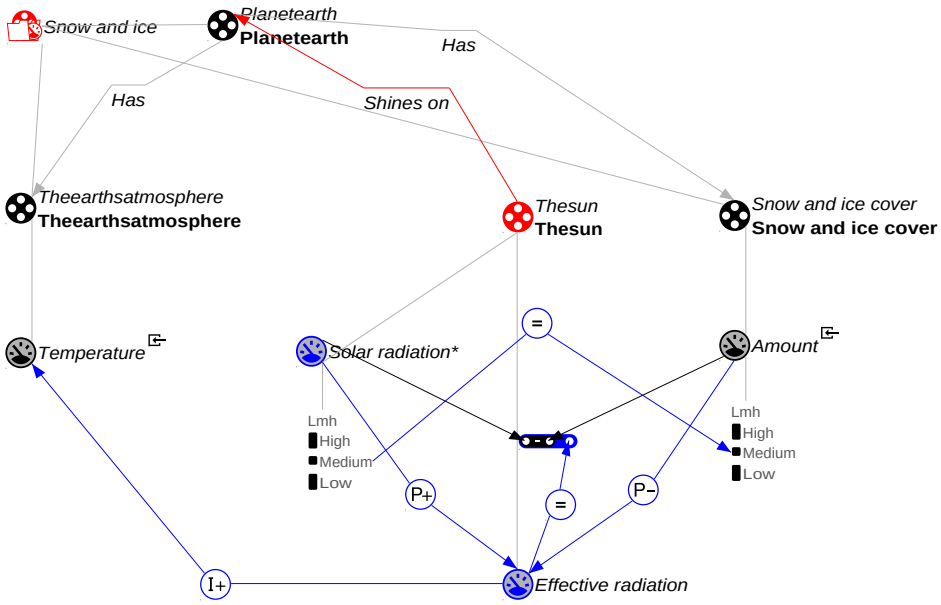


Figure 3.4: Snow and ice cover: Solar radiation reflectance model fragment

of the surplus being created within the system by external factors that we would like to model. This is a key point in all of the global warming amplifier domains.

We define the effective radiation magnitude as the difference between the solar radiation and the amount of snow and ice covering the planet (Figure 3.4). As the terms' quantity spaces don't contain zeros, in order to be able to calculate the difference, we connect the two *medium* points using an equality relation. We also define a positive proportionality from solar radiation and a negative one from snow and ice cover amount to effective radiation, indicating that a potential increase in incoming radiation would result in an increase in effective radiation, and an increase in snow and ice cover would set the effective radiation derivative to decrease. Finally, the Earth's temperature is directly and positively influenced by effective radiation.

The single external (positive) influence on the global temperatures, provided by the greenhouse gases is defined in the *Humans* agent model fragment and shown in Figure 3.5.

3.3.5 Scenarios

The snow and ice cover model contains 5 scenarios: two scenarios showing the opposite effects of solar radiation, two for the same effects of greenhouse gases and one scenario showing the balance between the snow and ice cover and the incoming solar radiation, if there are no disturbances. Figure 3.6 depicts the *Increasing greenhouse gases* scenario, where the amount of snow and solar radiation are set to *medium*, the Earth's temperature to *low* (to clearly show the expected increase in temperature) and the amount of greenhouse gases being emitted into the atmosphere to zero. The solar radiation quantity here is set to be steady (it doesn't change throughout the simulation), but the greenhouse gases are set to increase. This way, the change in the system will depend only on the human influence.

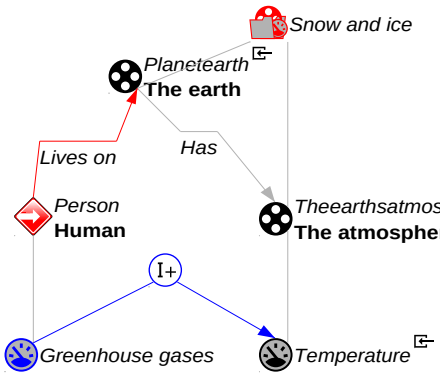


Figure 3.5: Snow and ice cover: Humans (agent) model fragment

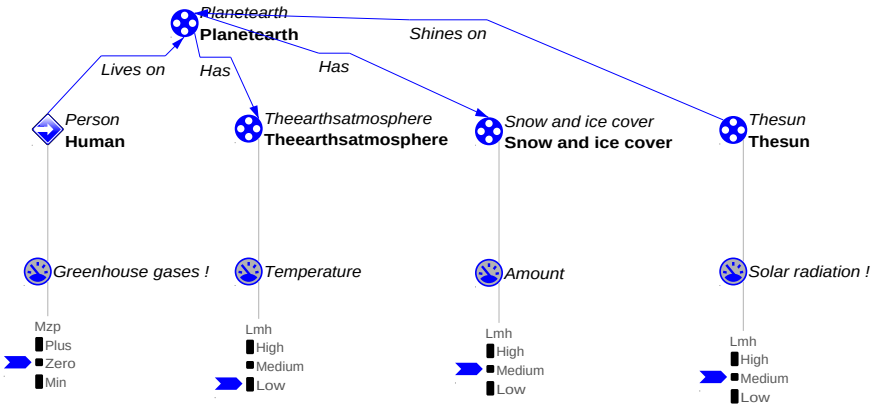


Figure 3.6: Snow and ice cover: Increasing greenhouse gases scenario

3.3.6 Simulation results

Simulating the *Increasing greenhouse gases* scenario results in a linear path state graph containing five states (Figure 3.7).

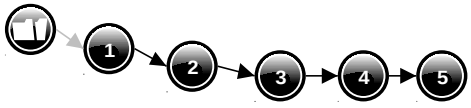


Figure 3.7: Snow and ice cover: Increasing greenhouse gases state graph

The value history presented in Figure 3.8 shows that the magnitude of the planet's temperature increases with an increase in greenhouse gases. That is, as soon as the amount of greenhouse gases enters the *plus* interval (state 2), the balance is disturbed. Moreover, as the temperatures increase, they decrease the snow and ice cover, which further amplifies the increase in temperature as the effective radiation increases, even though the solar radiation is steady (states 2 to 5).

Finally, we give the dependency graph for state 3 of the *Increasing greenhouse gases* scenario simulation in Figure 3.9. The graph shows the dependencies between the domain quantities, as well as the magnitudes. We see that the increasing greenhouse gases have a

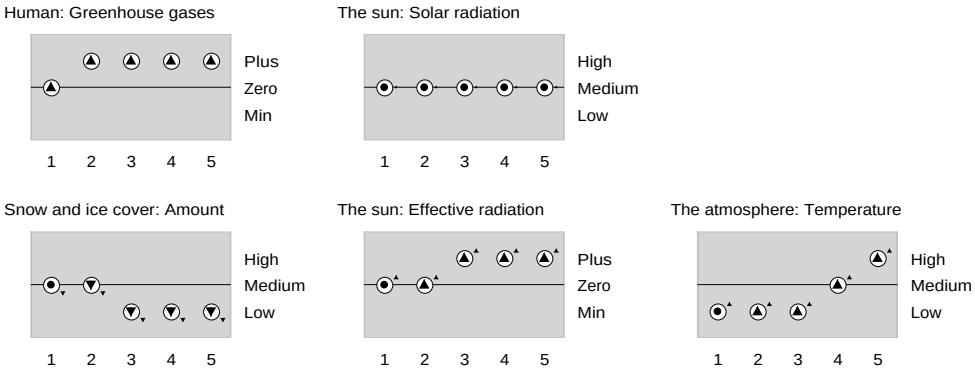


Figure 3.8: Snow and ice cover: Increasing greenhouse gases scenario value history

positive influence on the temperature, which also starts increasing. The negative indirect influence coming from the latter causes the snow and ice cover amount to decrease and enter the *low* interval, thereby disturbing the radiation balance (*Effective radiation* = *medium* − *low* = *plus*), which in turn adds an additional positive influence to the temperature and amplifies the warming.

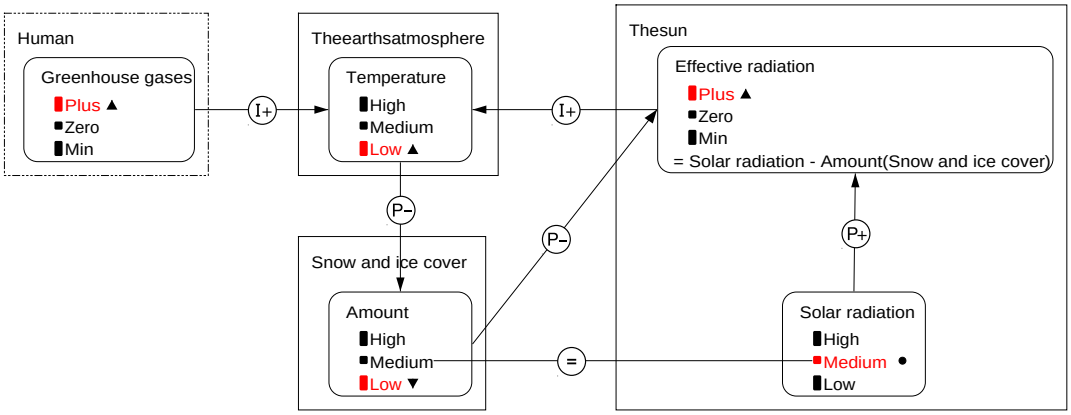


Figure 3.9: Snow and ice cover: Increasing greenhouse gases scenario dependencies for state 3

To show the opposite effect, we once again use a scenario that provides effective radiation disturbance via the same external agent, i.e. greenhouse gases, and, once again, the resulting simulation state graph contains a single path and five states. The value history graph (Figure 3.10) however, shows the expected opposite effect. That is, as the greenhouse gases take the negative value (state 2 again), the temperature drops and causes the snow and ice cover to increase, which, in turn, further cools the Earth (states 2 to 5).

3.4 Cooling aerosols

Aerosols are solid or liquid particles, small enough to remain suspended in the atmosphere up to a number of days (Philander, 2008). Created by both natural and human-caused

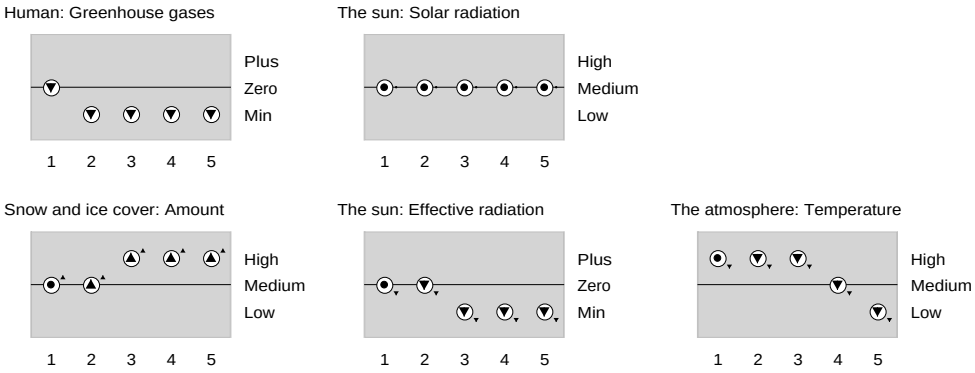


Figure 3.10: Snow and ice cover: Decreasing greenhouse gases scenario value history

factors, they can be both directly emitted from sources and formed in the atmosphere from the condensation of gases. The effect of aerosols on global warming is twofold, depending on their characteristics. For instance, sulfate (SO_4) aerosol is light-colored and reflects solar radiation back into space. The volcanic aerosols produced by the Mt. Tambora eruption of 1815 caused North America's "year without a summer" in 1816 (Philander, 2008).

Although aerosols can affect the Earth's climate both directly, by the scattering or absorption of radiation, and indirectly, through the modification of clouds and precipitation, we will be taking into consideration only the former effect.

3.4.1 Entities and agents

As expected, the entity hierarchy is very similar to that of the first global warming amplifier model (3.3). The only difference is in the bottom graph branch shown in Figure 3.11[6] where the snow and ice cover entity has been replaced by *Cooling aerosols*, a child node of the *Gas* entity.

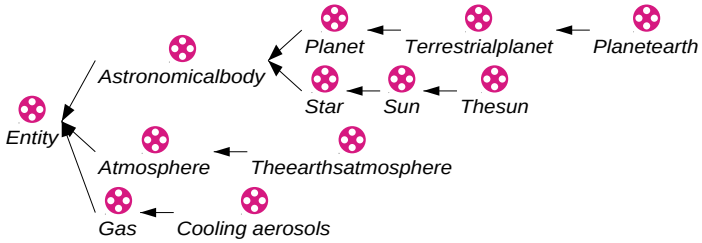


Figure 3.11: Cooling aerosols entity hierarchy

As for the agent hierarchy, it remains the same as in Figure 3.2. A notable difference here is that our agent, i.e. human being has a slightly different role in this model, as it

[6]As with the snow and ice cover model, a generic name was used for the cooling aerosols entity, as no appropriate concept could be found within Cyc's knowledge base.

Quantity	Quantity space
Amount	{low, medium, high}
Temperature	{low, medium, high}
Solar radiation	{low, medium, high}
Effective radiation	{min, zero, plus}
Human activity	{min, zero, plus}

Table 3.2: Cooling aerosols quantities and relevant quantity spaces

is the chief factor controlling the aerosol amount in the atmosphere. As a mere reminder, in our previous model, the snow and ice cover amount was being governed by the Earth's temperature.

3.4.2 Assumptions

The assumption hierarchy has no explicit assumptions defined.

3.4.3 Quantities and quantity spaces

The quantity/quantity space table remains almost the same as in Figure 3.1. The greenhouse gases quantity has been replaced with "human activity" (Table 3.2).

3.4.4 Model fragments

Although in this model the same set of MFs was used as the one described in the Section 3.3.4, the static model fragment this time has no direct or indirect influences defined, as the cooling aerosols have no direct relations to temperature. Instead, they're being controlled by the humans, as shown in Figure 3.12, where there is a direct influence from human activity to the amount of cooling aerosols in the atmosphere. The *Solar radiation reflection* MF matches that of the snow and ice model, being different only with respect to the right hand side of the effective radiation equation, where the *Snow and ice cover* entity has been replaced with *Cooling aerosol* entity (Figure 3.13).

3.4.5 Scenarios

The four scenarios defined are the ones showing the possible opposite effects of the two main factors on the system, i.e. solar radiation and the human activity. However, in the case of the two solar radiation scenarios, we set the cooling aerosol quantity to *steady*, as its magnitude and derivative depend only on external influences, but also in order to be able to see how the temperature behaves when only solar radiation changes. The other scenarios differ with respect to their snow and ice counterparts only in choice of external influences, as explained earlier.

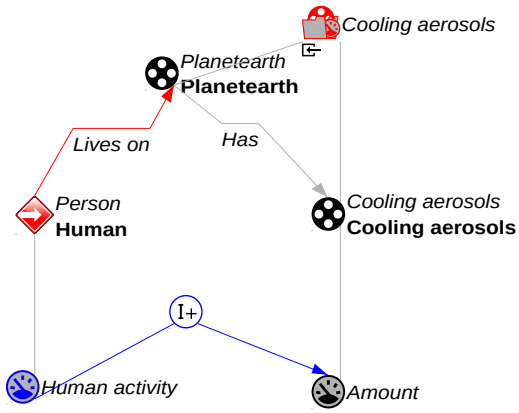


Figure 3.12: Cooling aerosols: Humans (agent) model fragment

3.4.6 Simulation results

The simulation graph for the *Increasing human activity* scenario (Figure 3.14) has one extra state compared to its predecessor. The reason for this can be seen in the value history given in Figure 3.15.

We see that, in the first two states, the temperature is increasing. The reason for this is the fact that the effective radiation is still in the positive interval. Once it hits zero (state 3) the temperature stabilizes, and then finally starts decreasing. This effect is due to the fact that in our second model, there is only one influence affecting the aerosol amount, whereas in the snow and ice model, the temperature itself was a factor, influenced directly by two quantities. This can be further inspected in the dependency view shown in Figure 3.16. *Human activity* increases the amount of cooling aerosols in the atmosphere which has taken the value *medium*. As the magnitude of *Solar radiation* is also set to *medium*, *Effective radiation* equals *zero*, and the positive influence on *Temperature* has no effect. The effective radiation is decreasing, though, due to the negative proportionality relationship coming from the cooling aerosol amount, which is set to increase.

3.5 Water vapor

The Earth's water present in the atmosphere in gaseous form, i.e. water vapor, is the most abundant and most important greenhouse gas on the planet (Lerner and Lerner, 2009). Although the humans are not significantly increasing its concentration, at least not directly, it contributes to the enhanced greenhouse effect, because the warming influence of greenhouse gases leads to a positive water vapor feedback. In other words, if the temperatures were to increase by a modest amount due to the other greenhouse gases, then evaporation from the oceans will increase, which will in turn increase the concentration of water vapor in the atmosphere, thus creating an enhanced greenhouse effect that increases temperatures further, and so on. The eventual runaway greenhouse effect is what is today believed to be the main reason why Venus has no water (Philander, 2008). Being further from the Sun than Venus, and sufficiently cool for the air to become saturated with water vapor (that is,

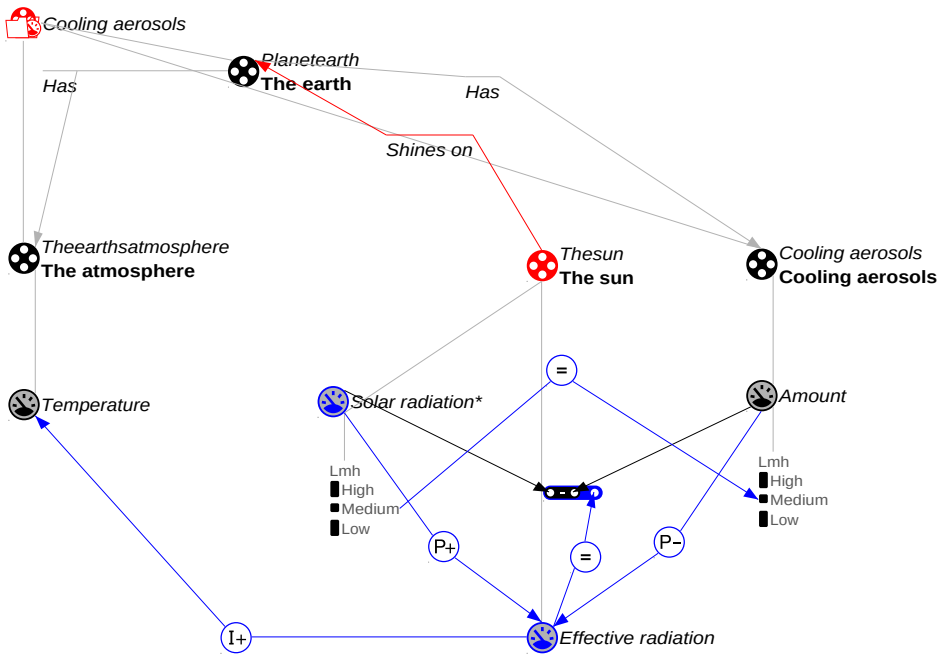


Figure 3.13: Cooling aerosols: Solar radiation reflectance model fragment

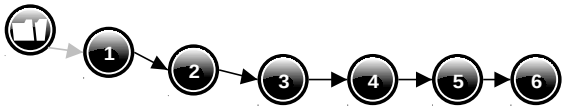


Figure 3.14: Cooling aerosols: Increasing human activity state graph

cool enough for clouds to form), the Earth was spared this fate.

3.5.1 Entities and agents

In comparison with the *Cooling aerosols* model, the leaf node of the bottom branch has been replaced by the *Water vapor* entity. No modifications have been applied to the agent hierarchy (Figure 3.2).

3.5.2 Assumptions

No explicit assumptions are defined in this model.

3.5.3 Quantities and quantity spaces

The new model comes with three new quantities. The *Return radiation* quantity represents the amount of outgoing radiation that gets emitted back by water vapor, as explained in section 3.2.1. *OLR* is the Outgoing Longwave Radiation and *Total effective radiation* is the combined effect of incoming solar radiation and return radiation on the global temperature. For all of the new quantities, we use the same quantity space {*min, zero, plus*}, so we can easily control the temperature (i.e. see it flow in both ways).

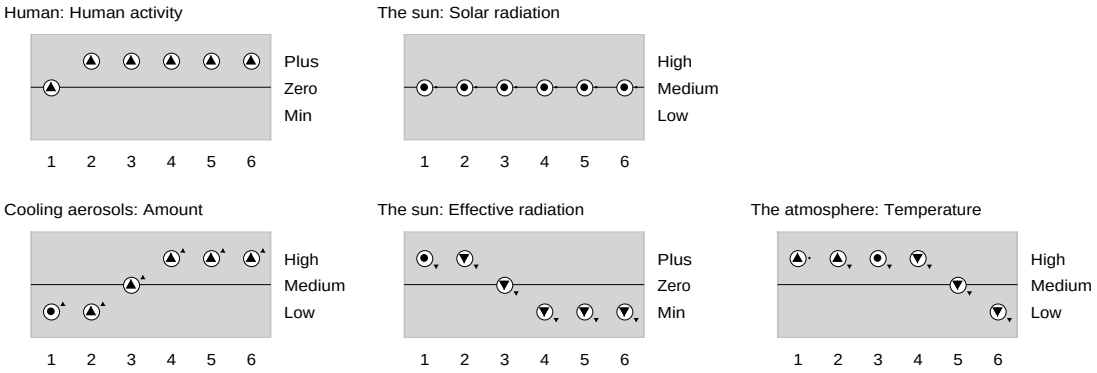


Figure 3.15: Cooling aerosols: Increasing human activity scenario value history

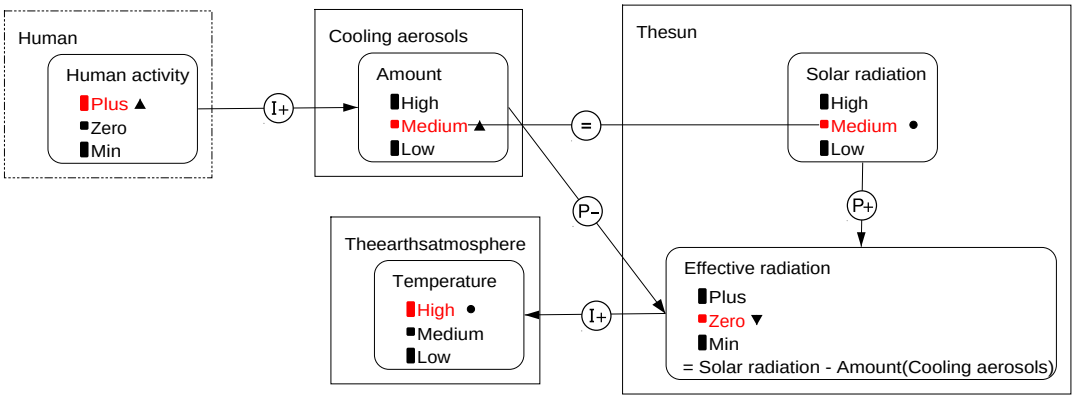


Figure 3.16: Cooling aerosols: Increasing greenhouse gases scenario dependencies for state 3

3.5.4 Model fragments

When compared to the models seen in sections 3.3 and 3.4, the first *positive feedback* model is based on a different mechanism, so its MF hierarchy is also different. We have one static model fragment, two processes and one agent MF. The static fragment (Figure 3.17) defines a one way relation, i.e. positive proportionality between the global temperature and water vapor. It goes only one way, since the water vapor effect on temperature is indirect, as we will see in the following process model fragments.

The *Outgoing Longwave Radiation* and *OLR absorption* model fragments define our planetary radiative energy balance mechanism and the creation of the surplus. The parent process, *Outgoing Longwave Radiation* (OLR) shows how the OLR cools off the Earth (Figure 3.18). The amount of outgoing radiation equals that of the incoming one, and an increase in the incoming solar radiation results in an increase in the outgoing longwave radiation, due to the indirect positive proportionality relationship between the two respective quantities. The negative influence from *OLR* to *Temperature* decreases the global temperature.

The *OLR absorption* model fragment (Figure 3.19) shows the second half of the balance mechanism, via the *Total effective radiation* quantity, which directly (positively) influences the planetary temperatures. The total effective radiation equals the sum of the solar radiation

Quantity	Quantity space
Amount	{min, zero, plus}
Temperature	{low, medium, high}
Solar radiation	{min, zero, plus}
Effective radiation	{min, zero, plus}
Total effective radiation	{min, zero, plus}
Return radiation	{min, zero, plus}
OLR	{min, zero, plus}
Greenhouse gases	{min, zero, plus}

Table 3.3: Water vapor quantities and relevant quantity spaces

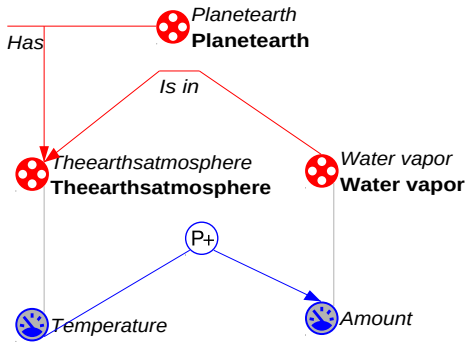


Figure 3.17: Water vapor: Water vapor model fragment

and return radiation emitted by water vapor. Therefore, if there is no water vapor in the atmosphere (i.e. the water vapor's *Amount* magnitude equals *zero*), the OLR compensates for the incoming solar radiation, and the system is at balance. However, the higher the amount of water vapor in the atmosphere, the greater the amount of return radiation, due to the equality and positive proportionality relationships; the quantity space correspondence relationship makes sure the magnitudes are always in the same interval. If the return or solar radiation increase, so will the total effective radiation influencing the temperature (due to the two positive proportionalities provided by their respective quantities and aimed at *Total effective radiation*).

3.5.5 Scenarios

As the outgoing longwave radiation is always there to compensate for its incoming solar counterpart, we don't have two versions of the solar radiation scenario. We did leave the *Increasing solar radiation* scenario in order to demonstrate the planetary radiation balance mechanism. The *Increasing greenhouse gases* scenario is similar to the previously described ones, and can be seen in Figure 3.20

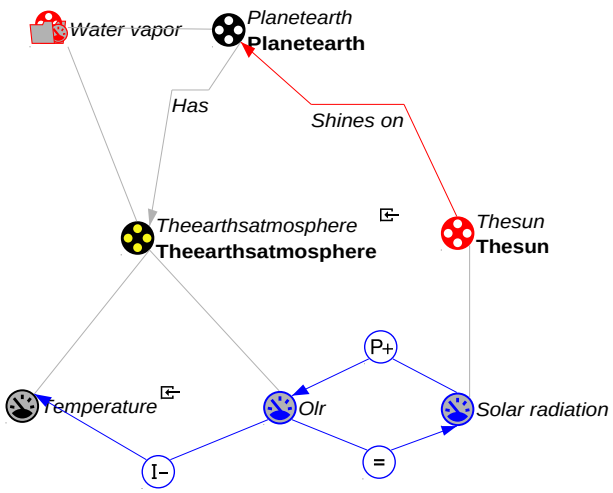


Figure 3.18: Water vapor: Outgoing Longwave Radiation model fragment

3.5.6 Simulation results

In the Increasing solar radiation scenario, the solar radiation quantity's initial value is set to *zero* and its behavior to increase. The behavior of other quantities in the resulting two-state simulation graph (Figure 3.21) shows the Earth's radiative energy balance mechanism at work. We see only the solar radiation and the outgoing longwave radiation increasing. The temperature remains stable, as expected.

The Increasing greenhouse gases scenario, on the other hand, produces a 5-state graph. The rising amount of greenhouse gases in the air increases the temperature, which is the cause of higher levels of water vapor present in the atmosphere. As soon as the water vapor amount enters the positive interval (Figure 3.22, state 3), the return radiation from this greenhouse gas ads to the total effective radiation, which provides the second direct influence on the global temperatures (Figure 3.23), and increases it further.

3.6 Warming aerosols

Although it is nowadays believed that the cooling effect of the aerosols in the atmosphere is dominant, the warming influence also leaves a significant mark on the global climate. Black carbon particles or *soot*, the coproduct of fossil fuel and vegetation burning, absorb and emit back the outgoing longwave radiation in the same way water vapor does (Staudt et al., 2009). Therefore, many of the aspects of the corresponding model will be based on the water vapor model. As a matter of fact, the main differences are in a way reminiscent of those we saw earlier between the two negative feedback domains. To be more specific, as with ice and snow albedo, the amount of water vapor both influences and strongly depends on the temperatures, whereas none of the aerosols have this relation to the planet's temperature.

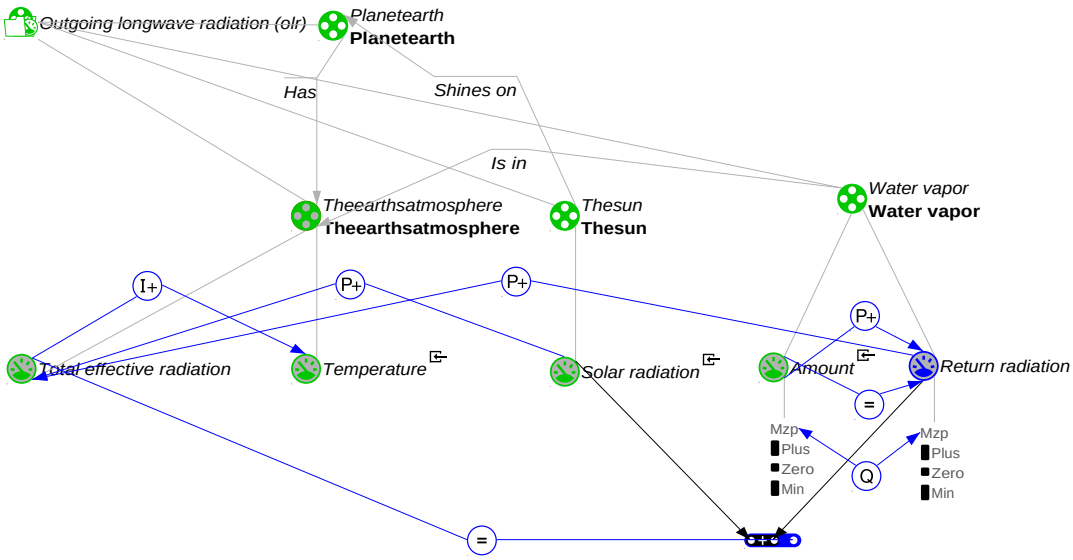


Figure 3.19: Water vapor: OLR absorption model fragment

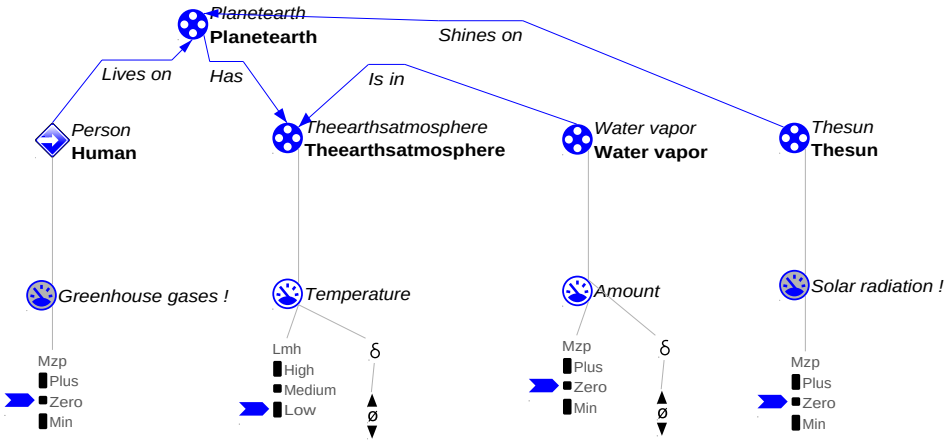


Figure 3.20: Water vapor: Increasing greenhouse gases scenario

3.6.1 Entities and agents

Compared to the cooling aerosol entity hierarchy (or any other entity hierarchy seen so far; Figure 3.1), the only new entity is *Warming aerosols*, which replaces the cooling aerosol / snow and ice cover / water vapor one in the bottom branch of the graph. The agent hierarchy remains the same (Figure 3.2).

3.6.2 Assumptions

As in the case of the previous models, this model doesn't contain any explicit assumptions.

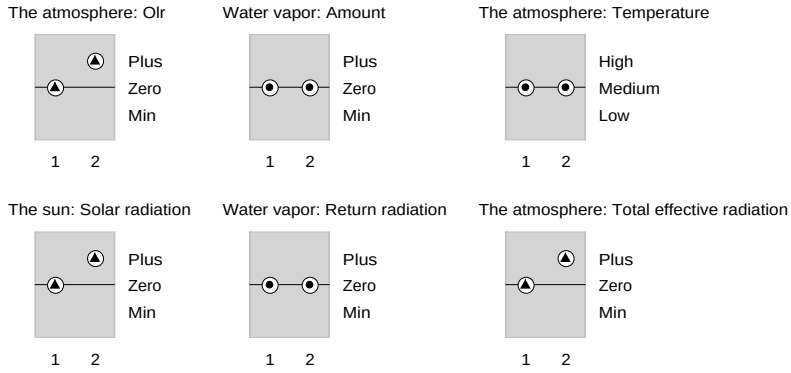


Figure 3.21: Water vapor: Increasing solar radiation scenario value history

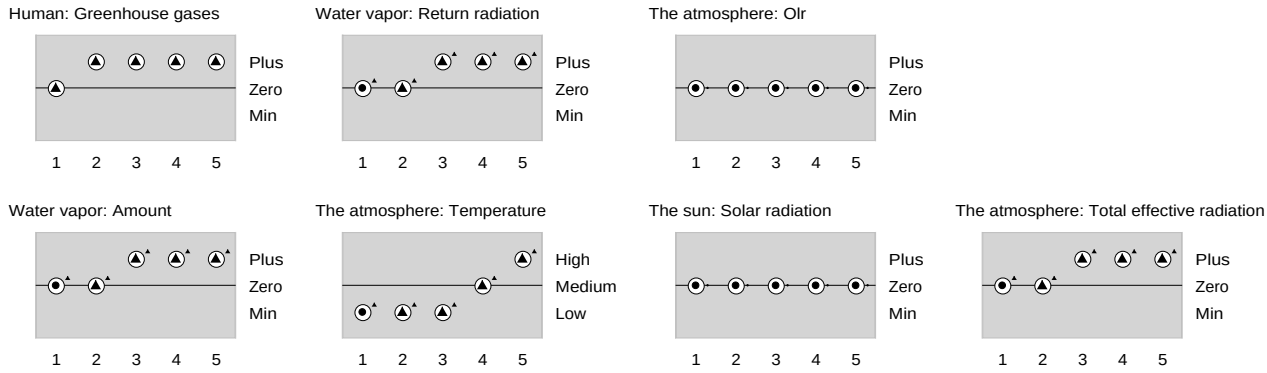


Figure 3.22: Water vapor: Increasing greenhouse gases scenario value history

3.6.3 Quantities and quantity spaces

As mentioned above, the warming aerosol amount quantity has no relation to the temperature. Therefore, we replace the greenhouse gases in the water vapor quantity space table (3.3) with *Human activity* (Table 3.4), a quantity that will correspond to various human activities directly leading to increases and decreases in aerosol amount in the atmosphere.

3.6.4 Model fragments

The positive feedback mechanism shown in Figure 3.19 remains the same. The only notable difference is the right hand side of the total effective radiation equation within the OLR absorption model fragment, where the water vapor entity has been replaced by warming aerosols (Figure 3.24).

Quantity	**Quantity space**
Human activity	{min, zero, plus}

Table 3.4: Warming aerosols: Human activity quantity and relevant quantity space

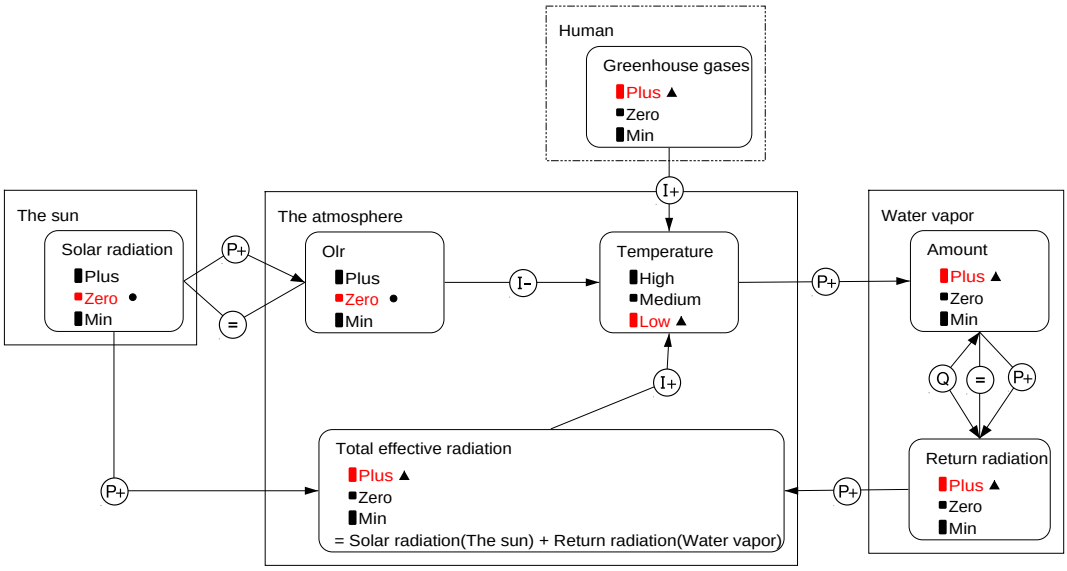


Figure 3.23: Water vapor: Increasing greenhouse gases scenario dependencies for state 3

As described above, the external influence, i.e. the Human agent, provides a positive direct influence on the amount of aerosols, as depicted in Figure 3.25. Moreover, this is the only influence governing the aerosol amount in the atmosphere.

3.6.5 Scenarios

We keep the same number of scenarios and the basic ideas we've seen in the first positive feedback example (Section 3.5), only this time the external influence is different (we use human activity instead of greenhouse gases).

3.6.6 Simulation results

The increasing exogenous behavior set for the human agent activity in the *Increasing human activity* scenario, produces a similar behavior as seen in the previous positive feedback model (Section 3.5.6). However, as with the cooling aerosols (Figure 3.16), we have only one positive influence affecting the amount of our greenhouse gas in the atmosphere, which results in a slightly slower temperature increase (Figure 3.26, states 2 and 3).

The dependency graph (Figure 3.27) shows the above mentioned influence relationships for state 3 of the Increasing human activity scenario. The warming aerosol *Amount* is in the positive interval, whereas *Solar radiation* has the value *zero*, hence, making *Total effective radiation* also positive. As the negative influence on *Temperature* has no effect due to the *OLR* also being zero, the positive one makes the global temperatures increase.

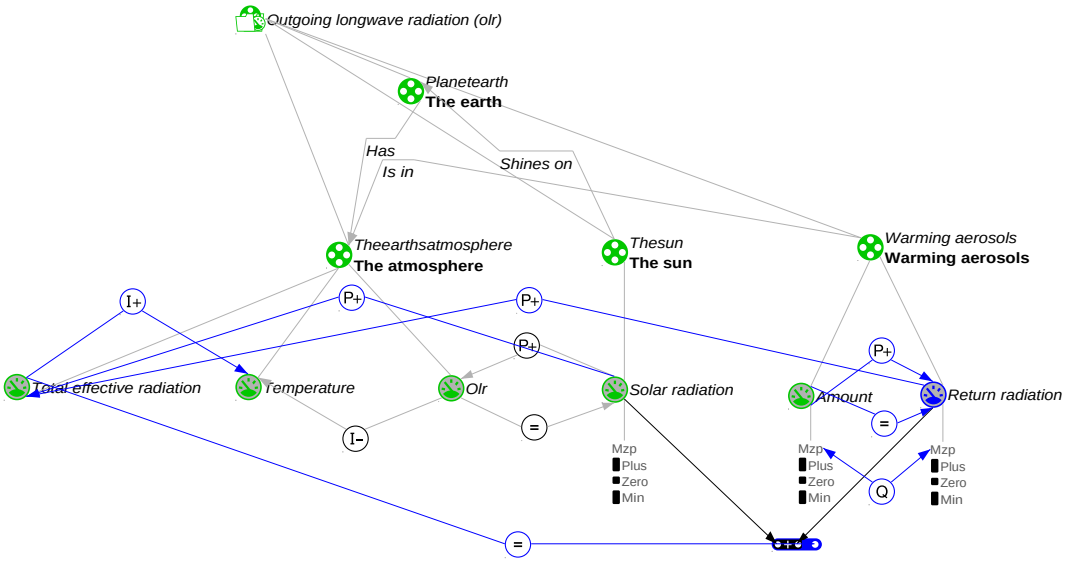


Figure 3.24: Warming aerosols: OLR absorption model fragment

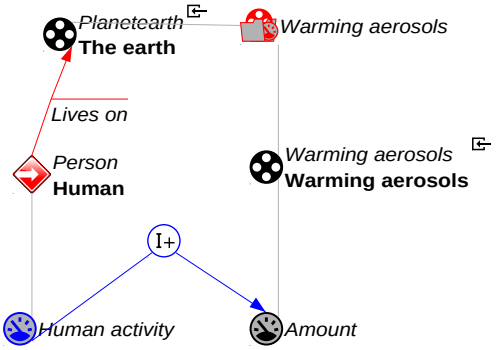


Figure 3.25: Warming aerosols: Human influence agent model fragment

3.7 Low and high clouds

Identified by the United Nations IPCC as one of the most uncertain processes in climate models (Philander, 2008), cloud feedback represents a complex domain that, unlike the factors we've seen so far, affects the global climate in more than one way.

High clouds are optically thin and cold, and reflect a relatively small amount of incoming solar radiation, but capture and emit the outgoing longwave radiation in the same way the two positive feedback factors we've seen so far. Therefore, their net effect on the planet is warming.

Low clouds, on the other hand, being thicker and warmer, work in the opposite way, i.e. they reflect the incoming shortwave radiation back into the outer space, therefore cooling the planet. An important fact to note is that, as the temperatures rise, so do the water particles in the air. Therefore, higher temperatures would lead to more clouds in the upper layers of the atmosphere, which would in turn lead to even further increases in the temperatures, and so on.

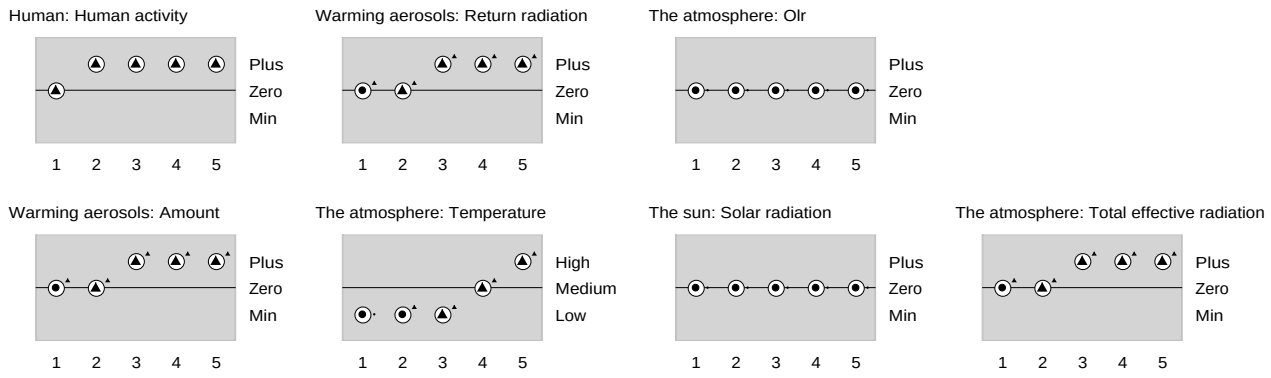


Figure 3.26: Warming aerosols: Increasing human activity value history

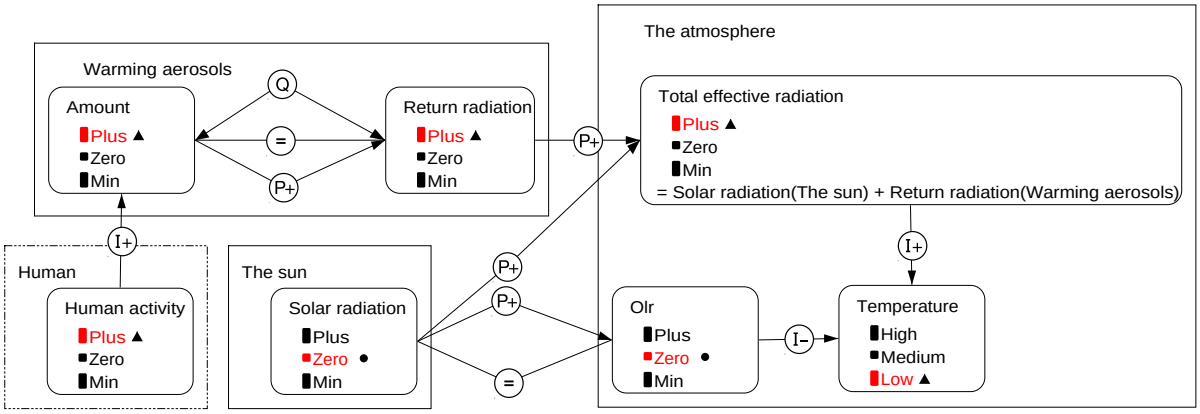


Figure 3.27: Warming aerosols: Increasing human activity scenario dependencies for state 3

Hence, the nature of the domain gives us a unique opportunity to see both of the above presented (competing) mechanisms at work in a single, larger model.

3.7.1 Entities and agents

The entity hierarchy shown in Figure 3.28 has a similar structure as the hierarchies of the models seen so far. The new branch is the one having CloudInSky, denoting the cloud entity we will use for both of our cloud types, as the leaf node.

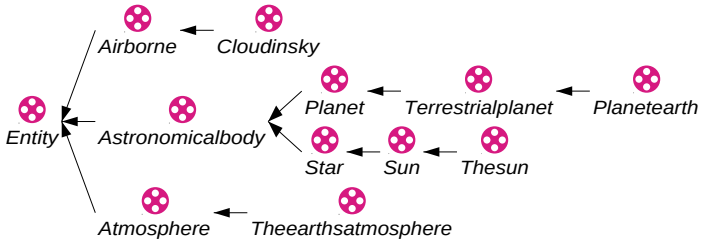


Figure 3.28: Cloud cover entity hierarchy

The agent hierarchy remains unchanged, which means our external influences will be

Quantity	Quantity space
Amount	{low, medium, high}
Temperature	{low, medium, high}
Solar radiation	{low, medium, high}
Effective radiation	{min, zero, plus}
Greenhouse gases	{min, zero, plus}
OLR	{min, zero, plus}
Return radiation	{low, medium, high}
Total effective radiation	{min, zero, plus}

Table 3.5: Cloud cover quantities and relevant quantity spaces

provided by human beings (Figure 3.2).

3.7.2 Assumptions

This time the assumption hierarchy is not empty - it contains three assumptions used to deal with discrepancies between the incoming solar radiation, and low cloud cover amount change rates. The hierarchy given in Figure 5.3 shows *Sr derivative eq lcc derivative*, *Sr derivative gt lcc derivative* and *Sr derivative lt lcc derivative*, where *sr* and *lcc* stand for "solar radiation" and "low cloud cover", and *eq*, *gt* and *lt* for "equals", "greater than" and "less than", respectively. Each of the assumptions will be assigned to a "child" model fragment (meaning it inherits all of the structural details from its "parent") of the *Solar radiation reflectance* MF.

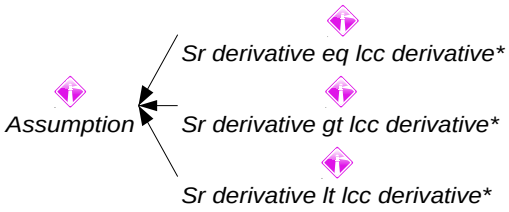


Figure 3.29: Cloud cover assumption hierarchy

3.7.3 Quantities and quantity spaces

The most important difference that needs to be pointed out with respect to the quantities and quantity spaces in this model, in order to avoid confusion, is the fact that *Effective radiation* replaces *Solar radiation* in the positive feedback mechanism, i.e. *OLR absorption* model fragment, as will be shown below. The *Amount* quantity is used to represent the saturation of the sky by both cloud covers. The remaining quantities and quantity spaces are as described earlier.

3.7.4 Model fragments

The model fragment hierarchy is given in Figure 3.30. The hierarchy also shows the three special cases, i.e. subtypes of the *Solar radiation reflectance* model fragment, each carrying an assumption described earlier.

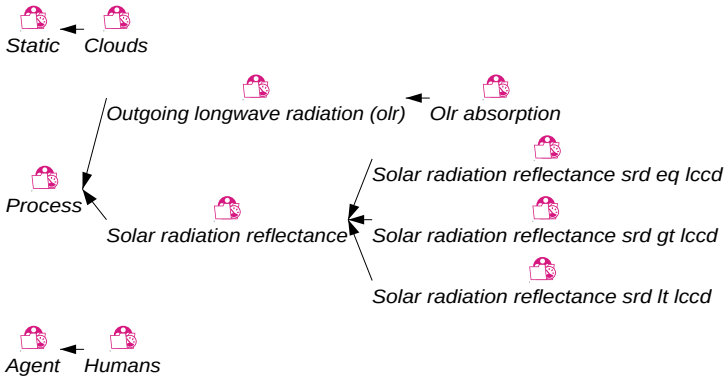


Figure 3.30: Cloud cover model fragments

The single static model fragment, named *Clouds* (Figure 3.31) depicts the idea of the planetary temperature affecting the altitude of water particles in the global cloud cover. That is, as the temperatures rise, the amount of the low cloud cover decreases, as the water particles move to higher altitudes and accumulate in the high clouds, and vice versa.

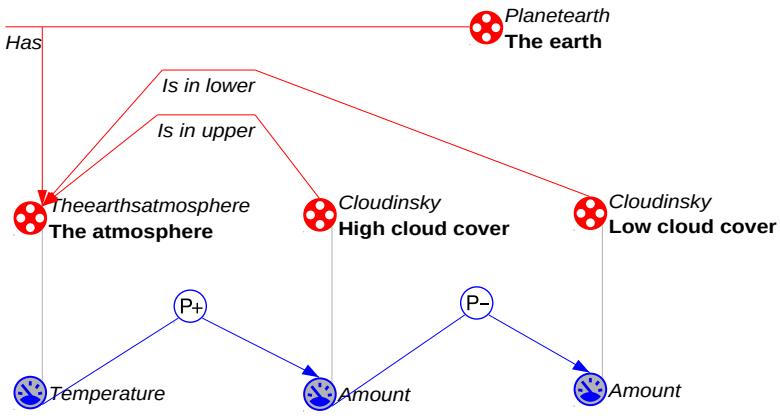


Figure 3.31: Cloud cover: Clouds model fragment

As discussed in Section 3.2.1, the OLR mechanism sends back to the outer space the incoming solar radiation that reaches the surface, i.e. doesn't get reflected by the negative feedback factors. In the two positive feedback models we've seen, we weren't taking into consideration any such factors, and the incoming solar radiation quantity was called simply *Solar radiation*. As the incoming radiation is no longer "pure", i.e. it is filtered by the low clouds before reaching the surface, the *Solar radiation* quantity in the *OLR absorption* is now represented by the *Effective radiation* quantity. The rest of the mechanism is clearly inherited from the two positive feedback models seen earlier (Figure 3.32).

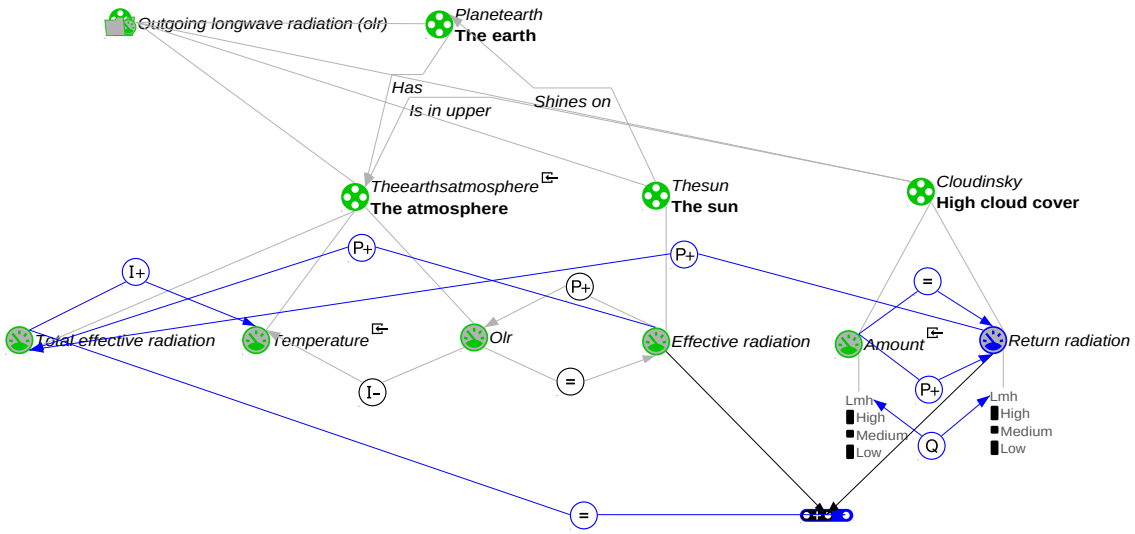


Figure 3.32: Cloud cover: OLR absorption model fragment

Finally, we give an example of the above mentioned mechanism for dealing with derivative change rates in Figure 3.33, where the *Solar radiation* derivative equals the *Return radiation* derivative.

3.7.5 Scenarios

The ideas for initiating the causal chain remain the same as seen earlier. Namely, the scenario ideas revolve around the changes in solar radiation and greenhouse gases, only this time, in order to reduce the number of states, and be able to interpret the results more easily, for the first time, we introduce assumptions in our scenarios. For instance, the *Decreasing solar radiation srd eq lccd* scenario, is shown in Figure 3.34.

3.7.6 Simulation results

In order to illustrate the importance of the assumption idea, we first simulate a decreasing solar radiation scenario without assumtions. Figure 3.35 shows that the discrepancies between the low cloud cover and solar radiation change rates result in the simulation engine producing a total of 17 *initial* states (the full simulation state graph contains 176 states).

We then test the scenario we saw in the above section, i.e. *Decreasing solar radiation srd eq lccd*. Figure 3.36 proves our point, as the simulation graph is now constrained to only 3 initial states, that is, 18 states, overall.

Although we still have a single terminal state, we now have multiple paths leading to it. The reason for this is depicted in Figure 3.37. It appears that the only difference between the three states lies in the magnitude discrepancies of the *Total effective radiation* quantity.

A more detailed look at one of the states (1) reveals how the value of *Total effective radiation* is determined (Figure 3.38). We see that the quantity's magnitude is calculated

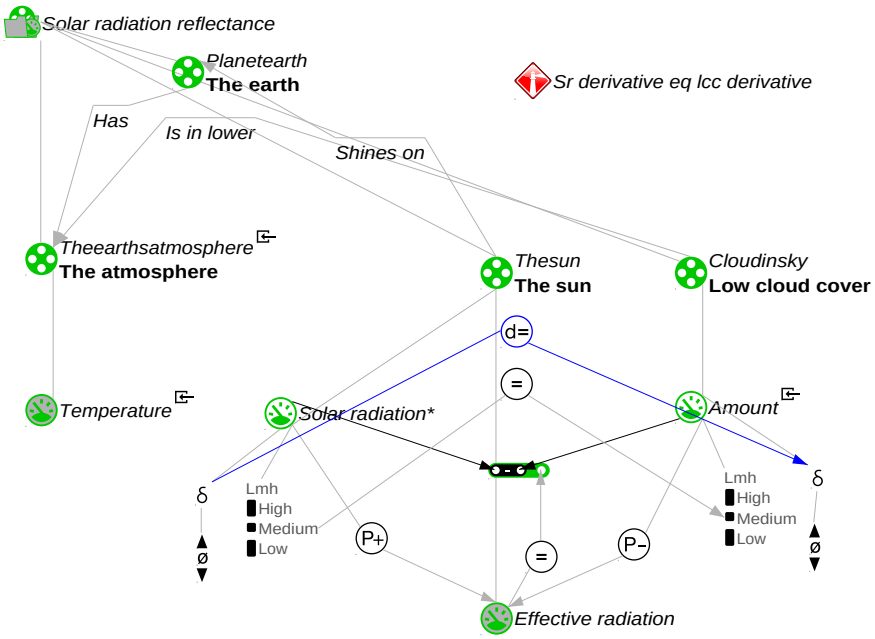


Figure 3.33: Cloud cover: Solar radiation reflectance srd eq lccd

as the sum of *Effective radiation* and *Return radiation*. However, due to the fact that the quantity space of the latter doesn't have point zero defined, or any other relevant inequality information, the reasoner is unable to determine how low interval *Low* actually is, and instead creates a path for each of the possible cases.

Finally, we take a closer look at one of the paths ([1, 5, 10, 20, 23, 18]), or, more specifically, its value history in Figure 3.39. The results are surprising. Instead of seeing the atmosphere cool itself off due to a decrease in incoming solar radiation, the warming effect prevails and the temperatures go up. Looking back at the dependency graph discussed above, we see that, when compared to the positive feedback mechanisms seen earlier, the essential bits haven't changed. However, one clear difference is the choice of quantity spaces. While we know that *min* is less than zero, a value in the *low* interval could be anywhere in the {*min, zero, plus*} quantity space. This seems to be why the amount of high clouds ends up increasing, eventually amplifying the warming effect and overpowering the negative one. A more appropriate choice of quantity spaces should solve this problem.

3.8 Evaluation by experts

Once completed, the models were submitted to domain experts for evaluation. Additionally, the experts were given a summary of the current chapter and an introduction text for clarification.

The experts were asked to answer a number of questions in writing, and add any comments they might have. They also provided the author with additional feedback during a live interview. The summary of their evaluation is given below. Questions 1-5 are model specific, whereas the remaining questions are more general.

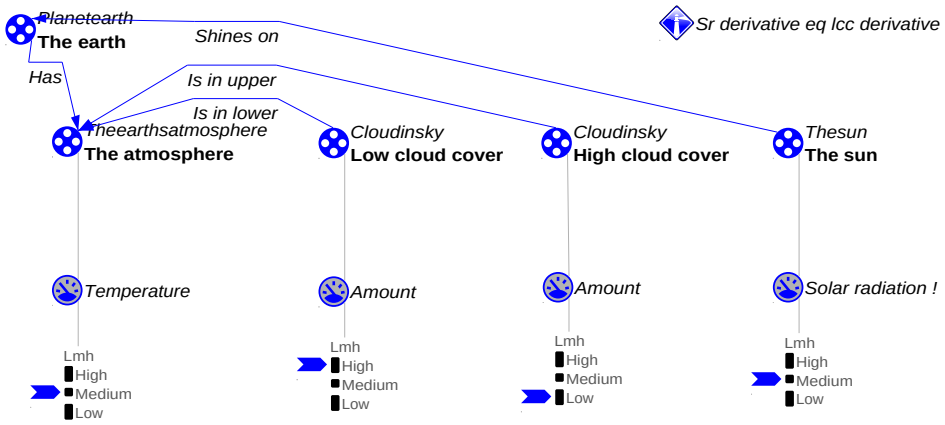


Figure 3.34: Cloud cover: Decreasing solar radiation srd eq lccd scenario

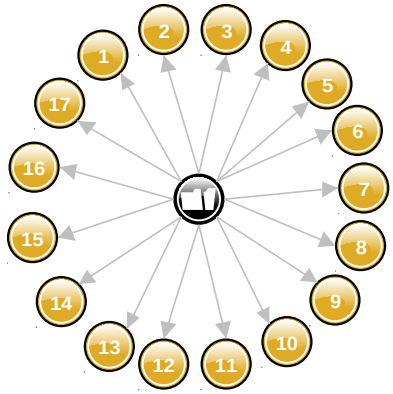


Figure 3.35: Clouds cover: Decreasing solar radiation scenario; initial simulation states

It is worth noting that, at the time of reviewing the models, the cloud model couldn't be assessed thoroughly due to a bug in the modeling environment which was causing unexpected behavior to occur during simulations.

3.8.1 Evaluation by expert 1

1. *How faithfully do you think the models represent the actual domain*?

The expert thinks the author has made a relatively simple representation of the domain that is, in general, in accordance with what is believed the mechanisms are. He did have comments on the way some of the ideas were modeled (to be explained below), but the kernel of the knowledge represented was found appropriate.

2. *What about the choice of quantities and quantity spaces? Are the quantity spaces detailed enough to show the behavior of the system (for learners/experts)*?

The expert believes the quantities and quantity spaces used in the negative feedback models are appropriate. Moreover, he found the quantities used in the negative feedback examples also a good choice. However, he had comments regarding the negative values in the negative feedback examples, which he believes are inadequate. For example, the amount

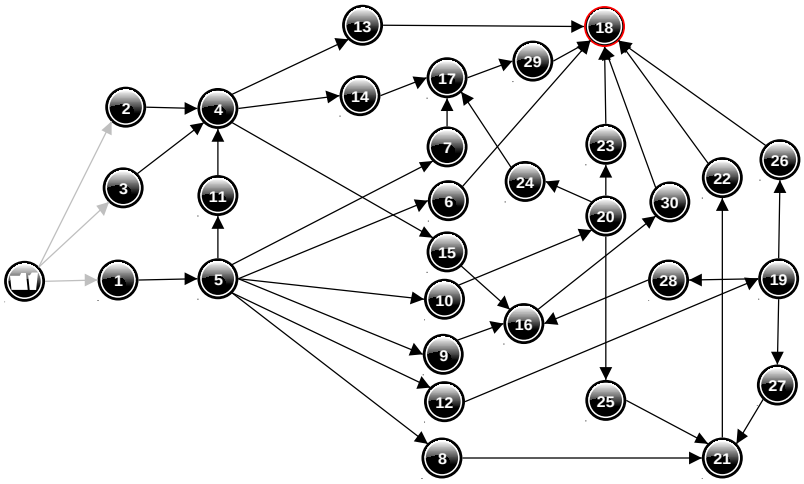


Figure 3.36: Clouds cover: Decreasing solar radiation srd eq lccd state graph

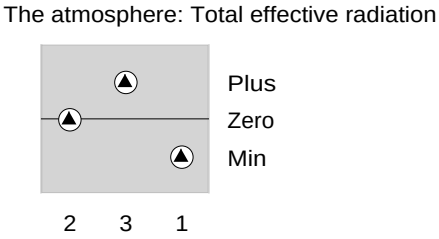


Figure 3.37: Clouds cover: Decreasing solar radiation srd eq lccd inital state value differences

of water vapor, aerosols and clouds could never be negative.

3. *Does the behavior shown in all of the scenario simulations represent the behavior of the actual system*?

If we do not consider the point about the negative values mentioned above, the simulation behavior appears to be representing the actual system faithfully. The cloud model could not be assessed though, due to the Garp3 bug mentioned earlier.

4. *Do you think the model would be suitable for educational purposes, i.e. DynaLearn*?

The experts believes the models would be suitable for use in education, as the causal models are mostly interesting and understandable, but once again, points out the quantity space issue mentioned above.

5. *Do you think the model could be used outside the DynaLearn project for other purposes*?

The answer is yes - in science education of stakeholders, for example. The models, probably, couldn't be widely used by scientists or experts in climate science. However, even for this audience the models can be sort of a starting point in their reasoning.

General questions:

6. *Do you think an abstracted positive feedback mechanism could be used to match against other positive feedback factors in nature*?

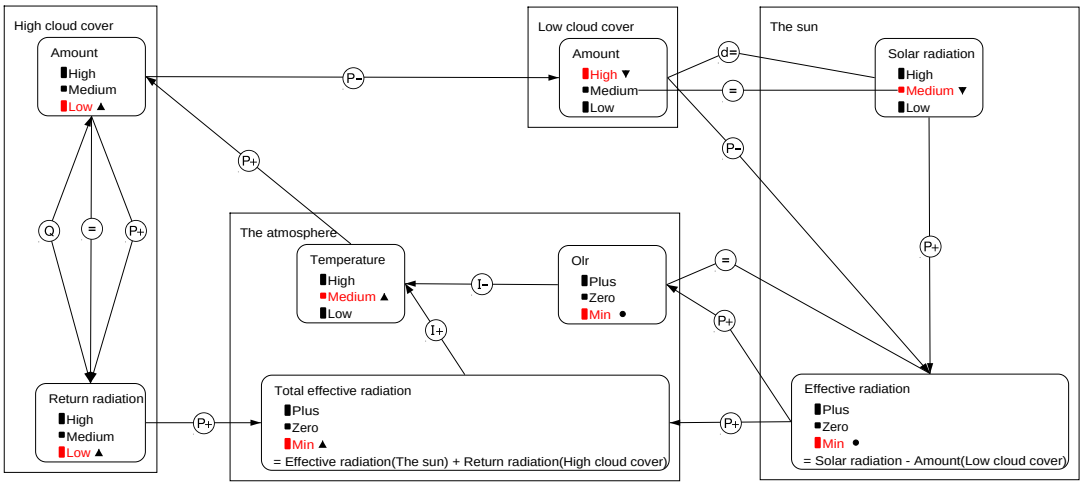


Figure 3.38: Clouds cover: Decreasing solar radiation srd eq lccd dependencies for state 1

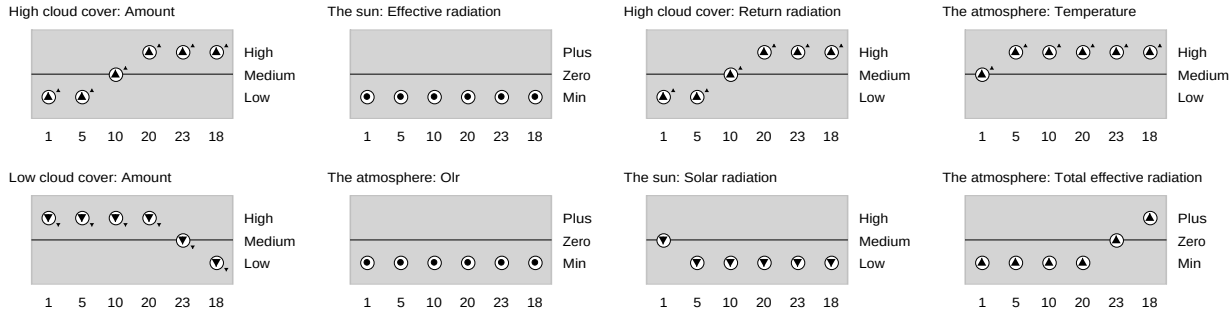


Figure 3.39: Cloud cover: Decreasing solar radiation srd eq lccd scenario value history

The expert agrees that the mechanism could find its use in other environmental science domains. However, due to time constraints, he couldn't provide the author with specific examples.

7. *Do you think an abstracted negative feedback mechanism could be used to match against other negative feedback factors in nature?*

Same as for 6.

8. *Do you have any additional comments?*

First of all, the expert agrees that the task at hand was not an easy one, as the domain in question is quite challenging. Apart from the negative value comment seen above, he suggests additional improvements to be considered. First, he points out that a more specific name for the planetary atmosphere entity could be *troposphere*, but also notes that the name would depend on the end user level, and that leaving it as it is for the beginner learner levels would probably be more appropriate. Additionally, he emphasizes that, as a clear feedback mechanism is not evident in the aerosol models, these factors should probably carry a different name. Finally, he suggest an alternative approach to depicting the outgoing longwave radiation absorption mechanism could be considered.

3.8.2 Evaluation by expert 2

1. *How faithfully do you think the model represents the actual domain*?

The expert's personal attitude toward modeling is different than that of the previous researcher in the sense that he prefers having domain details defined as explicitly as possible. Therefore, he found some of the solutions positive and some negative. For instance, he believes that simply saying that the greenhouse gases directly increase temperature is wrong. Also, subtracting radiation with snow is also inadequate, in his opinion. A possible solution for the greenhouse gases issue, would be to call the agent *greenhouse effect*, he suggests. Finally, he couldn't make any comments about the cloud model, since it is not his area of expertise.

2. *What about the choice of quantities and quantity spaces? Are the quantity spaces detailed enough to show the behavior of the system (for learners/experts)*?

The expert's comments regarding the quantities and quantity spaces were very similar to those made by the previous expert. He found the ones used in the negative feedback models appropriate, but had doubts regarding the negative values in the negative feedback examples. The negative value used for the *Total effective radiation* quantity, though, was found suitable, as having more radiation reaching than leaving the planet is conceptually possible.

3. *Does the behavior shown in all of the scenario simulations represent the behavior of the actual system*?

The expert didn't have any specific remarks regarding the simulations, so the author assumes the behavior matches that of the actual domain.

4. *Do you think the model would be suitable for educational purposes, i.e. DynaLearn*?

The expert is not familiar with the DynaLearn project. However, he's been working on his own set of global warming models for educational purposes, and believes that the author's models are more detailed and could be used for the same purpose (with minor changes).

5. *Do you think the model could be used outside the DynaLearn project for other purposes*?

No comments were made regarding the matter.

General questions:

6. *Do you think an abstracted positive feedback mechanism could be used to match against other positive feedback factors in nature*?

The expert agrees that the mechanism could find its use in other environmental science domains. However, due to time constraints, he couldn't provide the author with specific examples.

7. *Do you think an abstracted negative feedback mechanism could be used to match against other negative feedback factors in nature*?

Same as for 6.

8. *Do you have any additional comments*?

As his colleague above, this expert also believes that, in order for a mechanism to be called a "feedback", there should be a causal loop involved, which is not really present in the aerosol domain. Also, just like the previous expert, he believes that alternative names for the entity currently representing the Earth's atmosphere would depend on the end users (e.g. simply *the air* or *the environment* for absolute beginners; *biosphere* for advanced users). Finally, the expert notes that explicit representations of domains are simply his personal preference, and that there is often a trade off between having working models and models that closely resemble the actual domains.

3.9 Conclusion

According to the expert reviews, it appears the main goal of creating a set of simple models that would faithfully represent the domain of global warming amplifiers has been accomplished. Moreover, the secondary goal of creating a set of reusable mechanisms has also been achieved.

Most of the main deficiencies of the models have also been pointed out by the experts, as seen above. The most important issue so far appears to be the choice of quantity spaces, from the conceptual point of view, but also from the practical one, as seen in the case of the low and high cloud cover model. *Greenhouse effect* is probably a more suitable name for the external agent, than *greenhouse gases*, and the Marian Koshland Science Museum's *feedback factors* should probably carry a different name, too.

It appears that most of the other evaluators' comments revolve around personal preferences regarding the level of explicitness, or the way a mechanism is depicted. In summary, both the compliments and the criticism have been embraced, and will certainly help the models find their place in the DynaLearn project.

Chapter 4

Bayesian Network based Learner Modeling

Modeling students' knowledge is a fundamental part of intelligent tutoring systems (ITS's) and interactive learning environments, and, therefore, an important issue for the DynaLearn project, as well. Appropriate amounts of practice on each skill promote complete and efficient learning (Cen et al., 2007). However, many key aspects of DynaLearn, such as deciding which problems to focus on while tutoring, are reliant upon accurate estimations of the student's knowledge state at any point in time. As one may expect, one of the hardest obstacles to be overcome while designing such a system is the inherent lack of certainty as to how much the student really understands.

As we already know, one of the key concepts in the field of Artificial Intelligence is that of uncertainty. A popular framework that deals with quantification and manipulation of uncertainty is *probability theory*, which gave rise to powerful graphical models for reasoning with uncertain knowledge, called *Bayesian networks* (BNs), which we will be looking at in this chapter, on our way to building a state of the art learner modeling tool.

Bayesian networks have been applied successfully to build student models in several systems in the past, and here we briefly review some of them:

- HYDRIVE (Mislevy and Gitomer, 1995) is a system that models a student's competence at troubleshooting an aircraft hydraulics system. Student's knowledge is characterized in terms of general constructs (dimensional variables), and a belief network is used to update these constructs, using students' actions as evidence.
- Andes (Conati et al., 1997) is an ITS that teaches Newtonian Physics via coached problem solving. This system uses Bayesian networks to do long-term knowledge assessment, plan recognition, and prediction of student's action during problem solving.
- In (Collins et al., 1996), belief networks are applied together with granularity hierarchies. The students' mastery of the learning objectives is determined by using test

items as evidence. Three different structures for the network are compared in terms of the knowledge engineering effort required, test length and test coverage.

- Millán et al. (2000) provide an interesting approach to student modeling by merging the theories behind *adaptive testing* and Bayesian networks. Although our approach doesn't follow this work, it does share some ideas with it, especially those related to the network structure design and *knowledge aggregation*, as we will see later on.

4.1 Probability theory

A simple way to start is to define the probability of an event to be the fraction of times that event occurs out of the total number of trials (Bishop, 2009).

Imagine a simple setup: two boxes, one red, and one blue, each containing apples and oranges. Now, imagine that the red box contains 6 oranges and 2 apples, whereas in the blue one we have 1 orange and 3 apples (Figure 4.1). Finally, suppose that while randomly picking fruit from the boxes, out of 10 trials, we choose the red box 4 times (40%) and the blue one the rest of the times (60%) (we put the fruit back in its box after each trial). Now, to formalize this, we will replace the identities of the boxes and fruits by random variables, B and F, respectively. B can take one of two possible values: r (i.e. red box) or b (i.e. blue box). F, on the other hand, can take either o (corresponding to oranges) or a (corresponding to apples).

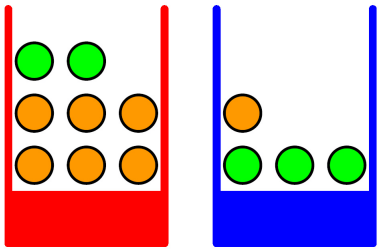


Figure 4.1: Two colored boxes, each containing apples (green) and oranges (orange) (Bishop, 2009)

4.1.1 Conditional and unconditional probability

As we have already mentioned, the probability of picking the red box is 40% or 4/10. Formally, we write this as $p(B = r) = 4/10$. Similarly, for the blue box, we write $p(B = b) = 6/10$[1]. As this probability is supplied to us beforehand and without any additional information, it is known as the *prior* or *unconditional* probability.

[1] Although this notation helps avoid ambiguity, as our example is rather simple, it is safe to write $p(B = b)$ as $p(b)$, and rewrite the notation of the other variables in a similar fashion. One can also use $p(B)$ to denote a distribution over the random variable B.

Suppose that we pick the red box by chance. Then, the probability of picking an orange is simply the fraction of the oranges in the red box, i.e. 6/8, or 3/4. This is called the *posterior* or *conditional* probability and, using the example we've just mentioned, we formalize it as $p(o|r) = 3/4$ (verbalized as "the probability of orange given red box").

4.1.2 Sum rule and product rule

In order to answer questions such as "given that we have chosen an apple, what is the probability that the box we picked it from was a red one?", we first need to understand the two elementary rules of probability, known as the *sum rule* and the *product rule*.

The sum rule says that if A and$B$ are events, the probability of obtaining *either* of them is:

$$p(A \vee B) = p(A) + p(B) - p(A \wedge B) \tag{4.1}$$

If the events A and B are mutually exclusive(that is, if they cannot occur simultaneously), then $p(A \wedge B)$ will be impossible, i.e. equal 0. Thus, the sum rule becomes:

$$p(A \vee B) = p(A) + p(B) \tag{4.2}$$

The product rule also deals with two events, but in these problems the events occur as a result of more than one task. This can be used to explain the only part that has perhaps remained unclear in the first of the two formulae is the so-called *joint probability*, i.e. $p(A \wedge B)$ (also, $p(A, B)$). If the two events are conditionally independent, e.g. rolling two dice, the joint probability is obtained simply by multiplying the probabilities of each event:

$$p(A \wedge B) = p(A)p(B) \tag{4.3}$$

However, if the events are not independent, as in our original example above, where the second trial (i.e. picking a fruit) depends on the first one (i.e. choosing a box) then we need to take the conditional probabilities of the events into account:

$$p(A \wedge B) = p(B|A)p(A) \tag{4.4}$$

4.1.3 Bayes rule

From the product rule, together with the symmetry property $p(A, B) = p(B, A)$ we immediately obtain that:

$$p(B|A)p(A) = p(A|B)p(B) \tag{4.5}$$

This relationship can then be rewritten in the following manner:

$$p(A|B) = \frac{(B|A)p(A)}{p(B)} \tag{4.6}$$

which is what we know as the *Bayes rule*, which today plays a central role in handling many problems in AI that are governed by uncertainty.

4.2 Bayesian networks

Many of the probabilistic inference and learning manipulations in the field of AI, no matter how complex, are based on repeated application of the above explained equations. *Probabilistic graphical models*, diagrammatic representations of probability distributions, are useful in analyzing and solving complicated probabilistic models. These graphical models consist of nodes connected by edges, where each node represents a random variable, whereas each edge denotes a probabilistic relationship between the variables it connects. *Bayesian networks* (BNs), also known as *belief networks*, are a popular class of graphical models, represented in terms of a directed graph, i.e. its links (edges) have a particular directionality indicated by arrows.

Consider an arbitrary joint distribution $p(a, b, c)$ over three variables a, b, and c. By applying the product rule of probability we can rewrite the above distribution in the following form:

$$p(a, b, c) = p(c|a, b)p(a, b). \tag{4.7}$$

It immediately follows that

$$p(a, b, c) = p(c|a, b)p(b|a)p(a). \tag{4.8}$$

In order to represent the right hand side of the above equation in terms of a graphical model, we first introduce a node for each of the variables a, b, and c. Then, for each conditional distribution, we add an arrow from the nodes corresponding to the variables on which the distribution is conditioned. In our example, for $p(c|a, b)$, we will add arrows from a and b to node c, and for $p(b|a)$, a link from b to a, as shown in Figure 4.2.

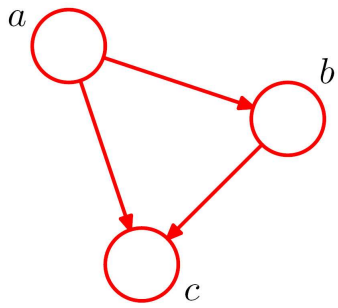


Figure 4.2: A directed graphical model representing Equation 4.8 (Bishop, 2009)

The link directionality tells us how to "read" the graph: if there is an arrow going from node a to node b, then a is the parent of b, and we say that b is the child of a.

We can now state the relationship between a given graph and the corresponding distribution over the nodes/variables in general terms. For a graph with n nodes, the probability of an event is given by the product of the probability of each node (x_i) given its parents, as shown in 4.9. This equation expresses the *factorization* properties of the joint distribution for a directed graphical model.

$$p(x_1, ..., x_n) = \prod_{i=1}^{n} = p(x_i|Parents(x_i)) \tag{4.9}$$

An important restriction that applies to such directed graphs is that they must be kept *acyclic*. In other words, there must be no closed paths within the graph that would allow us to start at one node, follow a path and come back to the point of origin. Therefore, Bayesian networks are essentially *directed acyclic graphs* (DAGs).

Now, assume the following setup: two events could be the cause of wet grass (W) in your garden - either the water sprinker (S) is on or it is raining (R). Moreover, if the weather is cloudy (C), you know it is less likely that the sprinkler is on and more likely that it's raining. These relationships, along with their corresponding probabilities are given in Figure 4.3.

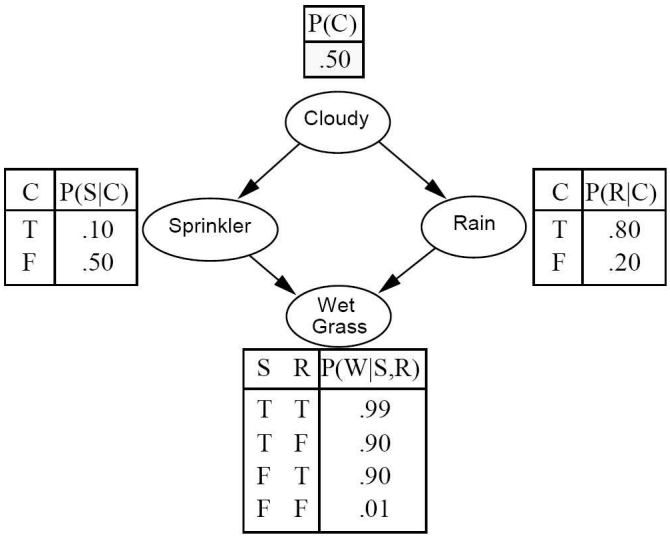


Figure 4.3: A sample Bayesian network (Bishop, 2009)

Since the *Cloudy* node has no parents, its probability table specifies the prior probability that it is cloudy (50%). Moreover, we see that if both events are *True*, the probability of $W = T$ is 99%, and with only one event active, it is still very likely that the grass is wet (90%).

A node is independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes. This relationship is also known as *conditional independence*.

Using Equation 4.9, we can state the joint probability of all the nodes in the graph as:

$$p(C, S, R, W) = p(C)p(S|C)p(R|C, S)p(W|C, S, R) \tag{4.10}$$

The conditional independence relationships allow us to rewrite the above equation as:

$$p(C, S, R, W) = p(C)p(S|C)p(R|C)p(W|S, R) \tag{4.11}$$

Now, assume that in our water sprinkler network we observe the fact that the grass is wet ($W = T$), but we don't know the cause. We can use the Bayes rule to find the "culprit". Suppose we want to find out whether the sprinkler was on:

$$p(S = T|W = T) = \frac{p(S = T, W = T)}{p(W = T)} = \frac{\sum_{C,R} p(C, R, S = T, W = T)}{p(W = T)} = 0.430 \tag{4.12}$$

Therefore, the probability of the sprinkler being the cause of wet grass is 43%. The application of the same rule to the *Rain* node, tells us that Rain is the more likely reason, as $p(R = T|W = T) = 0.708$, or approximately 71%.

4.2.1 Dynamic Bayesian networks

Dynamic Bayesian networks (DBNs) differ to the "static" approach in respect to the dimension of time as an additional ingredient (the term "dynamic" means we are modeling a dynamic system, and not that the graph structure changes over time; Murphy (2002)). The structure can be partitioned into slices corresponding to different points in time. Being able to represent time series or sequences of symbols, DBNs are often used in speech recognition, for modeling protein sequences etc. The *Hidden Markov Model*(HMM) (Figure 4.4) is one of the simplest dynamic Bayesian network examples.

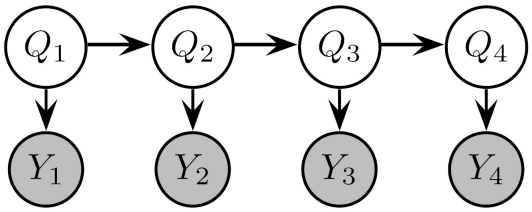


Figure 4.4: A simple DBN (HMM) unrolled for 4 time slices (Murphy, 2002)

The joint distribution for a sequence of length T can be obtained by "unrolling" the network to obtain T slices, and then multiplying together all of the conditional probability distributions (CPDs). For example, the joint distribution for the network depicted in Figure 4.4 can be obtained by:

$$P(Q_{1:T}; Y_{1:T}) = P(Q_1)P(Y_1|Q_1)\prod_{t=2}^{T} P(Q_t|Q_{t-1})P(Y_t|Q_t) \tag{4.13}$$

4.3 Bayesian networks and learner models

In recent years, Corbett and Anderson's Bayesian *Knowledge tracing* model (1995) has been widely popular among the researchers applying Artifical Intelligence methods to education. It is worth noting that, although this approach uses Bayesian inference for tracking the students' knowledge, it doesn't explicitly discuss the use of Bayesian networks *per se*. However, it lays down a strong foundation for future research that does so. The description of the theory underlying the design of a Bayesian network based learner model is given by Reye (2004). Beck and Chang (2007) evaluate this method and point out the first signs of weakness, while Baker, Corbett and Aleven (2008) suggest further improvements, and explore the topic of estimating the guess and slip probabilities of a student's answer. Although we might refer to other researches in the upcoming sections, in general, this will be the path we will follow while explaining the essentials of our approach, building on the work of Brielmann (2009).

4.3.1 Knowledge tracing

As mentioned above, Knowledge tracing was an early approach to student modeling that exploited the power of Bayesian inference, initially used by Corbett and Anderson in their ACT Programming Tutor (APT; 1995), but was also proven to be useful for designing tutors for mathematics (Koedinger, 2002), and improving reading skills (Beck and Chang, 2007), and is statistically equivalent to Reye's two-node dynamic Bayesian network used in many other learning environments (Reye, 2004).

The APT was used in high school and university introductory level programming courses to help students learn how to write short programs in Lisp, Prolog or Pascal. The student would be presented with a short piece of text, followed by a number of exercises being given to the student until the system believes the learner has mastered all the skills introduced in the section. The students wouldn't write the code completely on their own, though. Instead, they would get to choose pieces of code from a list of templates, and then fill in the identifiers and constants.

The system relied on the *ideal student model* – a set of several hundred language specific rules for writing programs that form a "complete, executable model of procedural knowledge of the domain". Every step in the editor made by the student would be compared to this ideal student model, and immediate feedback would be given. If the student's action matched an applicable rule in the ideal model, the internal representation and the editor window would be updated. If not, the student would be provided with a hint, so they would always remain on a "recognizable solution path". A *skill meter* served as an indicator of the student's knowledge state, i.e. the probability that the student has acquired the skills necessary to master a section.

In order to support mastery learning, knowledge tracing was introduced. Corbett and Anderson proposed a two-state model: each rule is either *learned* or *unlearned* (i.e. *known* or *unknown*). The model doesn't implement the idea of *forgetting*, but it does take the possibility of *guesses* and *slips* into consideration. Their knowledge tracing approach uses

four initial parameters:

1. G – the *guess* parameter, i.e. the probability of the student's correct answer being a "lucky guess" (when they don't know the actual correct answer);
2. S – the *slip* parameter, i.e. the probability of the student accidentally giving an incorrect answer (when they actually know the correct one);
3. L_0 – the probability of knowing a skill at the beginning of a tutoring session;
4. T – the probability of learning an "unknown" skill, at any given point during a tutoring session.

Therefore, the probability of knowing a certain language specific rule after an action n is calculated using that action as evidence for the posterior probability (Equation 4.14). In the APT, the student is given exercises until their mastery level reached the probability of 95%.

$$p(L_n|evidence) = p(L_{n-1}|evidence) + ((1 - p(L_{n-1}|evidence))p(T)) \tag{4.14}$$

4.3.2 Student modeling based on belief networks

Reye (2004) suggests that, although it is possible to design a system in which the student model is merely a collection of isolated, independent beliefs, such a model wouldn't suit many domains. He takes the SQL database language as an example, where it is extremely unlikely that a student would be familiar with the "having" clause while being unfamiliar with the "group by" clause.

Hence, he emphasizes that no other tool would be more appropriate to model relationships between beliefs, than belief networks, as their name implies. Reye puts special emphasis on *prerequisite relationships*, as pointed out by Collins and Stevens (1982):

> "Rather we assume only a partial ordering on the elements in the teacher's theory of the domain. ... The teacher's assumption is that students learn the elements in approximately this same order. Therefore, it is possible to gauge what the student will know or not know based on a few correct and incorrect responses. These responses are used to determine a criterion point in the partial ordering; above this point, the student is likely to know any element and below it, the student is unlikely to know any element."

According to Reye, in order to model such a partial ordering, where knowledge of topic A is a prerequisite for knowledge of topic B, there are two aspects we need to take into consideration:

1. The *lack of a student's knowledge* of A implies lack of knowledge of B. This obvious constraint is simply expressed in terms of predicate logic as:

$$\neg\text{student-knows(A)} \Rightarrow \neg\text{student-knows(B)}$$

which is logically equivalent to:

$$\text{student-knows(B)} \Rightarrow \text{student-knows(A)}$$

2. The *evidence of a student's knowledge* of A can be taken as evidence for revising our belief that the student also has knowledge of B. For example, in SQL, knowing the "group by" clause is a prerequisite for knowing the "having" clause, and if we encounter a student who already understands the "group by" clause, then it is more likely that they understand the "having" clause. In other words, evidence for knowledge of A increases our belief that the student also has knowledge of B. However, sometimes there is a weaker relationship between A and B, and we may wish to make no such assertions. For instance, in SQL, the "select" statement is a prerequisite for the "view" statement, but understanding the former doesn't carry much evidence for understanding the latter (for students in the process of learning SQL).

Figure 4.5: A simple BN showing a prerequisite relationship (Reye, 2004)

A graph corresponding to simple two node Bayesian network depicting a prerequisite relationship between two topics is shown in Figure 4.5. Reye further explains how these relationships can be modeled using conditional dependencies, depending on how closely the topics in question are related:

- Knowing A is a prerequisite for knowing B:

$$p(\text{student-knows(A)}|\text{student-knows(B)}) = 1$$

- If there is a close relationship between the concepts (if a student understands A, then it is more likely they understand B):

$$p(\text{student-knows(B)}|\text{student-knows(A)}) = 0.95$$

- If there is a weak relationship between the concepts, i.e. the knowledge of A does not affect the knowledge of B, and the prior probability of B is 0.01, then:

$$p(\text{student-knows(B)}|\text{student-knows(A)}) = p(B) = 0.01$$

Furthermore, a topic can have multiple prerequisites, and Bayesian networks support gathering information about the student's state of knowledge. Consider the following analogy between the suggested network structure and a digital circuit built of logical AND gates with inputs for the prerequisite topics and one output: each gate outputs 1 if and only if all the inputs are 1, i.e. all the topics are known. Moreover, a single non-functioning gate/*fault*, will disable all of the following gates, i.e. not knowing a prerequisite topic means not knowing any topics building on that. Millán et al. (2000) propose the same *knowledge aggregation* idea, as shown in Figure 4.6 (the C nodes represent concepts to be learned, Ts are the topics, and A is a subject).

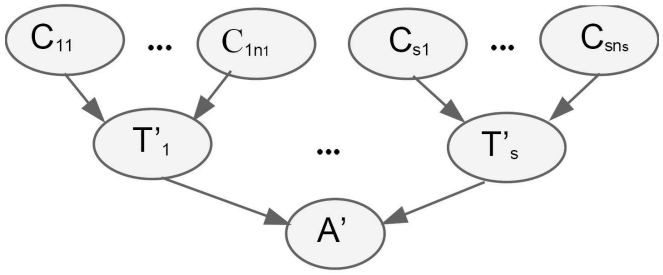


Figure 4.6: Knowledge aggregation in a Bayesian network (Millán et al., 2000)

A student may initially know very little about a domain (i.e. have many faults). De Kleer and Williams's (1987) research behind their General Diagnostic Engine (GDE) system is an efficient approach for finding faults, based on the idea of minimizing the number of measurements (analogously, minimizing the number of questions asked of the student), by making a series of measurements, each of which maximizes the expected amount of information gained by that measurement. Technically, minimizing the expected entropy (H) of the Bayesian network after making that measurement:

$$H = -\Sigma p_i \log p_i \tag{4.15}$$

So, after asking a student whether they know topic t_n, the weighted sum of the two possible responses gives us the expected entropy (H_e):

$$H_e(sk(t_n)) = p(sk(t_n))H(sk(t_n)) + p(\neg sk(t_n))H(\neg sk(t_n)) \tag{4.16}$$

where sk is an abbreviation of "student-knows".

The fact that the number of possible combinations of faults grows exponentially with the number of components is a serious obstacle faced by the GDE procedure. Fortunately, the number of possible combinations for student models is far less. The reason for this lies in the fact that if there is a single faulty node in the learner model, then all subsequent (partially ordered) nodes must also be faulty (at any one point in time). By comparison, in an electronic circuit, subsequent nodes need not be faulty, and so there are more cases to consider.

4.3.2.1 Learners as sources of information

Having described how Bayesian networks can be used to gather information about a learner's knowledge, we now turn to handling the fact that, when it comes to determining their actual state of knowledge, *students are (somewhat) unreliable sources of information.*

As seen in the previous section, we shouldn't ignore the fact that students sometimes make lucky guesses, sometimes make slips and sometimes even give wrong answers deliberately. Therefore, an ITS shouldn't *directly* update its beliefs about the student's knowledge using such information. Consider the prerequisite relationship described above. A single lucky guess could result in the system believing that the student also knew all the prerequisite material. Instead, the we should weigh up the available evidence, including the student's response, to decide how likely it is that the student knows the correct answer.

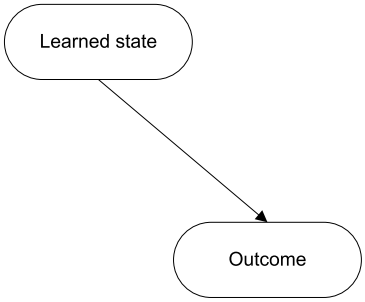


Figure 4.7: Modeling an unreliable source of information using a BN (Reye, 2004)

Let's imagine the simplest possible case, i.e. a domain with only one topic (Figure 4.7). The *learned state* is what the system believes the student knows about the topic. The *outcome* is the evidence acquired after assessing the student's knowledge. Finally, the arrow represents a probabilistic causal relationship (a correct answer is more probable if the student knows the topic). To see the relationship at work represented as a conditional probability, assume the following:

1. $p(outcome = correct|learned) = 0.95$, i.e. the conditional probability of a correct outcome when in the learned state is 0.95 (allowing for the occasional slip);
2. $p(outcome = correct|learned) = 0.20$, i.e. the conditional probability of a correct outcome when in the unlearned state is 0.20 (allowing for lucky guesses);
3. $p(learned) = 0.5$, the prior probability of the learned state being true is 0.5 (we have no initial idea as to whether the student knows the topic or not).

A correct response, would set the probability of the learned state to 0.78. Due to the high possibility of the student's answer being a lucky guess, the value increases, but not too much. An incorrect response, on the other hand, would set the learned state to approximately 0.06. This value is a good indicator that the student doesn't know the topic, but it still leaves a little room for the answer being just a slip.

4.3.2.2 A belief net backbone

As we have seen earlier, the Bayesian network theory puts no constraints on the way beliefs are linked, apart from the prohibition of directed cycles in the structure. However, an actual structure must be specified in any system based on a belief network. For intelligent tutoring systems, Reye (2004) proposes a structure based on three connected ideas:

- a belief net *backbone*, linking all the "student-knows (*topic*)" nodes together in a partial ordering, according to their prerequisite relationships;
- a set of *topic clusters*, each consisting of a single "student-knows (*topic*)" node and a set of additional nodes directly or indirectly influencing the system's belief that the student knows the topic;
- a small set of *global nodes* keeping track of overall student characteristics, e.g. "student-is-bored()" and "student-overall-aptitude()".

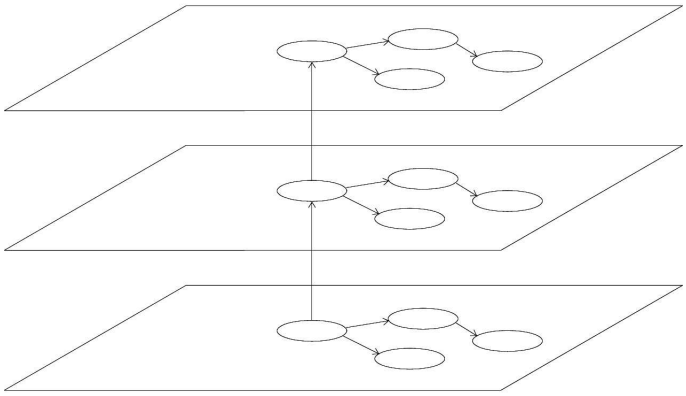


Figure 4.8: A belief net backbone (Reye, 2004)

The proposed structure is depicted in Figure 4.8. The backbone is represented by the vertical arrows, connecting the horizontal planes, and each topic cluster is represented by the nodes in each horizontal plane. The global nodes are not shown for the sake of clarity. Although the figure may seem confusing at first, a closer look reveals that the basic idea is much like the one proposed by Millán et al. (2000) seen earlier (Figure 4.6).

4.3.2.3 Two-phase updating of the learner model

After an interaction with a learner, the ILE should revise its beliefs about the student's state of knowledge. Although it might seem natural to perform this update as a single process (representing the simple transition from prior to posterior probabilities), one should keep in mind that each system - student interaction carries evidence about two important pieces of information, i.e. the probability of the student knowing the topic *before* and *after* the interaction. In other words, it tells us if the interaction has caused any change in the system's beliefs. Therefore, a two-phase updating approach seems more appropriate, where in:

1. *phase 1* we incorporate the evidence acquired from the interaction (if any), about the student's pre-interaction state of knowledge ;

2. *phase 2* we calculate the expected changes (if any) in the learner's state of knowledge after the interaction.

It is obvious that phase 1 is necessary if we want to acquire information about a topic for which there has not been any previous interaction with the learner. However, this phase is also of great importance for gathering information at each interaction, because we must take the possibility of external factors that can affect the learner's knowledge into consideration (forgetting, studying independently etc.).

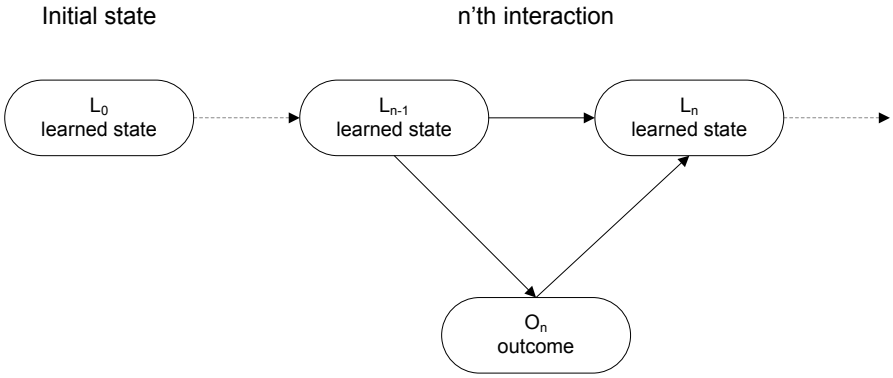


Figure 4.9: Two-phase updating of the student model (Reye, 2004)

The approach is illustrated in Figure 4.9. Here:

- O_n is assumed to be an element in a set of possible outcomes (allowed student responses, e.g. *correct* or *incorrect*);
- $p(L_{n-1})$ is the prior belief that the student already knows the topic, before the n'th interaction (where $n \in \mathbb{N}$);
- $p(O_n|L_{n-1})$ represents the system's belief that the student's answer will be O_n when they already know the domain topic;
- $p(O_n|\neg L_{n-1})$ represents the system's belief that the student's answer will be O_n when they don't know the domain topic.

Now that we have specified the necessary ingredients, given values for each of them, we can use the Bayes rule to revise the system's beliefs in $p(L_{n-1})$ when outcome O_n occurs:

$$p(L_{n-1}|O_n) = \frac{p(O_n|L_{n-1})p(L_{n-1})}{p(O_n|L_{n-1})p(L_{n-1}) + p(O_n|\neg L_{n-1})p(\neg L_{n-1})} \tag{4.17}$$

If we let $\gamma(O_n)$ be the likelihood ratio:

$$\gamma(O_n) = \frac{p(O_n|L_{n-1})}{p(O_n|\neg L_{n-1})} \tag{4.18}$$

we can simplify Equation 4.17 to:

$$p(L_{n-1}|O_n) = \frac{\gamma(O_n)p(L_{n-1})}{1 + [\gamma(O_n) - 1]p(L_{n-1})} \tag{4.19}$$

Equation 4.19 shows why phase 2 cannot be omitted when updating the learner model. When $p(L_{n-1})$ is 0, then the posterior belief $p(L_{n-1}|O_n)$ must also be 0. If there wasn't a phase 2, $p(L_n)$ would be the same as $p(L_{n-1}|O_n)$ and so would be 0 also, in this case. In a similar fashion, if $p(L_{n-1})$ is 1, then $p(L_n)$ would also be 1. In other words, values 0 and 1 represent absorbing states, without phase 2. As a consequence of this, this equation would never allow the system to change its belief, if at some point it became convinced a student did or didn't learn a topic. Moreover, unless $\gamma(O_n)$ is very large, if $p(L_{n-1})$ is very close to 0 or 1, then so will be $p(L_n)$. Therefore, using only phase 1 would make it hard for the system to move away from values close to 0 or 1.

4.3.2.4 Expected changes due to tutoring

In phase 2, in order to model the expected changes in the student's knowledge as a result of the interaction we need a formula for $p(L_n|O_n)$, so, for each possible outcome, O_n, in the set of possible outcomes, we can assign a probability to $p(L_n)$. To fully define the node relationships of the right-hand side of Figure 4.9 ("n'th interaction"), we need two conditional probabilities for each possible outcome:

- $p(L_n|L_{n-1}, O_n)$ – the probability of remaining in the learned state as a result of the outcome (i.e. the rate of remembering/not forgetting);
- $p(L_n|\neg L_{n-1}, O_n)$ – the probability of moving from the unlearned state to the learned state as the result of the outcome (i.e. the rate of learning).

Now we can easily express the revised belief after an interaction as:

$$p(L_n|O_n) = p(L_n|L_{n-1}, O_n)p(L_{n-1}|O_n) + p(L_n|\neg L_{n-1}, O_n)p(\neg L_{n-1}|O_n) \tag{4.20}$$

If, for notational simplicity, we assume that:

- $\rho(O_n) = p(L_n|L_{n-1}, O_n)$, and

- $\lambda(O_n) = p(L_n|\neg L_{n-1}, O_n)$,

we can simplify Equation 4.20 as follows:

$$p(L_n|O_n) = \lambda(O_n) + [\rho(O_n) - \lambda(O_n)]p(L_{n-1}|O_n) \tag{4.21}$$

4.3.3 Bayesian knowledge tracing model issues

Beck (2007) pointed out a deficiency of most methods for developing Bayesian Knowledge Tracing models for specific skills – the *identifiability* problem, where models with equally good statistical fit to performance data may make very different predictions about a learner's knowledge state, which could result in different numbers of problems to be assigned to a student. Consequently, the problem could result in under- or over-practice.

To address this problem, Beck and Chang (2007) tried addressing the problem by constraining the four basic model parameters by finding a prior probability across all skills, using three basic approaches, namely, the *baseline approach*, the *bounded guess and slip* method, and the *Dirichlet priors*. However, their solutions are vulnerable to a different statistical problem, termed *model degeneracy* by Baker et al. (2008), where it is possible to obtain model parameters which lead to paradoxical behavior, such as the probability the student knows a skill dropping after three correct answers in a row.

A model is considered *theoretically degenerate* when its guess or slip parameter is greater than 0.5. A guess parameter over 0.5 signifies that a student who does not know a skill is more likely to get a correct answer than a wrong one. Similarly, a slip parameter over 0.5 means that a student who knows a skill is more likely to get a wrong response.

A model is deemed *empirically degenerate* if it fails one of two tests. If a student's first n responses are correct, but the model's estimated probability that they know the skill is lower than before the n actions, we say the model failed the first test of empirical degeneracy. Also, if a student gets a skill correct m times in a row without reaching skill mastery, we assume the model failed the second empirical degeneracy test (the values of n and m are arbitrary; Baker et al. (2008) consider $n = 3$ and $m = 10$ as reasonable cut-off points for the two tests).

4.3.3.1 Contextual estimation of the guess and slip parameters

Baker et al. (2008) propose a new method for finding two of the four basic parameters, that differs in estimating the guess and slip parameters for individual actions (i.e. related to the context), instead of holding them constant for all situations. First, the authors use history log files from earlier interactions with the student to estimate whether an answer was a guess or slip, using Bayesian analysis. Second, machine learning methods, such as *linear regression*, *support vector machines* and *multilayered perceptron*, are used to identify features of an action that characterize whether that action was a guess or a slip, independent of subsequent actions. Finally, parameter values are fit for $P(T)$ and $P(L_0)$, for each skill, using curve-fitting.

Contextual estimation of the two parameters doesn't eliminate model degeneracy, but decreases it substantially. Furthermore, the method leads to significantly higher accuracy than prior methods, and seems generalizable – the machine-learned model of guess and slip was trained on only 64 skills, but functioned effectively within all 253 skills it was tested on.

Chapter 5

From Garp3 Models to Bayesian Network Based Learner Models

In this chapter, we will see how we can move from a Garp3 model to a learner model that will serve as a basis for interacting with a student. More specifically, as this type of learner-system discussions in DynaLearn are handled by the QUAGS automatic question generator for qualitative simulations (Goddijn et al., 2003), which works on a scenario/simulation basis, we'll be looking at possible ways to represent scenario specific student knowledge in terms of a belief network.

We will first show how DynaLearn quiz question generation works in Section 5.1, which will help us understand the kind of information that's supposed to flow in an out of the network. Then, we will take a look at an existing approach proposed by Brielmann (2009), and discuss the possibilities for improvements (Section 5.2). In Section 5.3, we will see a proposal for a global architecture our learner model should fit in. We will then describe the essentials of the belief network structure for individual scenarios, while paying special attention to knowledge aggregation and some design issues that are specific for the domain of Qualitative Reasoning (Section 5.4). Finally, we will evaluate the model, discuss the results and make our final remarks (Sections 5.5 and 6).

5.1 Question generation

Figure 5.1 shows the architecture of the DynaLearn quiz subcomponent. The information flow starts with the user who constructs and simulates QR models. The information from these models and simulations is used to construct a Bayesian network. Based on the information in the network, i.e. the probabilities of knowing each concept, a question focus is determined, and a question request is sent to QUAGS. The question generator uses this request, the information about the models, question templates and a number of criteria (to be discussed in Section 5.4.1) to output a list of possible questions. A question is selected and forwarded to the learner. After they provide an answer, the information about the question and the response is stored in the dialog history, and the belief network is updated.

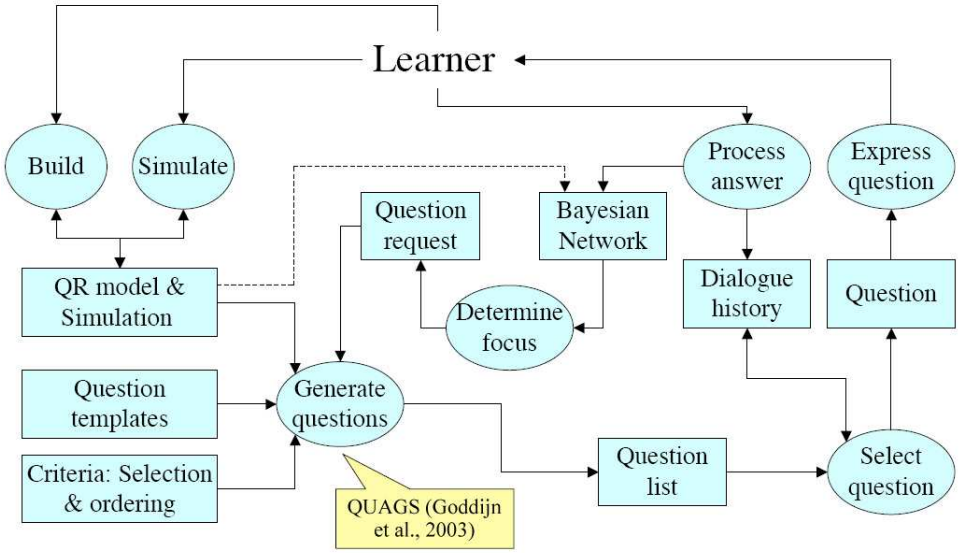


Figure 5.1: Dynalearn quiz – question generation

5.2 A learner model based on a Bayesian network for Garp3

In Brielmann's (2009) learner model, each Garp3 primitive is represented as a node in the model Bayesian network, where magnitudes, derivatives, dependencies and correspondences are set to be the root nodes. We've already discussed the *knowledge aggregation* methods proposed by Reye (2004) and Millán et al. (2000), and Brielmann adopts the same approach in her model. We can observe the low level concepts as the concept nodes, the quantities as topic nodes, and entities as subject nodes found in Millán et al.'s paper. That is, in order for a student to understand an entity, they must know all of its quantities first. In order to know a quantity, on the other hand, a learner must understand all of the root concepts *directly related to it*. The idea is shown in Figure 5.2 for the domain of *Tree and shade*. We see that the proportionality and influence nodes both contribute to two upper level nodes, as each of them carries knowledge about two concepts.

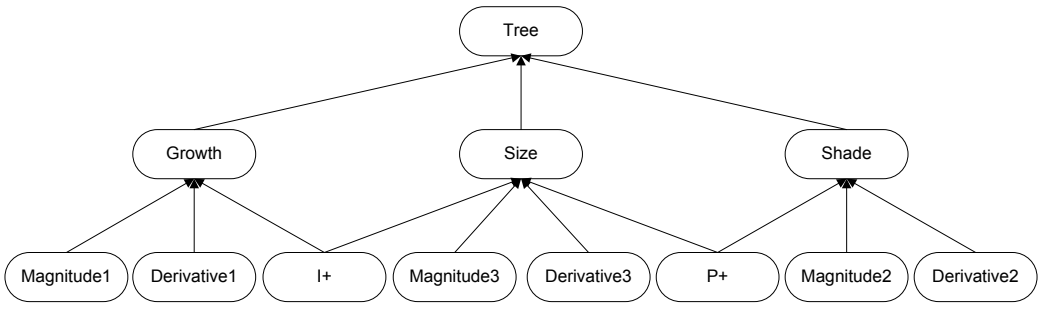


Figure 5.2: Tree and shade: Knowledge aggregation

Each root node is connected to an extra child node representing an observation from a system-learner interaction. This approach makes it possible to include the possibility of

guesses and slips, as proposed by Reye (2004). This is shown in Figure 5.3 for the same domain as in network after a single interaction (Brielmann places her root nodes at the top; in our future discussions, we will still refer to the root nodes as the "low level concepts", as shown in Figure 5.2).

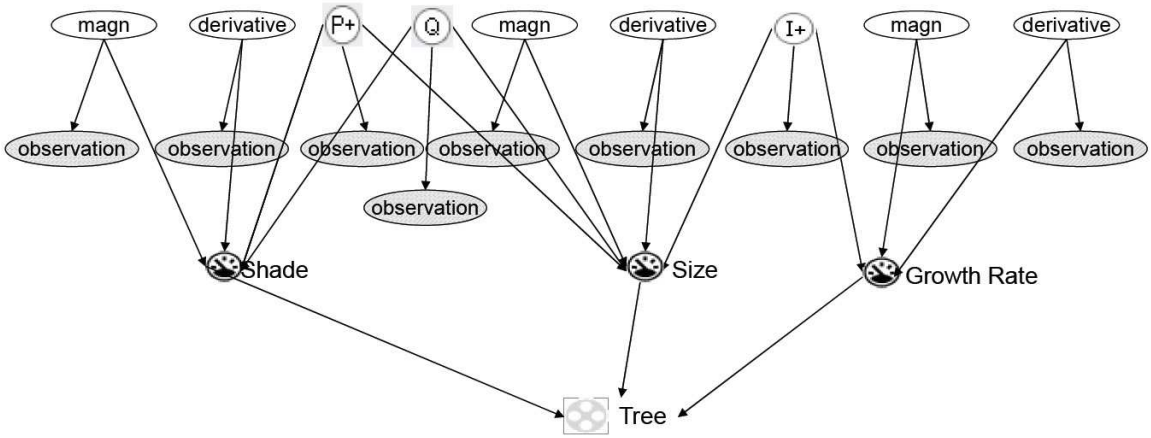


Figure 5.3: A learner model network for a small growing tree, 1 time slice/interaction (Brielmann, 2009)

As the learner's knowledge changes over time, after each interaction, the network is extended by another time slice (Figure 5.4). This lets us collect more than one piece of evidence for a piece of knowledge and provides us with the means to add the possibility of the learner learning or forgetting something between two points in time/interactions, as discussed in Section 4.3.2.

There is a number of key issues not covered by Brielmann's approach. One of them is dealing with mathematical dependencies, i.e. inequalities and calculations. The latter is particularly interesting, as it is more complex than any other relationship in a model. Furthermore, the approach proposes a solution only for a single scenario. If a model has multiple scenarios, the system would see it as a completely different domain. This also means that if a new scenario involves concepts that were already covered by a learner, the system would have no way of knowing this, and the student would have to answer the same questions again. Moreover, it doesn't cover recurring concepts or substructures within the same scenario. Learning about one instance of a model fragment should imply we have also learned something about any other instance of the same MF. We will discuss possible solutions to these, and other issues in sections 5.3 - 5.5.

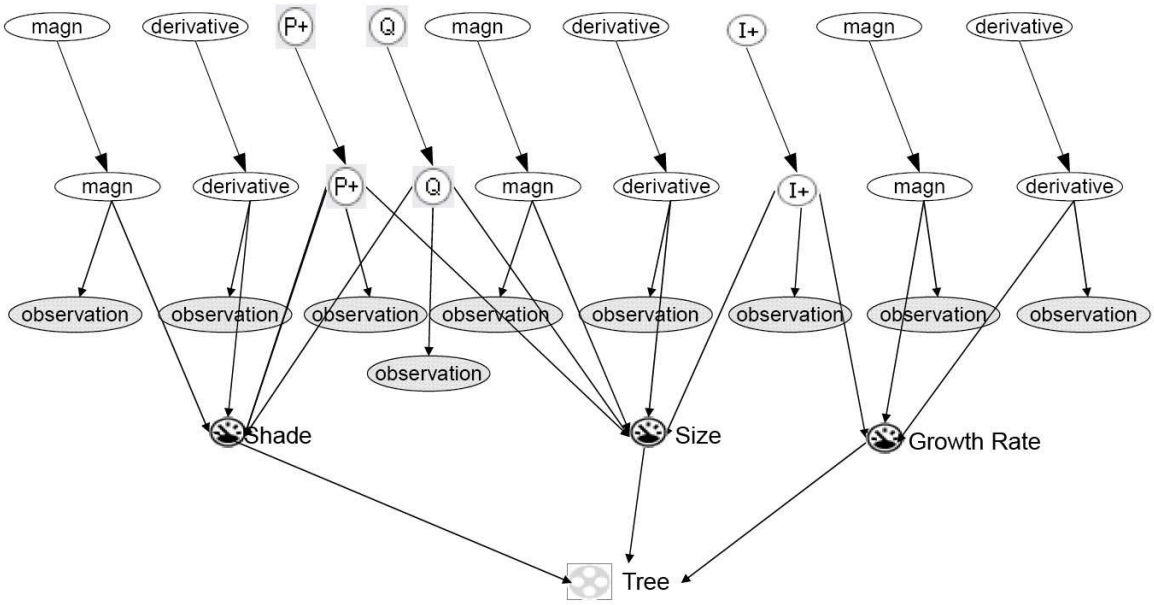


Figure 5.4: A learner model network for a growing tree, 2 time slices/interactions (Brielmann, 2009)

5.3 Global architecture

A single scenario/*superstate file*[1] within a Qualitative Reasoning model holds only partial information about the domain covered by the model. Similarly, a single QR model could be only a small part of a bigger domain a student's supposed to cover (take the global warming amplifier domain and models seen earlier, for example). Therefore, our learner model should not only be able to handle individual scenarios, as per single examination session provided by the question generator, but also support moving between scenarios, or even models.

Just like in the problem description, we'll approach the task bottom-up, i.e. start with scenarios. Thinking about how not to lose the information acquired from one examination session when moving to the next scenario, two baseline possibilities immediately seem appropriate. We could either update the information inside the existing Bayesian network (Figure 5.5), or create a new (global) one, based on the information contained in the previously created network(s)(Figure 5.6).

The advantage of the second approach over the first one is mainly in the fact that the individual scenario based BNs are independent, which makes per case belief network handling much easier. However, this also means that the system would be unaware of the knowledge gained in the previous scenarios. The first approach, on the other hand, would solve this,

[1]The structure of superstate files, i.e. QR model simulations represented in OWL, the Web Ontology Language, as well as the language itself, are given in Appendix A. It is strongly advised to consult the Appendix for the sake of better understanding of the upcoming discussions.

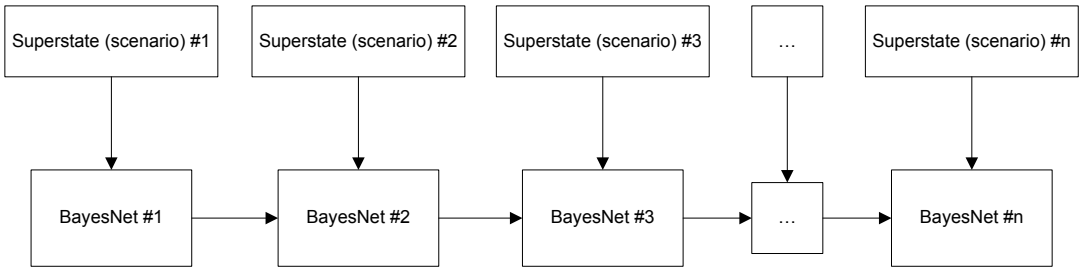


Figure 5.5: Approach 1: The information from one BN is propagated to the next one

but it would also move a lot of excess information from one network to another.

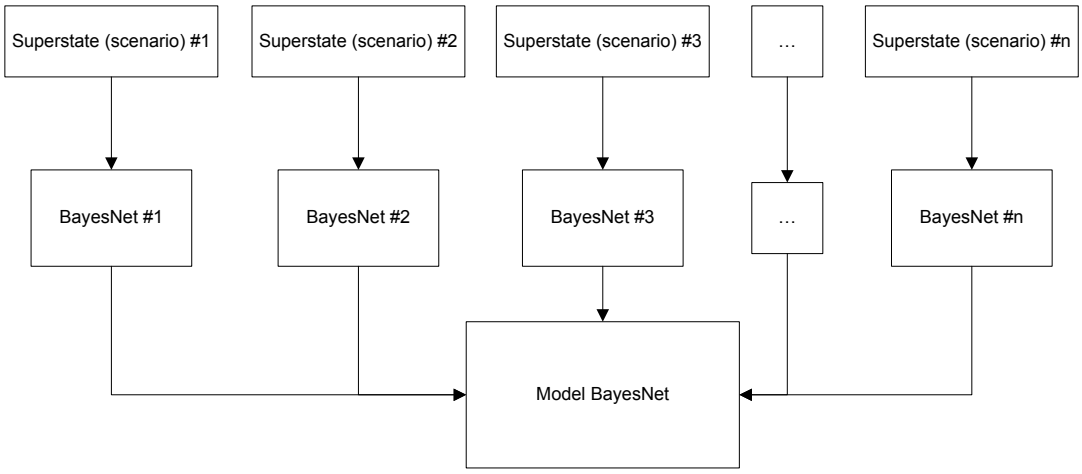


Figure 5.6: Approach 2: The information from each scenario BN is added to a global BN

The advantages/deficiencies of the two approaches bring us to a conclusion that the optimal solution lies in a combination of the two methods. We know we don't want to simply move the knowledge directly from one scenario to another and aggregate unnecessary information as we move on. Also, trapping the acquired information in a single external network is definitely not the way to move forward. Therefore, an ideal approach should store acquired knowledge in an external repository after each examination session, but also be able to retrieve the *necessary* information from that source when switching to a new scenario (Figure 5.7). We're saying "repository" instead of simply "Bayesian network", since one such construct should probably also keep track of the number/type of questions already asked for each concept, or other session specific information.

When loading a new simulation, the system would compare the concepts occurring in it with the ones contained in the model knowledge base. It would then create the belief network for the given scenario and update the information about the concepts the student has already seen earlier accordingly. The same approach could be easily extended to the global/domain level, so that the information would flow between a domain knowledge base (KB) and the individual scenario KBs, as shown in Figure 5.8.

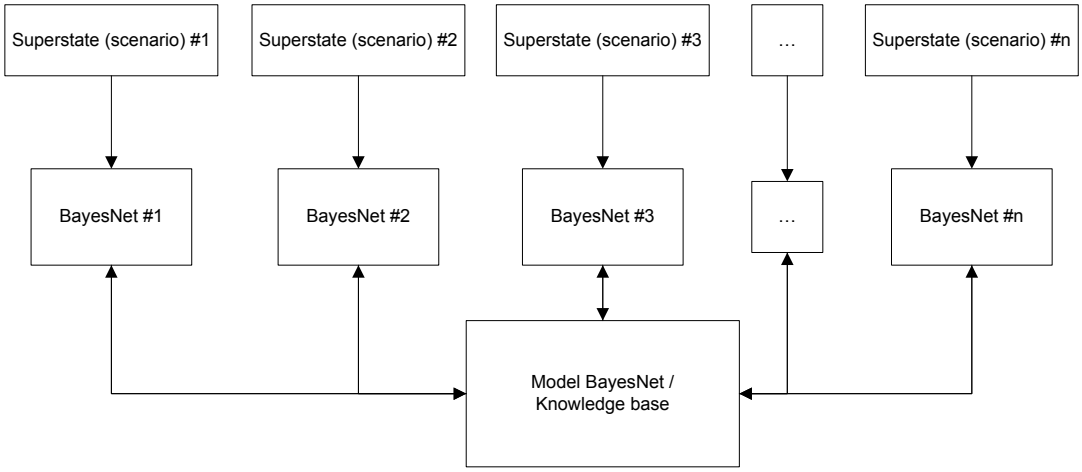


Figure 5.7: Approach 3: The information from each scenario BN is added to and retrieved from the model BN/KB

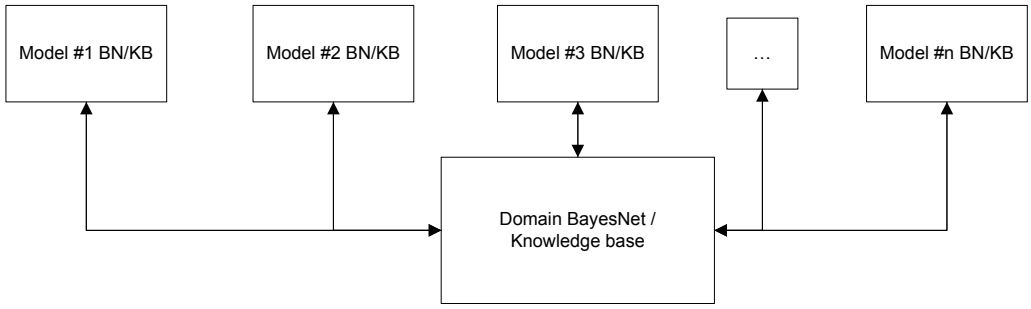


Figure 5.8: Global view: The information from each model BN/KB is added to and retrieved from the domain BN/KB

5.4 Individual scenarios

Before discussing the structure of the scenario specific belief networks, we need to answer the following fundamental question: what needs to be learned? The task becomes more complex once we realize the question is twofold. That is, one needs to keep two perspectives in mind – that of the learner and one of the system. For instance, suppose the student needed to answer the following question from the Tree and shade domain: *How big is the shade of the tree going to be by the end of the simulation?*[2]. The learner could see this as merely a question about the shade of the tree. However, in the "eyes" of the ILE, this is a specific question aimed at the concept of *Magnitude* belonging to the *Shade* quantity of the *Tree* entity. This low level view is what we should pay special attention to, as it holds the essential information stored in each QR model, as suggested by Brielmann (2009). To determine exactly what

[2]This is not necessarily what an actual QUAGS question would look like.

low-level concepts need to be represented in the network structure, we need to see first what questions can be asked, and what information the answers to questions hold.

5.4.1 QUAGS, questions and answers

The question generator for Garp3 can reduce the total number of possible questions about a domain to a reasonable number using multiple methods. The general idea is to first restrict this total by using certain criteria, and then select the most appropriate questions from the restricted set, according to a different set of criteria (Goddijn et al., 2003).

Setting the scope of the question generator could be slightly out of the scope of this research, and will not be discussed in detail. What is important, however, is understanding what the focus of the individual questions can be. A thorough analysis of all QUAGS question types has produced the following list of low level concepts (to be explained below) the answers to such questions can hold information about:

1. Magnitudes
2. Derivatives
3. Quantity spaces
4. Influences and proportionalities
5. Inequalities
6. Calculations
7. Correspondences

Each of the items in the above list can be addressed *directly* by a QUAGS question. We consider quantities, entities and model fragments high level concepts, as the question generator can't ask any direct questions about any of them. Keep in mind, though, that if the current focus of the question generator was a certain quantity, we would like to track the student's knowledge about that quantity. Therefore, we're not saying high level nodes should be left out from the network structure.

5.4.2 Dealing with mathematical dependencies

As mentioned in Section 5.2, Brielmann doesn't discuss the issue of inequalities and equations. The former doesn't seem to be difficult to solve, as it involves only two concepts. This means it can be treated in the same fashion as Brielmann treats influences and proportionalities, i.e. as a single root node connected to the related quantities (Figure 5.2).

Equations, on the other hand, involve three quantities *and* an inequality. One solution would be to add separate root nodes for the inequality and the calculation. However, considering the fact that, due to the basic properties of addition and subtraction, the result of an

equation can become a term in a new equation involving the same concepts (e.g. $A+B=C$, but also $C-B=A$), we should regard QR equations as what they essentially are, i.e. three way relationships. This means that instead of two, we can add a single root node representing the equation and connect it to the three quantities involved, as shown in Figure 5.9.

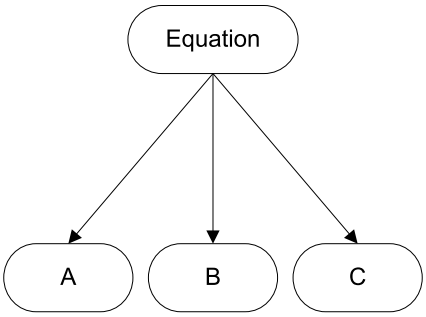


Figure 5.9: Bayesian network substructure for equations

5.4.3 Augmenting the superstate

The superstate file structure can't support the learner model we have in mind, as it lacks information about most of the low-level concepts seen above (to be more specific, everything but influences and proportionalities). Therefore, we start by augmenting the superstate file to include the missing concepts, in the same fashion these concepts are included in the full model ontology, i.e. by adding individuals and appropriate relationships between these individuals.

Moreover, as the naming algorithm wasn't differentiating between model and simulation specific concepts, we add a suffix (*_s*) to the existing *owl_* prefix to be able to tell the difference between the two types. The shortcoming of the previous method was the fact that the first appearance of the *Shade* quantity in a simulation and the model fragment could carry the same name (e.g. *Shade1*). We'll see why this is important immediately below.

5.4.3.1 Recurring concepts

Imagine the following setup – in an additional model fragment of the *Tree and shade* model, instead of one, we have two trees, where one grows in the shade of the other (bigger) one. Therefore, the more the big tree grows, the slower will the small tree grow. Formally, this could be represented by a negative proportionality between the *Shade* quantity of the big tree and the *Growth* quantity of the small one (Figure 5.10). Keep in mind that this third model fragment would become active only if the scenario setup allowed for it. Now, suppose in one scenario, this new model fragment activates. In terms of low level concepts seen above, same forces are at work for both trees, because they are merely instances of the same entity, that serves as a condition for two other model fragments, that specify the dynamics of the tree and shade growth process. Going back to the original question, i.e. "What needs to be learned?", we realize that what we want to learn about is the *Tree* entity, and any instances of this entity should be treated as such. That is, every instance of a concept carries a certain

amount of information about the generic entity, but the global (model/domain) knowledge base should keep track only of the generic concepts, as this is what we want to know about.

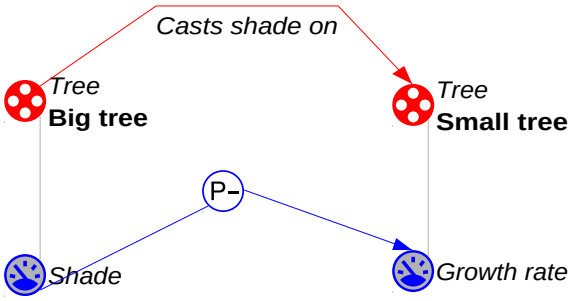


Figure 5.10: Tree and shade: Growing in shade model fragment

It should be clear by now why we want to be able to differentiate between the simulation specific concepts and the full model ones. In other words, any OWL individual carrying a *_s* suffix is simply an instance of some concept a student needs to learn about. Therefore, we strengthen this relationship by including a *hasInstance* property and adding the generic concept relationships to the superstate file, as shown in the following example:

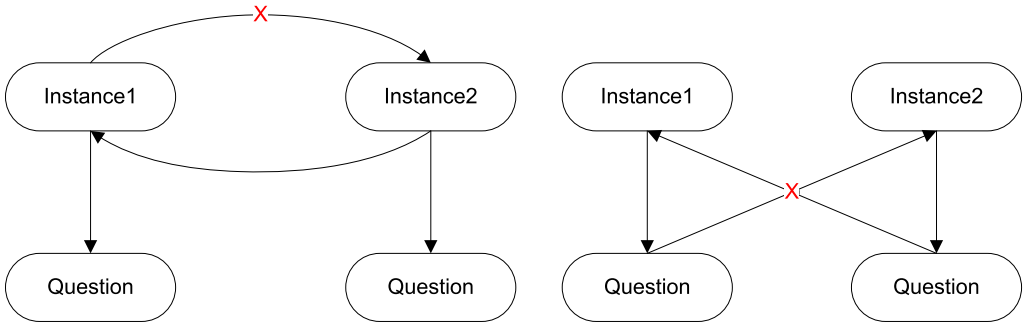
$$\begin{aligned}&\text{owl\_ae\_Tree} \sqsubseteq \text{Entity}\\&\text{owl\_ae\_Tree1} : \text{owl\_ae\_Tree}\\&\text{owl\_ae\_s\_tree1} : \text{owl\_ae\_Tree}\\&\text{owl\_ae\_Tree1} \sqsubseteq \exists\ \text{hasInstance}.\{\text{owl\_ae\_s\_tree1}\}\end{aligned}$$

The above axioms state that *owl_ae_Tree* is a subclass of the *Entity* class, *owl_ae_Tree1* and *owl_ae_s_tree1* are individuals of type *owl_ae_Tree*, and *owl_ae_s_tree1* is an instance of *owl_ae_Tree1*.

5.4.4 Recurring substructures

The example with two trees seen in Section 5.4.3.1 has accidentally served another purpose by bringing up a rather interesting question: if a student learns something about one of the trees, haven't they learned something about the other one as well? The problem at hand is a peculiar one, as it suggest the information in such cases should flow both ways. We already know that loops are not allowed in Bayesian networks. That means that a direct connection from one instance to another, as shown in Figure 5.11a, wouldn't work. Another approach, preserving the guess and slip factor for instance-specific question nodes, would also result in an illegal Bayesian network structure (Figure 5.11b). Many other options also fail.

Therefore, we avoid the cycle trap through the idea of *answer collectors*. It should be noted that this is not a solution to the BN cycle problem, but a workaround. This approach adds an additional layer to the network, between the low level concepts and the question (outcome) nodes. The structure in Figure 5.12 shows a low level concept called *PositiveProportionality_AH* and three of its instances. Each answer collector, as its name implies, *collects answers* coming from the outcome nodes and distributes it among all of the instance nodes. Moreover, the idea of guesses and slips is preserved.



(a) Direct connection (b) Via question nodes

Figure 5.11: Bidirectional flow of information: illegal BN structures

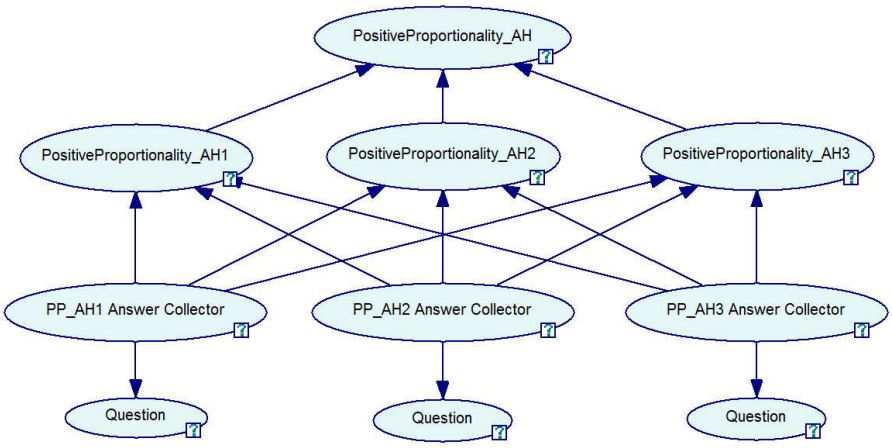


Figure 5.12: Recurring concept knowledge distribution via answer collectors

5.4.5 Options for updating knowledge

When we compare the above approach and the resulting network structure to the one proposed by Reye (2004), it becomes evident that only the bottom two layers fits the picture seen in Section 4.3.2. This is sufficient if we want to keep track of a learner's state of knowledge about a single instance, and independently of any other instances of the same generic concept. In order to be able to track the learner's knowledge at both the instance level and the generic node level, the whole substructure would have to become temporal (dynamic). This means that, for each interaction/time slice, instead of n additional nodes, we would be adding a total of $2n$ nodes, which could result in an explosion in the number of states for a large number of time slices, and represent a network complexity problem for larger models.

5.4.5.1 Other possibilities

Soft evidential updating – Should we still decide to track only the knowledge about individual instances, independently of other "duplicates", we could reduce the number of additional nodes per time slice even further, i.e. to $n/2$, by relying on the idea of *soft*

evidence.

As Valtorta et al. (2000) explain, evidence is a collection of hard or soft findings on variables. Whereas a hard finding specifies which value a variable is in, a soft one specifies a probability distribution of a variable. Soft evidence is a collection of soft findings. The following example explains this. Suppose that the initial position and direction of an object are known, and its acceleration is zero. Any future position (P) of the object can then be determined by its actual speed (S) according to some probabilistic law. This relationship can be represented easily in terms of a Bayesian network (Figure 5.13).

Figure 5.13: Position and speed (Valtorta et al., 2000)

Now, suppose we don't have precise information about the actual speed, i.e. the information we have is uncertain. Try thinking of E as a separate agent (sensor) providing us with an estimate of current speed (Figure 5.14). Therefore, this estimate is a belief, a probability distribution over the possible values of S. This kind of evidence is called *soft* evidence.

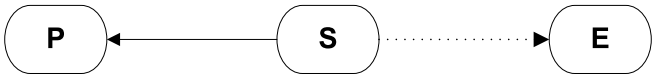


Figure 5.14: Position and speed with soft evidence about speed (Valtorta et al., 2000)

In other words, instead of saying a concept is simply *known* or *unknown*, we could directly involve uncertainty using soft evidence and, after an interaction, say something is, for instance, *65% known*. Then, by leaving only the question node dynamic, we could mimic Reye's (2004) approach by directly setting the probabilities for each time slice, with a controlled level of uncertainty.

No temporal nodes – The last option is to abandon the idea of time entirely (this doesn't mean we should ignore the concept of interaction history, though). One way to do this is to assign multiple question nodes to each concept (i.e. one node for each question). However, in order to preserve the idea of guesses and slips, and merge the new approach with the rest of the architecture seen so far, we would need to add another, intermediate layer (as the direction of the edges pointing to question nodes would have to be reversed). Figure 5.15 shows what this new substructure would look like for an answer collector with three questions.

5.4.5.2 How many questions?

An obvious question that remains to be answered is how many questions we need to ask for each concept in order to mark it as "known".

Concept types – First of all, we need to be aware of the number of possible question types the question generator can ask for each concept. This can also serve as a relevance

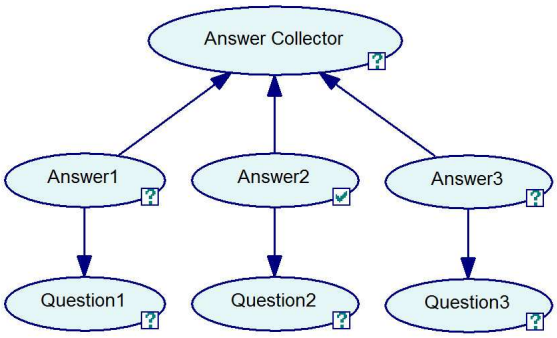


Figure 5.15: Multiple questions, no time slices

measure for each concept type. For instance, QUAGS can ask only one question about a quantity space:

- `Which values can Quantity5 adopt?`

On the other hand, multiple questions can be asked about magnitudes, derivatives, (in)direct influences etc. For example, below is a number of possible questions for a direct influence relationship between two quantities:

- `Is the influence from Quantity2 on Quantity4 effective?`
- `Why does Quantity4 decrease?`
- `Which quantities have a direct influence on Quantity4?`
- `How does Quantity2 influence Quantity4?`

Domain difficulty – What is important regarding the above example is the fact that even more questions are possible. However, although QUAGS does reduce the number of possible questions using a number of methods, we should have additional criteria to select an appropriate number of questions for each concept, or, what's even more important, each scenario/model. That is, a tutor would probably want to have a more extensive examination session with a student for a domain they consider more difficult. When it comes to Qualitative Reasoning models, domain difficulty could be measured in terms of model, or even simulation *complexity*. However, an appropriate complexity measure would have to be a relative one, taking into consideration a large library of models, scaling from simple domains to complex ones. We're saying "simple" and "complex", instead of "small" and "large", as a large model is not necessarily a complex one, as it may produce a simpler state graph than a considerably smaller model.

Quantity spaces vs. network size - As only a single question can be asked about a quantity space, not taking quantity spaces into consideration as separate concepts is also an option. Each quantity node (both generic and scenario specific) requires adding an

additional node to represent its quantity space. Therefore, by excluding such nodes, we would be reducing the size of the network by a number corresponding to the total number of quantity nodes. As a substitute solution, we could attribute quantity space specific questions to magnitudes.

5.4.5.3 Low level concept impact

In the simplest setup, we assume each of the low level concept nodes contributes to the higher level nodes equally. In the example seen in Figure 5.2, for instance, that would mean that each of the four nodes connected to the *Size* node would "weigh" 25% in terms of knowledge. However, one could disagree that the concepts are equally important, and say that, for example, the knowledge about a magnitude is more important than the one about an influence. This could be regarded as a matter of preference.

5.4.5.4 Answer collector impact

One might argue that an instance specific answer collector should "share" more knowledge with the instance it belongs to, i.e. have a little more impact on it than the other collector nodes. We can try to assign a weight, e.g. X, that will help us distribute the probabilities less "fairly". For example, in the above mentioned example with 3 instances, with a fair probability distribution, the definition table for PositiveProportionality_AH1 looks as given in Table 5.1 (K = Known, U = Unknown).

PositiveProportionality_AH1								
PP_AH1_Answer Collector	K				U			
PP_AH2_Answer Collector	K		U		K		U	
PP_AH3_Answer Collector	K	U	K	U	K	U	K	U
Known	1	0.67	0.67	0.33	0.67	0.33	0.33	0
Unknown	0	0.33	0.33	0.67	0.33	0.67	0.67	1

Table 5.1: CPD table for the PositiveProportionality_AH1 node

Therefore, if the instance specific answer collector is known, the algorithm should boost the probabilities by the amount assigned to the weight variable, and if it is unknown, the probabilities should be decreased by the same amount. To be more specific, currently, the probability of a node being known (K) is calculated as follows:

$$K = \frac{k}{N} \tag{5.1}$$

where N is the number of instances and k the number of known instances. By altering the distribution process as explained above we get:

1. if the instance specific answer collector is known:

$$K = \frac{k}{N} + \frac{X}{N} \tag{5.2}$$

2. if the instance specific answer collector is not known:

$$K = \frac{k}{N} - \frac{X}{N}. \tag{5.3}$$

We normalize the weight (X/N), so the probabilities get boosted/decreased by a normalized "fraction of knowledge" rather than by a fixed number (we don't want to add/subtract the same amount, e.g. 5%, to an instance with 1 and 10 duplicates. The modified CPD for $X = 0.06$ can be seen in Table 5.2.

PositiveProportionality_AH1								
PP_AH1_Answer Collector	K				U			
PP_AH2_Answer Collector	K		U		K		U	
PP_AH3_Answer Collector	K	U	K	U	K	U	K	U
Known	1	0.73	0.73	0.39	0.61	0.27	0.27	0
Unknown	0	0.27	0.27	0.61	0.39	0.73	0.73	1

Table 5.2: Revised CPD table for the PositiveProportionality_AH1 node

Figure 5.16 depicts this idea. The displayed structure uses the non-dynamic approach (i.e. a fixed number of questions) shown in Figure 5.15 for the sake of clarity. We can see that one of question nodes was marked as known. Here, we can also see the guess factor at work – the reason the answer node right above the question one shows the probability of 99% instead of 100%, as one may expect, is the fact that the probability of a guess is set to 0.01 (i.e. 1%). This further boosts the probability of the instance-specific answer collector to 74% and the probability of knowing that particular instance (*PositiveProportionality_AH1*) to 60%. At the same time, the other instances also receive an increase in probability, but the impact of the external answer collector is slightly less, as expected, so the new value for each of the duplicates is 57%. Each of the instances contributes equally to the generic concept, so the probability of knowing *PositiveProportionality_AH* is now 58%.

5.5 Final Bayesian network structure and evaluation

For the sake of evaluating our work, we have chosen the well known *Communicating vessels* (also known as the *U-tube*) model (Bredeweg et al., 2006). It is considerably more complex than the *Tree and shade* model, and it has been used on numerous occasions to illustrate problems and solutions concerning many elementary issues in QR. But, what's much more important, it contains a number of issues discussed in this chapter which were not covered by Brielmann's approach, including inequalities, equations and recurring substructures.

The domain is represented by multiple containers containing a homogeneous fluid, connected to each other at the bottom by a pipe. If there is more liquid in any of the vessels, the difference in hydrostatic pressure will make the liquid flow out of the container with the higher amount of content into the other vessels via the pipe, until it eventually finds an equilibrium (balances out to the same level in all of the containers). The pressure depends

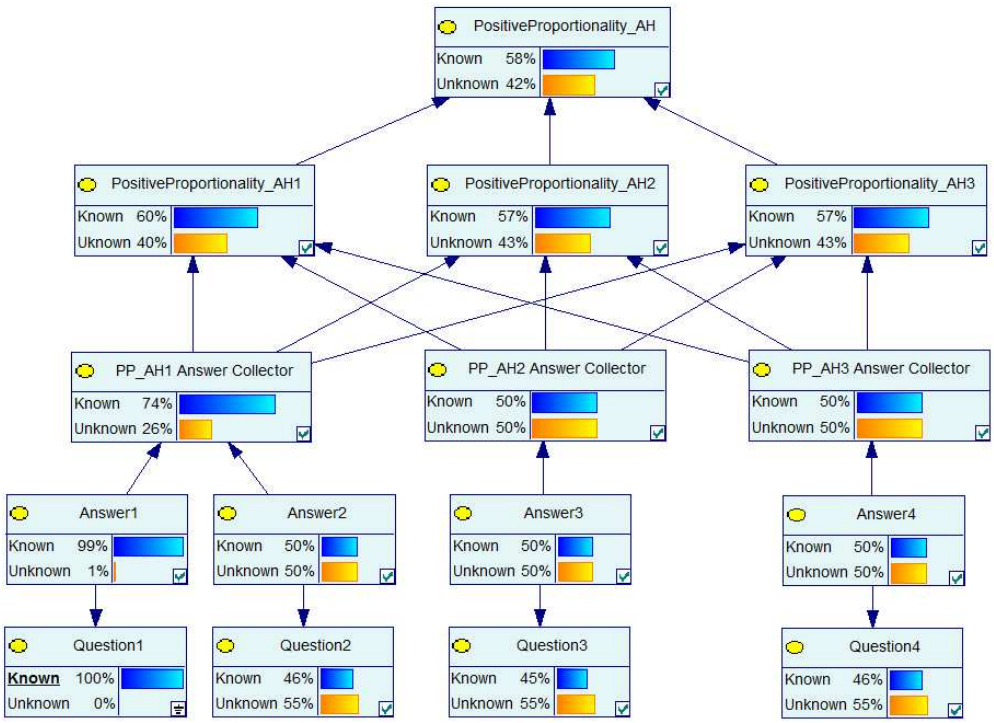


Figure 5.16: Answer collector impact with instance specific weights

only on how far the liquid surface is from the bottom of the container (the liquid is acted upon by gravity), and is not affected by the shape of the container.

We will not discuss all of the model details in this section, as the purpose of getting an insight into the model is merely to better understand the belief network structure we will use for evaluation.

5.5.1 Communicating vessels model fragments

The structure and behavior of the communicating vessel system is described by two model fragments. The *Contained liquid* MF (Figure 5.17) models a single container with liquid. The quantities *Amount*, *Height*, and *Pressure* are introduced as consequences of a *Liquid* entity, contained inside a *Container*. Since we know that if *Amount* changes, *Height* changes in the same direction, but so does *Pressure*, we model these relationships using positive proportionalities in appropriate directions. Moreover, all of the quantities also have to be in the same interval at the same time. This is modeled using quantity space correspondences. Finally there is an equality relation between *Height* and *Pressure*.

The second model fragment, named *Liquid flow* describes the communicating vessels mechanism for two containers, and is given in Figure 5.18. The *Contained liquid* MF serves as a condition twice, i.e. once for each container. A conditional *Pipe* entity instance connects the two containers, and introduces the *Flow* quantity, which, when in the positive interval, increases *Amount* in the right container and decreases the one in the left one via a positive and

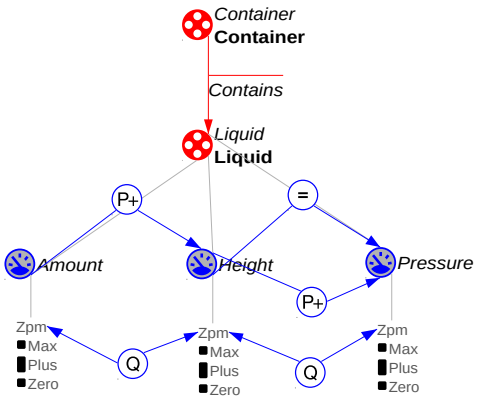


Figure 5.17: Communicating vessels: Contained liquid model fragment

a negative influence relationship, respectively. The actual magnitude of *Flow* is calculated as the difference between the magnitudes of the two *Pressure* quantities, i.e. by subtracting *Pressure* on the right side from the pressure on the left side. In order for the simulation engine to be able to determine *Flow*'s derivative, the change is described using a positive proportionality from *Pressure* on the left hand side, and a negative one from *Pressure* on the right hand side. That is, when the pressure of the left container increases, the flow in the pipe increases as well, and when the pressure in the right vessel increases, the liquid flow decreases.

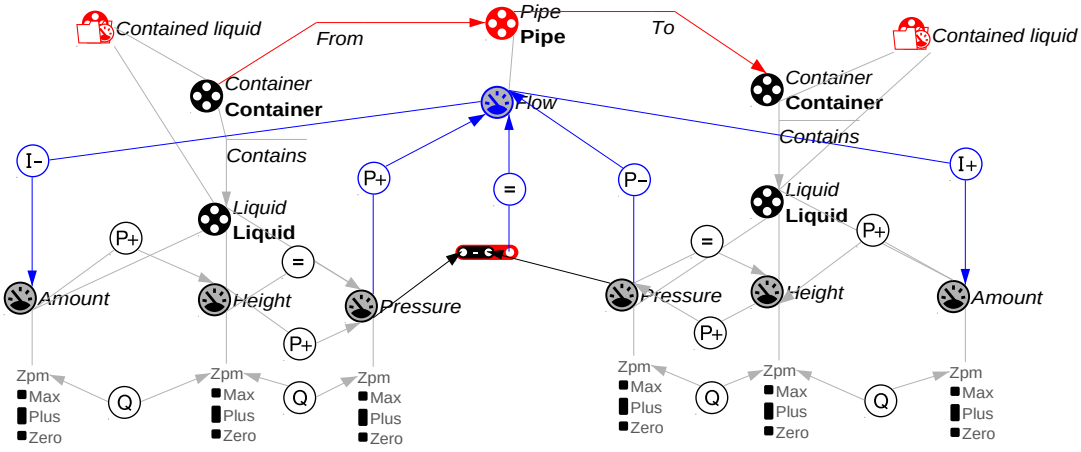


Figure 5.18: Communicating vessels: Liquid flow model fragment

5.5.2 Communicating vessels scenario

We will take only a single scenario into consideration and construct our network based on it. The scenario is called *Both tanks partially filled but left is higher* and is shown in Figure 5.19. It represents two containers, partially filled with oil (magnitudes set to *plus*), where the height of the liquid column on the left is greater than the one on the opposite side (modeled using an inequality relationship).

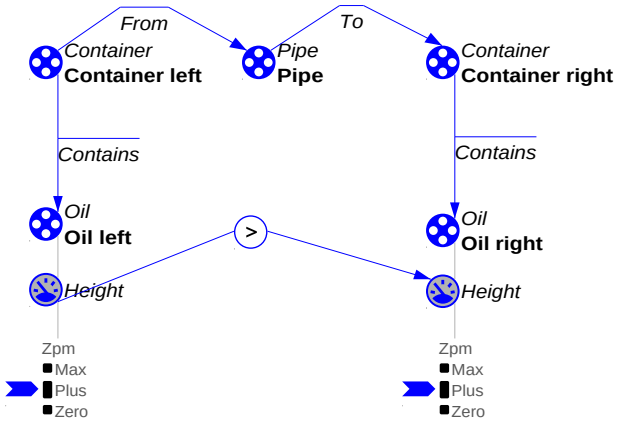


Figure 5.19: Communicating vessels: Both tanks partially filled but left is higher scenario

5.5.3 Final Bayesian network structure

The final structure of the Bayesian network for the *Communicating vessels* model is given in Appendix B[3]. We can distinguish five basic node levels/layers in the structure.

The upper half of the network / topmost level contains only nodes representing generic concepts. The names of these nodes are unique across the scenarios, so we could move the information from one scenario/network to another, as described in Section 5.8.

Each one of the generic concepts is connected to its instance(s) in the lower (scenario-specific) half of the network, at level two. For instance, the *Pipe* entity occurs only once in the superstate, and, therefore, it has a single instance node representing this occurrence. The *Contained liquid* model fragment, on the other hand, "fires" twice due to the structure of the *Liquid flow* MF. Hence, each of the *Contained liquid* concepts at the generic level is connected to two instances at level two. Moreover, concept dependency at the second level is represented via direct links, as discussed in Section 5.2.

Each root node instance is connected to an answer collector at level three. If the root node instance is not unique, i.e. there are multiple instances of the same generic concept, every answer collector, in turn, is also connected to every other instance of the same concept.

Finally, depending on the structure choice, as seen in Section 5.4.5, the bottom two levels contain question nodes or question-answer node pairs, i.e. each answer collector bears a connection to either a single temporal question node or multiple non-temporal question/answer pairs.

[3]The Figure doesn't depict the bottom (question/answer) level(s), as this part of the network depends on the structure choice, as discussed in Section 5.4.5

5.5.3.1 Implementation choice

Unfortunately, our choice of options was constrained due to a number of bugs, and a lack of support for certain concepts (at the time of this writing) in the jSMILE API[4] the DynaLearn research group was working with. jSMILE is a platform independent library of Java classes for reasoning in graphical probabilistic models, such as Bayesian networks and influence diagrams.

To be more specific, the concepts that could not be implemented were the ideas of temporal nodes and soft evidence. Therefore, we were left with only one option, namely, to abandon the concept of time and determine the number of possible question nodes beforehand.

5.5.4 Choice of parameters

Table 5.3 lists the choice of parameters used for evaluation of the network. The prior probabilities of all concept nodes are set to 50%, i.e. the probabilities of a student knowing or not knowing a topic are the same. Furthermore, we assume that all parent (root) nodes of a concept contribute equally to the child (*concept impact* entry). For instance, if an entity has four quantities, each quantity node will be "worth" 1/4, i.e. 25% in terms of knowledge. The probability of a learner giving a correct answer, although he doesn't know it (i.e. guessing) is 0.1. The probability of answering incorrectly, while knowing the topic (i.e. slipping) is 0.01. Finally, the last seven entries in the table represent an experimental set of choices for the number of questions per concept, based on the number of possible questions that can be asked regarding each of those concepts by QUAGS.

Parameter	**Parameter value**
Concept node prior	0.5
Concept impact	1/#parents
Answer collector weight	0.06
Probability of a guess	0.1
Probability of a slip	0.01
Questions per magnitude	5
Questions per derivative	5
Questions per influence	5
Questions per proportionality	5
Questions per correspondence	2
Questions per inequality	2
Questions per calculation	1

Table 5.3: Choice of parameters

[4] http://genie.sis.pitt.edu/

5.5.5 Evaluation

We test the network given in Appendix B using the parameters shown in Table 5.3[5]. It is worth noting that the names of the Appendix B concepts don't match the names of the same concepts to be shown in evaluation graphs. The structure given in the Appendix was created manually and made to be more human-readable. The evaluation graph names correspond to the internal representation of each concept in OWL, as the network construction process has been automated. For each evaluation concept name, however, we will indicate which Appendix concept it corresponds to, for the sake of clarification.

We first evaluate a single root node, see how it's probability responds to evidence updates, and how this knowledge is propagated to upper level nodes. For each concept we test, we first visit all of the question nodes once, and arbitrarily add evidence. Then, we revisit the same nodes, and update any answer that was initially set to be incorrect, to correct, so the probability of the root node reaches its maximum value (for its answer collector; duplicate instance nodes are affected by other duplicates' answer collectors, too). Figure 5.20 shows how the system's belief of the user's knowledge of *Magnitude_s_7*[6] and *owl_q_s_flow1* (the quantity the magnitude belongs to) develops across eight learner-system interactions/questions. As given in Table 5.3, since the node is of type *Magnitude*, we ask five questions. The learner gives an incorrect answer to the third question, and the probability of both the magnitude and quantity concepts decreases. The remaining two answers are correct, though, and the probabilities go up. Then, as described above, we "fix" the single incorrect answer at interaction number six, so the concept of *Magnitude_s_7* becomes 91%, which is the maximum value that node can attain, due to the guess and slip factors and the fixed number of questions (we will discuss this point in Section 6.1). The *owl_q_s_flow1* quantity node reaches 56%, which is the maximum probability for that node if there is no evidence about the other parents of the same node.

The above test shows how the system's belief progresses when we're dealing with a single instance with no duplicates, i.e. only one answer collector. Next, we observe the behavior of a quantity instance which is not unique, namely, *owl_q_s_amount1*[7] through it's magnitude, i.e. *Magnitude_s_3* (Figure 5.21). As the first four answers are all wrong, the probability of knowing the magnitude goes down to only 24%. Meanwhile, the probability of knowing the quantity it belongs to drops down to 45%, and the second instance of the *Amount* quantity decreases slightly less, i.e. to 46%, due to the answer collector impact parameter. Then, the remaining question is answered correctly, and the examination session is extended by another four interactions, so all of the answers are fixed. The magnitude concept gets to 72%

[5]At the time of evaluation, the DynaLearn environment, i.e. QUAGS, wasn't ready for testing. To be more specific, automatic question focus selection and question generation were not possible. Therefore, we chose to simulate parts of an examination session, manually selecting the focus, and updating the concepts (simulating learner's answers) arbitrarily.

[6]Appendix B: $Magnitude\_s\_7 = Magnitude\_Flow; owl\_q\_s_flow1 = Flow1$

[7]Appendix B: $Magnitude\_s\_3 = Magnitude\_Amount1; owl\_q\_s_amount1 = Amount1; owl\_q\_s_amount2 = Amount2$

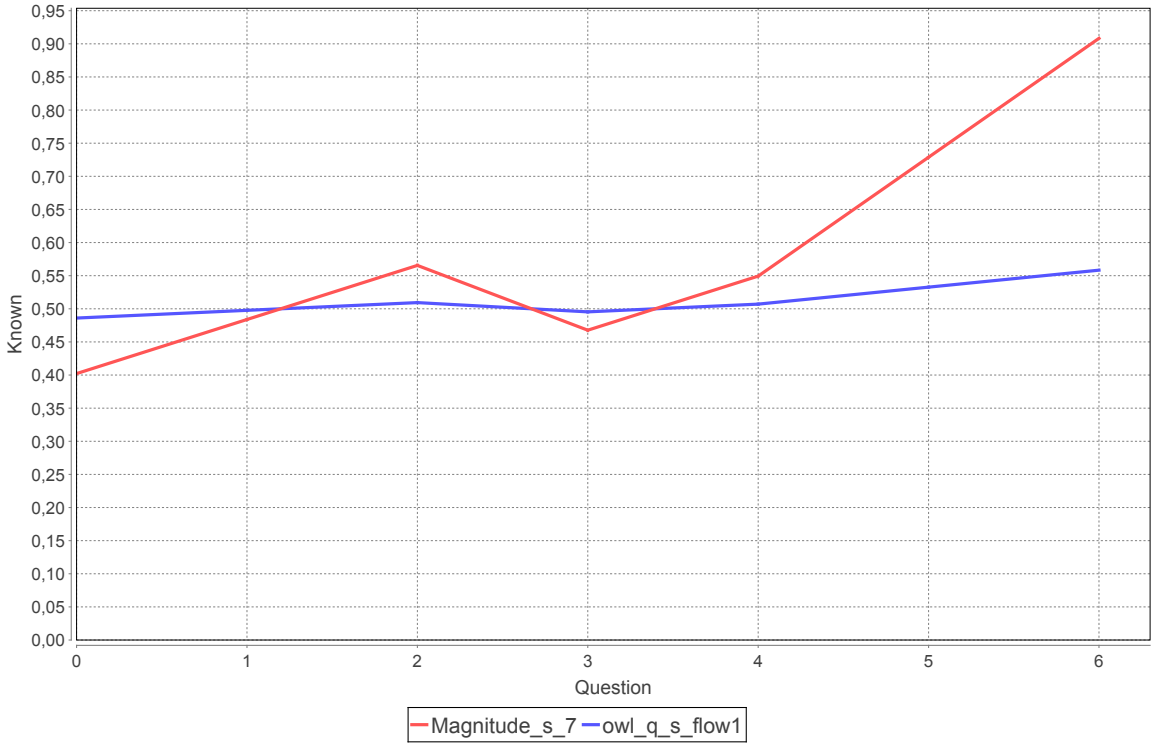


Figure 5.20: Learning chart for the flow quantity's magnitude

after interaction number nine, which is the maximum it can reach, as the second instance holds the remaining knowledge. At the same time, *owl_q_s_amount1* increases to 54%, and the second instance to around 53%, which is now slightly less, but again, due to the answer collector impact.

Instead of a root node, we now turn the focus to a whole quantity, namely, *owl_q_s_flow1*[8]. We test all concepts related to the quantity[9] using the same method as seen above, and watch the system's belief about both the parent nodes and the child develop. We also see the system asks a single question about *Minus_s_1*, as the quantity is part of the equation seen in Figure 5.18. The probability of knowing the quantity slowly progresses towards the maximum (91%).

In Figure 5.23 we show the results of a session focusing on the quantity *owl_q_s_amount1*[10]. We test all quantity related concepts, and show the results for both the quantity and the second instance of the *Amount* quantity (*owl_q_s_amount1*). We see that as the probability

[8]Appendix B: $Derivative\_s\_7 = Derivative\_Flow$; $Minus\_s\_1 = Calculation\_PPF$; $NegativeInfluence\_s\_1 = NegativeInfluence\_FA$; $PositiveProportionality\_s\_5 = PositiveProportionality\_PF$; $NegativePropotionality\_s\_1 = NegativeProportionality\_PF$; $PositiveInfluence\_s\_1 = PositiveInfluence\_FA$

[9]Here, we're not considering quantity spaces as separate concepts, as proposed in Section 5.4.5.2.

[10]Appendix B: $Derivative\_s\_3 = Derivative\_Amount1$; $QuantitySpaceCorrespondence\_s\_1 = Correspondence\_AH1$; $PositivePropotionality\_s\_1 = PositiveProportionality\_AH1$

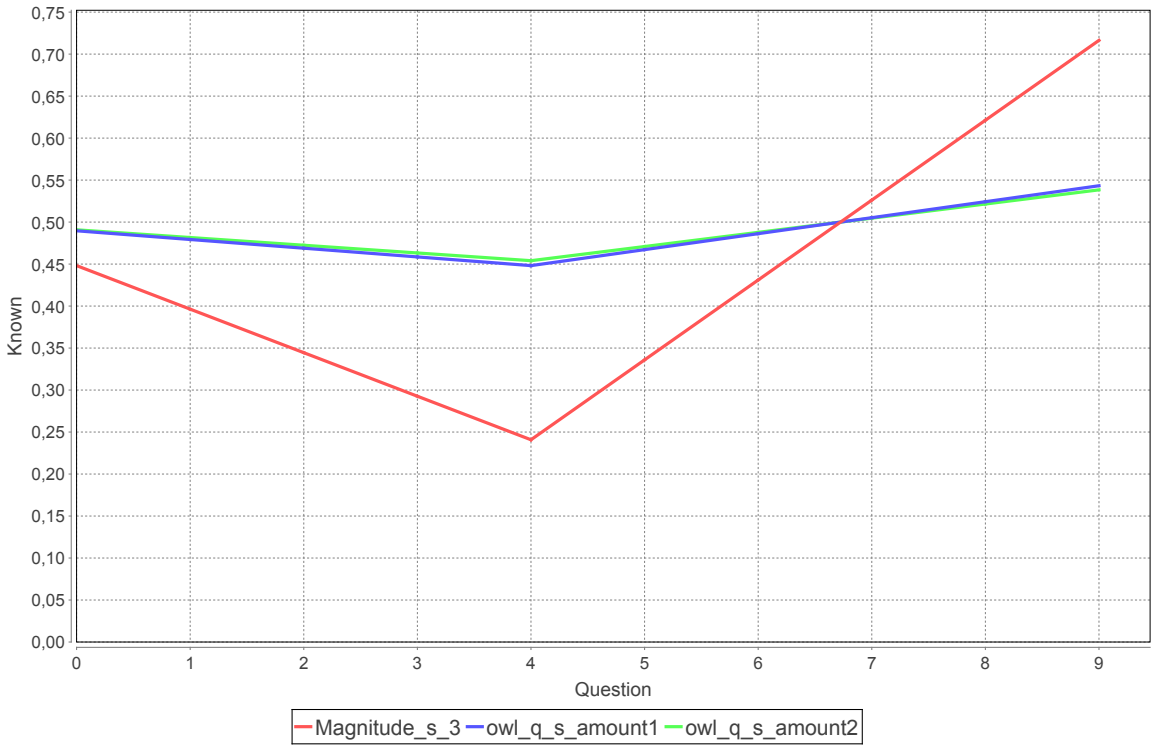


Figure 5.21: Learning plot for the amount quantity's magnitude

of knowing first instance increases, so does the probability of understanding the second one, only at a slightly slower rate. The breaking point is at question number 24, where the system's belief about the second instance stops changing. This is due to the fact that questions 24–28 are related to the concept of *NegativeInfluence_s_1*, i.e. the negative influence relationship between the quantities *Flow* and*Amount* (Figure 5.18). This information is only *Liquid flow* model fragment specific and, as such, carries no information about the second instance of the *Amount* quantity.

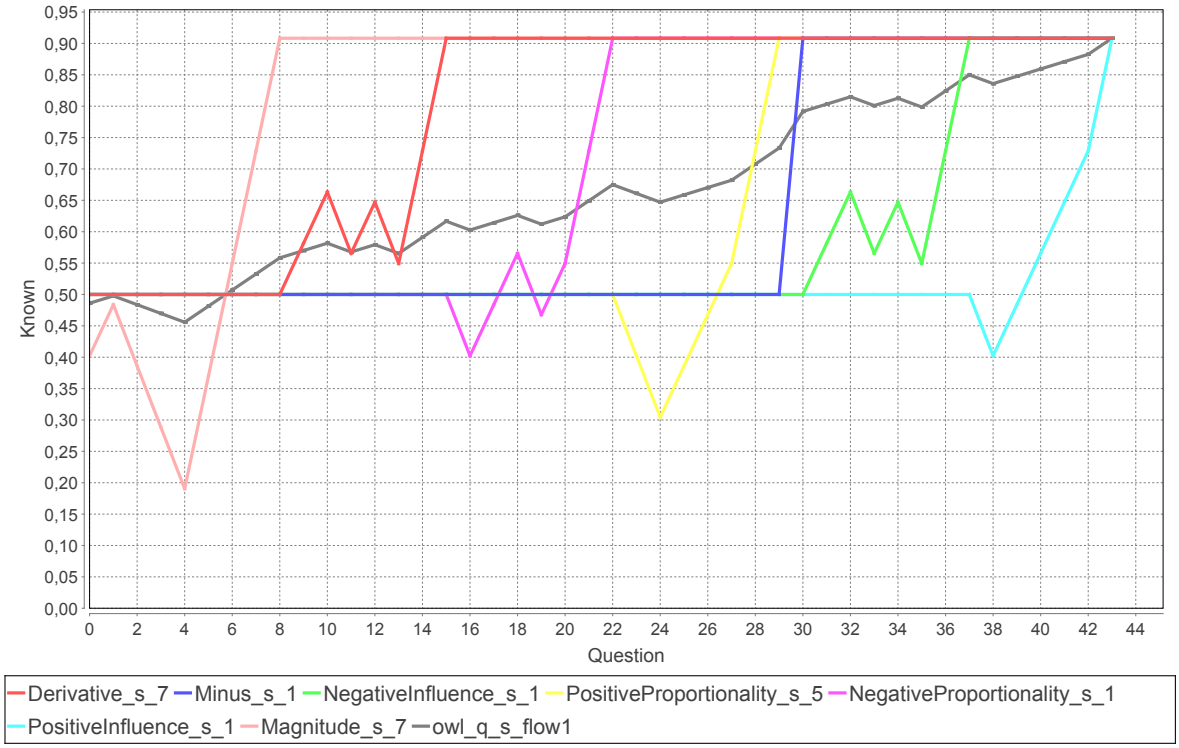


Figure 5.22: Learning plot for the flow quantity and related concepts

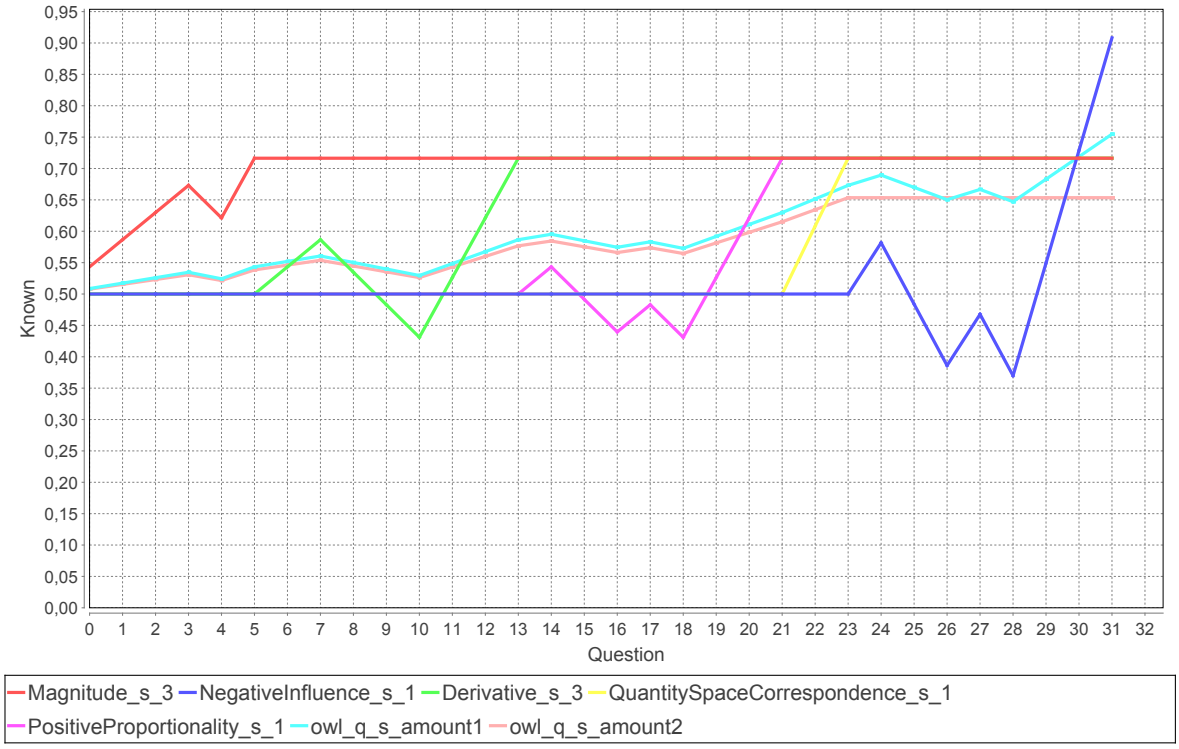


Figure 5.23: Learning plot for the amount quantity and related concepts

Chapter 6

Conclusion

We have managed to move from the general knowledge tracing method proposed by Corbett and Anderson (1995) and later embodied as a belief network by Reye (2004), following the path of Brielmann (2009), to a successful approach to learner modeling in the context of Qualitative Reasoning. In summary, Brielmann's idea was first taken to a higher level with the proposal for a global architecture and tracing the learner's knowledge state across scenarios and models. Then, at a lower level, a more detailed approach was taken to constructing the learner's knowledge in terms of a belief network. We have seen more concepts that need to be taken into consideration while collecting information about a learner's understanding of a domain. We have also seen how to deal with uncommon structures, such as mathematical dependencies, but also how to move knowledge between recurring substructures. The possibilities for updating knowledge and determining the number of questions that need to be asked have also been discussed. Also, we tackled the idea of the impact that instance specific questions should have on their instance nodes and the duplicates of that node. Finally, the applicability of the approach was proven as we saw the system build up its belief about a learner's state of knowledge of different parts of a QR model, during a simulated evaluation session.

6.1 Discussion

Brielmann's approach was considerably improved, as many fundamental issues related to constructing a Bayesian network from a Qualitative Reasoning model were successfully solved. However, the evaluation of our work was partially hampered by external factors. Therefore, we were unable to thoroughly test the original idea that included the concept of time. The alternative approach showed a lot potential. It provides the system with the means to successfully gather information about a student's understanding of a domain. Still, it appears some obstacles need more work, whereas some others seem to be impossible to overcome. For instance, selecting an appropriate number of questions per concept type beforehand needs to be further investigated, as it can depend on many factors. On the other hand, the lack of temporal nodes makes it impossible to see the learner's knowledge of any concept go above a limit set/constrained by the guess and slip parameters. Moreover, Reye's idea of

learning/forgetting rates also had to be left out.

6.2 Future work

Although the progress made with this approach is considerable, there is still plenty of room for future work. One obvious improvement is implementing and testing the alternative methods for updating a user's knowledge (both temporal nodes and soft evidence). This would solve the above mentioned problems regarding the maximum probability that can be achieved for any node, and learning/forgetting rates. Another idea from Reye's approach is a proposal for global nodes. Ideas such as a student's aptitude or boredom, however, require more (larger) models and more thorough examination sessions. Also, the idea of different concept types carrying different amounts of information should be given more attention.

We could/should also keep track of the student-system interaction history. More specifically, on the learner's side, we could track the types of questions being asked (so we don't ask the same question multiple times), or track the user's overall progress. On the system's side, we would gain information such as which parts of a model, or which models students are having problems with, and therefore have an indication of a concept difficulty or model's complexity (which could in turn help us adjust the concept impact or set the number of questions we would like to ask). Ideally, this approach should work in sync with the global architecture, which is also yet to be implemented, tracking the users and interactions across scenarios and models.

Another topic that should be investigated further is the idea of "landmarks". For instance, certain model fragments have conditions, such as a specific quantity value that needs to be reached in order for the MF to get activated. This information appears to be crucial, and, as such, should be properly represented in the learner model.

With these ideas in mind, and the current state of work on understanding the process of learners gaining and constructing conceptual knowledge of system behavior, the DynaLearn project is provided with the necessary means and a strong foundation for future research.

Bibliography

Allaby, M. 2000. *Basics of Environmental Science*. Routledge.

Antoniou, G. and van Harmelen, F. 2004. *A Semantic Web Primer*. MIT-Press.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F. 2004. *Description Logic Handbook*. Cambridge University Press.

Baker, R. S., Corbett, A. T., and Aleven, V. 2008. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *ITS '08: Proceedings of the 9th international conference on Intelligent Tutoring Systems*, pages 406–415. Springer-Verlag.

Beck, J. E. 2007. Difficulties in inferring student knowledge from observations (and why you should care). In *Educational Data Mining: Supplementary Proceedings of the 13th International Conference of Artificial Intelligence in Education*, pages 21–30.

Beck, J. E. and Chang, K.-M. 2007. Identifiability: A fundamental problem of student modeling. In *UM '07: Proceedings of the 11th international conference on User Modeling*, pages 137–146. Springer-Verlag.

Bishop, C. M. 2009. *Pattern Recognition and Machine Learning*, chapter 8, pages 359–418. Springer.

Bobrow, D. G. 1984. Qualitative reasoning about physical systems: An introduction. *Artificial Intelligence*, 24(1-3):1–5.

Bredeweg, B., André, E., Bee, N., Bühling, R., Gómez-Pérez, J. M., Häring, M., Liem, J., Linnebank, F., Nguyen, B. T. T., Trna, M., and ner, M. W. 2009a. Technical design and architecture. *DynaLearn, Project deliverable report D2.1*.

Bredeweg, B., Bouwer, A., Liem, J., and Salles, P. 2006. Curriculum for learning about qr modelling. *Naturnet-Redime, STREP project co-funded by the European Commission within the Sixth Framework Programme (2002-2006), Project no. 004074, Project deliver-*

able report D6.9.1.

Bredeweg, B., Gmez-Prez, A., Andr, E., and Salles, P. 2009b. Dynalearn - engaging and informed tools for learning conceptual system knowledge. In *Proceedings of the AAAI Fall Symposium Series.*

Bredeweg, B., Linnebank, F., Bouwer, A., and Liem, J. 2009c. Garp3 - workbench for qualitative modelling and simulation. *Ecological Informatics*, 4(5–6):263–281.

Bredeweg, B., Salles, P., Bouwer, A., Liem, J., Nuttle, T., Cioaca, E., Nakova, E., Noble, R., Caldas, A. L. R., Uzunov, Y., Varadinova, E., and Zitek, A. 2008. Towards a structured approach to building qualitative reasoning models and simulations. *Ecological Informatics*, 3(1):1–12.

Bredeweg, B. and Winkels, R. 1998. Qualitative models in interactive learning environments: An introduction. *Interactive Learning Environment (special issue)*, 5(1–2):1–18.

Brielmann, M. 2009. A learner model based on a bayesian network for garp3. Master's thesis, AMSTEL Institute, University of Amsterdam.

Brown, J. S., Burton, R. R., and de Kleer, J. 1982. Pedagogical, natural language and knowledge engineering techniques in sophie i, ii and iii. In *Intelligent Tutoring Systems*, pages 227–282, New York, NY, USA. Academic Press.

Cen, H., Koedinger, K. R., and Junker, B. 2007. Is over practice necessary? - Improving learning efficiency with the cognitive tutor through educational data mining. In *Proceedings of the 2007 conference on Artificial Intelligence in Education*, pages 511–518. IOS Press.

Cioaca, E., Linnebank, F., Bredeweg, B., and Salles, P. 2009. A qualitative reasoning model of algal bloom in the danube delta biosphere reserve (ddbr). *Ecological Informatics*, 4(5-6):282–298.

Collins, A. M. and Stevens, A. L. 1982. Goals and strategies for inquiry teachers. *Advances in Instructional Psychology*, 1:65–119.

Collins, J. A., Geer, J. E., and Huang, S. X. 1996. Adaptive assessment using granularity hierarchies and bayesian nets. In *ITS '96: Proceedings of the Third International Conference on Intelligent Tutoring Systems*, pages 569–577. Springer-Verlag.

Conati, C., Gertner, A. S., Vanlehn, K., and Druzdzel, M. J. 1997. On-line student modelling for coached problem solving using bayesian networks. In *Proceedings of the Sixth International Conference on User Modelling*, pages 231–242.

Corbett, A. T. and Anderson, J. R. 1995. Knowledge tracing: Modelling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278.

de Kleer, J. and Williams, B. C. 1987. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130.

Forbus, K. D. 1984. Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Forbus, K. D. 1988. Qualitative physics: past, present, and future. In *Exploring Artificial Intelligence*, pages 239–296, San Mateo, USA. Morgan Kaufmann.

Forbus, K. D. 1997. Qualitative reasoning. In Tucker, A. B., editor, *The Computer Science and Engineering Handbook*, pages 715–733. CRC Press.

Goddijn, F., Bouwer, A., and Bredeweg, B. 2003. Automatically generating tutoring questions for qualitative simulations. In *Proceedings of the 17th International Workshop on Qualitative Reasoning*, pages 87–94.

Haupt, S. E., Pasini, A., and Marzban, C. 2008. *Artificial Intelligence Methods in the Environmental Sciences*. Springer Publishing Company, Incorporated.

Hollan, J. D., Hutchins, E. L., and Weitzman, L. M. 1984. Steamer: An interactive, inspectable, simulation-based training system. *AI Magazine*, 5:15–27.

Horrocks, I., Patel-Schneider, P. F., and Harmelen, F. V. 2003. From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics*, 1:2003.

Jackson, S. L., Krajcik, J., and Soloway, E. 1998. The design of guided learner-adaptable scaffolding in interactive learning environments. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 187–194, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Koedinger, K. R. 2002. Toward evidence for instructional design principles: Examples from cognitive tutor math 6. invited paper. In *Proceedings of PME-NA XXXIII (the North American Chapter of the International Group for the Psychology of Mathematics Education)*, pages 21–49.

Lerner, B. W. and Lerner, K. L. 2009. *Environmental Science: In Context*. Gale, Cengage Learning.

Liem, J. and Bredeweg, B. 2006. Document Type Definition (DTD) for QR model fragments redime. *NaturNet - Redime, Specific targeted research project.*

Millán, E., Pérez-de-la Cruz, J.-L., and Suárez, E. 2000. Adaptive bayesian networks for multilevel student modelling. In *ITS '00: Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, pages 534–543. Springer-Verlag.

Mislevy, R. J. and Gitomer, D. H. 1995. The role of probability-based inference in an intelligent tutoring system. *User Modeling and User-Adapted Interaction*, 5(4):253–282.

Mitchell, T. M. 1997. *Machine Learning*, chapter 6. McGraw-Hill, New York, NY, USA.

Murphy, K. P. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, Computer Science Division, UC Berkeley.

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York : Basic Books.

Philander, S. G. 2008. *Encyclopedia of Global Warming and Climate Change*. SAGE Publications, Inc.

Resnick, M. 1994. *Changing the centralized mind*. MIT press, Cambridge, MA.

Reye, J. 2004. Student modelling based on belief networks. *International Journal of Artificial Intelligence in Education*, 14(1):63–96.

Richmond, B. and Peterson, S. 1992. *STELLA II: An introduction to systems thinking*. High Performance Systems, Hanover, NH.

Salles, P. and Bredeweg, B. 2006. Modelling population and community dynamics with qualitative reasoning. *Ecological Modelling*, 195(1–2):114–128.

Simmons, R. 1986. Commonsense arithmetic reasoning. In *Proceedings of AAAI-86*, pages 118–124, Menlo Park, USA. AAAI Press.

Staudt, A., Huddleston, N., and Kraucunas, I. 2009. *Understanding and Responding to Climate Change.* National Academies.

Valtorta, M., gyun Kim, Y., and Vomlel, J. 2000. Soft evidential update for probabilistic multiagent systems. *International Journal of Approximate Reasoning*, 29:71–106.

W3C, World Wide Web Consortium. 2009. OWL Web Ontology Language overview. *<http://www.w3.org/TR/2003/CR-owl-features-20030818/>*.

Appendix A

Sharing and Reusing Qualitative Reasoning Knowledge

To be able to store, share and reuse models and model fragments within a community building qualitative models, a format needs to be defined that allows the exchange of these fragments via the Internet. One such language would need to have a clear, well-defined syntax, a formal semantics and provide sufficient expressive power, efficient reasoning support and convenience of expression.

A.1 OWL

Description Logics (DL; formerly known as Terminological Systems or Terminological Logics) are a family of knowledge representation languages based on first-order logic (FOL). Revolving mainly around concepts (entities), roles (relationships between the entities) and role restrictions, DL were created in attempt to switch from the frame and network systems which lacked formal semantics to a logic-based approach to knowledge representation (Baader et al., 2004). As DL are based on FOL, concepts in DL can be seen as unary predicates, while roles can be thought of as binary predicates. The increasingly popular DL family members $\mathcal{SHIQ}$ and $\mathcal{SHOIN}$ are both extensions of DL $\mathcal{ALC}$ (Attributive Language with Complement), so we will illustrate Description Logics through this language.

DL $\mathcal{ALC}$ allows concepts, concept hierarchies, role restrictions and the Boolean combination of concept descriptions. The terminological knowledge consisting of concept definitions in an $\mathcal{ALC}$ knowledge base is represented in the *TBox*, whereas the *ABox* holds the assertions about the actual individuals. In Description Logics, concept expressions are variable free; an FOL expression denoting the intersection of two concepts, i.e. saying we would like to consider only the individuals belonging to both concepts, for instance, C and D, written as $C(x) \wedge D(x)$, in DL would be represented simply as $C \sqcap D$. In the TBox, the concept axioms can be of the form $C \sqsubseteq D$ (C is *subsumed* by D; the extension of C is a subset of the extension of D) or $C \equiv D$ (meaning $C \sqsubseteq D$ and $D \sqsubseteq C$). Furthermore, we can build the concept descriptions according to the following syntax rule:

$$
\begin{array}{rl}
C,D \longrightarrow & \text{A (atomic concept)} \\
& \top \text{ (universal concept)} \\
& \bot \text{ (bottom concept)} \\
& C \sqcap D \text{ (intersection)} \\
& C \sqcup D \text{ (union)} \\
& \neg\ A \text{ (negation)} \\
& \exists\ R.C \text{ (existential restriction)} \\
& \forall\ R.C \text{ (universal restriction)}
\end{array}
$$

It is worth noting that the more advanced DL family members support additional constructors, such as the cardinality restrictions, as we'll get to see later on. We define a formal semantics of the concepts by considering an interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function which to each atomic concept A assigns a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to each atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Hence, we can extend the interpretation function to the concept descriptions seen above by the inductive definitions shown below.

$$
\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg\ A)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus \mathcal{A} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \exists y.(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \forall y.(x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}
\end{array}
$$

Now assume we want to express that a mother is a person who is a female and at the same time has a child which is also a person. In DL syntax, we write this as:

$$\text{Mother} \equiv (\text{Person} \sqcap \text{Female}) \sqcap \exists\ \text{hasChild.Person}$$

The *OWL Web Ontology Language* was created in need of a language that could overcome the limited expressivity of RDF and RDF Schema, but at the same time combine their power with a full logic language (Horrocks et al., 2003; Antoniou and van Harmelen, 2004). The Semantic Web vision is to "build on XML's ability to define customized tagging schemes and RDF's flexible approach to representing data" (W3C, 2009). OWL makes use of the fact-stating ability of RDF and the class and property structuring capabilities of RDF Schema[1], but adds more to the Web document semantics through an ontology language that can formally describe the meaning of terminology used in those Web documents. In order to

[1]For the sake of saving some space in this thesis and discussing more important issues, we will not indulge in describing the structure and properties of RDF or RDF Schema in detail. For more information, please consult either Horrocks et al. (2003) or Antoniou and van Harmelen (2004).

fulfill the ontological requirements we have already mentioned, but also find a compromise between expressiveness and tractability, the W3C's Web Ontology Working Group defined three different "species" of OWL: *OWL Lite, OWL DL* and *OWL Full*[1]. Since each of the sublanguages is basically a syntactic extension of its simpler predecessor, each legal ontology in OWL Lite is a legal OWL DL ontology. Also, every valid OWL Lite conclusion will be valid in OWL DL as well. Moreover, the same relationships hold for OWL DL and OWL Full.

We've already mentioned that OWL builds on RDF and RDF Schema. Therefore, it uses RDF/XML syntax. However, this syntax is not very readable. Take for instance the following extract from a university ontology OWL file, defining the property *teaches*:

```
<owl:ObjectProperty rdf:about="#teaches">
   <rdfs:range rdf:resource="#Course"/>
   <rdfs:domain rdf:resource="#TeachingStaff"/>
   <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

To solve this problem, other syntactic forms have been defined, including an XML syntax which doesn't follow the RDF conventions and is easier to read. We will use OWL *abstract syntax* to illustrate OWL DL (the abstract syntax is not defined for OWL Full).

OWL Abstract Syntax	**DL**
Class(Professor partial TeachingStaff)	Professor $\sqcap$ TeachingStaff
ObjectProperty(isTaughtBy domain(course) range(TeachingStaff))	$\top \sqsubseteq \forall\ isTaughtBy^{-}.course$ $\top \sqsubseteq \forall$ isTaughtBy.TeachingStaff
Individual (John type(Professor))	John : Professor
Class(mathCourse partial restriction (isTaughtBy hasValue (John)))	mathCourse $\sqsubseteq \exists$ isTaughtBy.{John}

The expressions in the above table describe that *Professor* is a subclass of *TeachingStaff* and that a course can be taught only by a teaching staff member. Moreover, it says that John is a professor and that he teaches a math course.

[1]The details and differences of each of the languages will not be discussed here, as such a discussion would be out of the scope of this thesis. For more details please consult Baader et al. (2004).

A.2 Qualitative Reasoning models and simulations in OWL

The ordering of the QR vocabulary in an OWL class hierarchy is shown in Figure A.1 (Liem and Bredeweg, 2006). The graph shows the taxonomy of the QR model ingredients. Both QR concepts and relations are included, as the latter are reified (treated as classes). Since every class in the taxonomy is a possible ingredient of a qualitative model, the topmost class is called *QualitativeModelIngredient*. The model ingredients are split into two class sets, namely, *BuildingBlock* and *Aggregate*. Separate model ingredients belong to the former, while the latter describes collections of related model ingredients. As qualitative models describe both the structural and the behavioral aspects of systems, the building blocks are further divided into the sets *Structural* building blocks, *Behavioural* building blocks, and *AssumptionType* (section 8.1). As assumptions do not describe inherent aspects of the system, they are considered to be separate.

It is clear that ordering the QR vocabulary in a class hierarchy is not enough to formalize the whole domain of Qualitative Reasoning in the Web Ontology Language. Data properties, such as *has_xposition_on_screen*, make sure the same look and feel of each model is preserved after it's exported to OWL. Object properties ensure the relationships between concepts from Garp3 are maintained for each of the ingredients. Take the following quantity-specific axioms, stating that each quantity has a single magnitude and a single derivative, for example:

$$\begin{aligned}
&\text{Quantity} \sqsubseteq \forall\ \text{hasMagnitude.Magnitude}\\
&\text{Quantity} \sqsubseteq =1\text{hasMagnitude}\\
&\text{Quantity} \sqsubseteq \forall\ \text{hasDerivative.Derivative}\\
&\text{Quantity} \sqsubseteq =1\text{hasDerivative}
\end{aligned}$$

A.2.1 Representing models

One can think of a model specific OWL ontology as an augmented version of the QR vocabulary ontology. Each model ontology imports the QR ontology as a "foundation" on top of which model specific concepts are added. These concepts come in terms of subclasses and individuals. For instance, a new model fragment in the MF hierarchy is added to the QR vocabulary class hierarchy, as depicted in Figure A.2 (extracted from the Tree and shade model).

On the other hand, model fragment and scenario specific occurrences of concepts are added as individuals. For example, the *Size* quantity in the Tree and shade model is added to the class hierarchy as a subclass of the *Quantity* class. However, each of the three occurrences of this quantity (one in each model fragment, and one in the scenario) are added as individuals and named (*owl_q_*)*Size1*, *Size2* and *Size3*.

A.2.2 Representing simulations

Storing an entire state graph in a single OWL ontology doesn't really seem as a very practical solution. Therefore, a different approach is taken. The so called *superstate* file can be

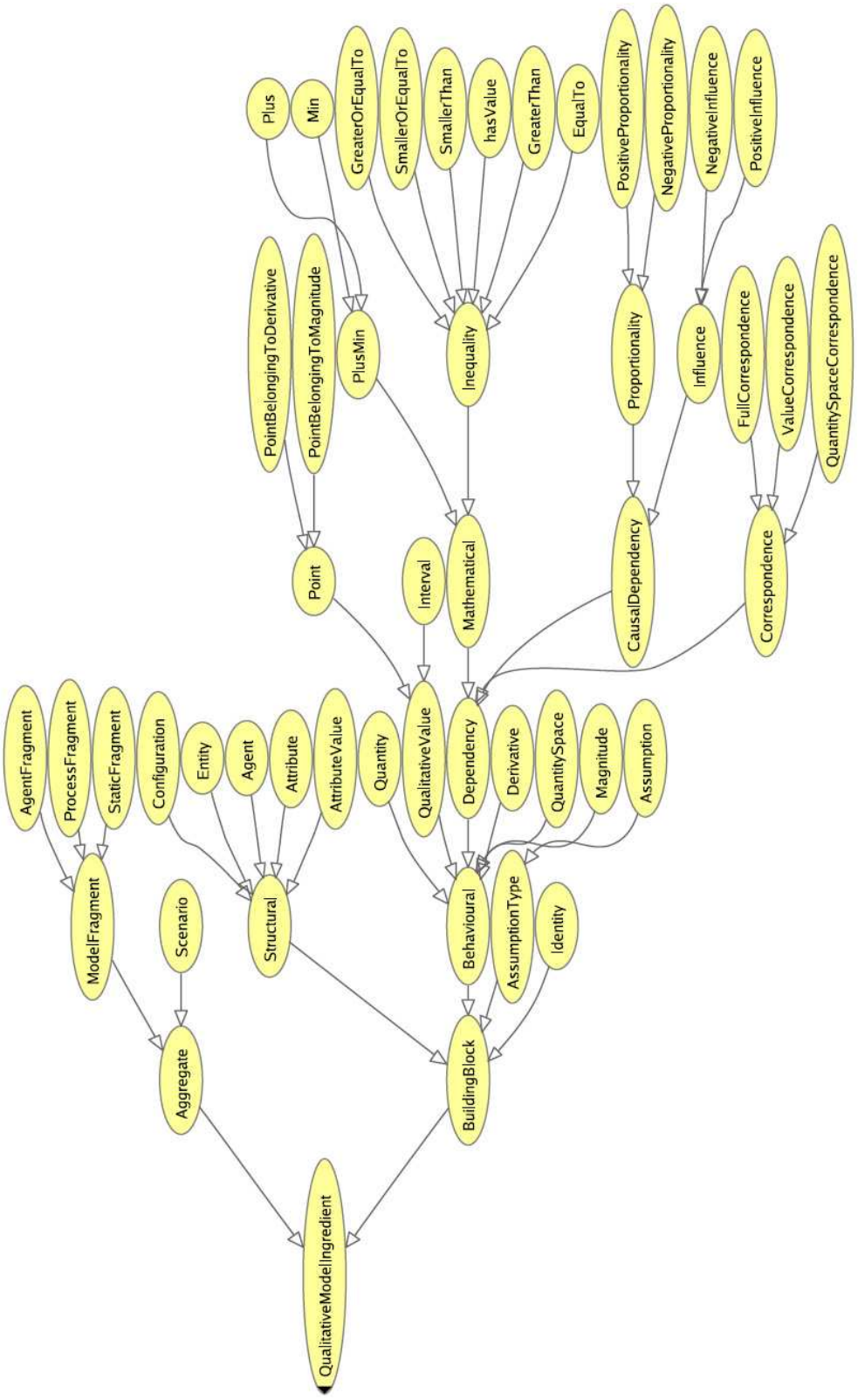


Figure A.1: The QR ingredient taxonomy defining the QR vocabulary (Liem and Bredeweg, 2006)

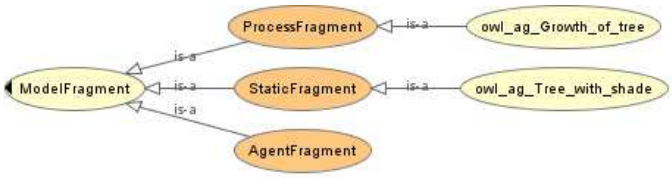


Figure A.2: Tree and shade MF hierarchy in OWL

observed as an intersection of all states in the simulation graph, not taking into consideration magnitudes and derivative assignments, but also (in the early stages of this research) excluding a few other important ingredients, such as model fragment information, correspondences, inequalities and calculations. Equation A.1 explains this idea in a more formal fashion, where s_n are individual states, and N is the total number of states.

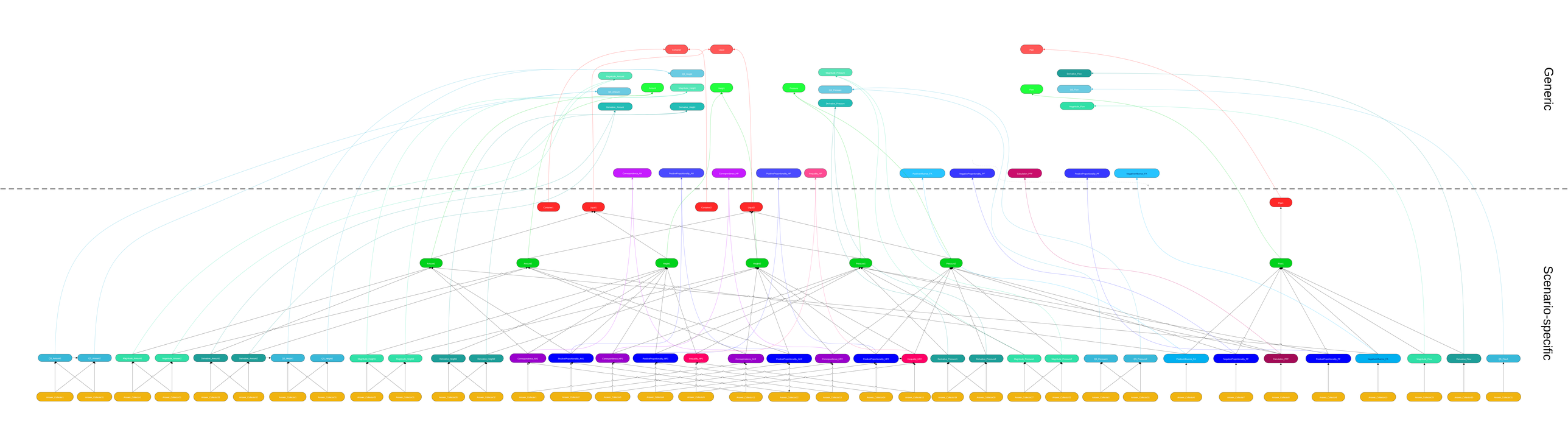
$$S = \bigcap_{n=1}^{N} s_n \tag{A.1}$$

The superstate file inherits the QR vocabulary class hierarchy and the state graph intersection is given in terms of a set of common individuals. It is worth noting that the original state of the superstate file is insufficient to support the learner model discussed in Chapter 5. Some improvements are given in Section 5.4.3.

Appendix B

Communicating Vessels: The Bayesian Network

The final structure of the *Communicating vessels* Bayesian network is given in the figure below. A detailed description of the structure can be found in Section 5.5.3.


Generic
Scenario-specific