A Learner Model based on a Bayesian Network for Garp3

Project in the Specialty Science Discipline,
Master's in Mathematics & Science Education

Miriam Brielmann

AMSTEL Institute
Faculty of Science
University of Amsterdam
Student Number: 0642924

Supervisor: Bert Bredeweg

Human Computer Studies Laboratory
Informatics Institute
Faculty of Science
University of Amsterdam

April 1, 2009

Contents

1 Introduction

Bayesian networks are tools for reasoning with uncertain knowledge (Darwiche, 2008). There are several examples in which Bayesian networks have been used for learner modeling (e.g. (Baker et al., 2008; Beck et al., 2008; Reye, 2004)). The goal of the project described in this report is to study the feasibility of using a learner model based on Bayesian networks for guiding the learner in the process of studying a model represented in the Garp3 software.

The Garp3 software provides a workbench for building, simulating and inspecting qualitative models. It facilitates the formalization of conceptual knowledge which is important for understanding the behavior of systems (Bredeweg et al., 2006). One application of Garp3 is in education, for learning about the behavior of systems. However, domain models and their simulations easily grow complex and become difficult to grasp. This is why some means for guiding a learner through the model is needed. Quags is a question generator for Garp3 (Goddijn, 2002; Goddijn et al., 2003; Bouwer, 2005) which generates questions about a simulation. The purpose of the new learner model is to focus the question generation on the parts that are important for the learner.

This document first summarizes the basic principles of Bayesian networks (2) and examples how Bayesian networks have been used for learner modeling (3). Then the main part of the project is reported (4), i.e. the development of a prototype of a learner model based on a Bayesian network for the Garp3 software for testing the applicability of this technique. This includes a representation of the learner's knowledge and an implementation for an example. Quags, the question generation module for Garp3 is controlled according to the state of the learner's knowledge recorded in the network. The usability of the approach is shown by a dialog in which the system guides the learner through all parts of the model until full knowledge (on behalf of the learner) is reached with the probabilities reflecting the increase of knowledge for the respective elements.

2 Bayesian Networks – an Overview

Bayesian networks are tools for reasoning with uncertain knowledge (Darwiche, 2008). A Bayesian network is a data structure which is used for representing dependencies between the probabilities of variables. It displays the structure of reasoning knowledge, in particular including independence relations, which help to reduce the complexity of the reasoning. Bayesian networks are also known as *belief networks*, *probabilistic networks*, *causal networks* and *knowledge maps*. For understanding Bayesian networks in more detail, we look at the underlying principles first (2.1), give a definition of Bayesian networks (2.2) and summarize key points in the design of Bayesian networks (2.3). This summary is based on Russell & Norvig (1995), unless otherwise noted.

2.1 Underlying Principles

Bayesian networks are an application of probability theory. In this section, we given an overview of the basic rules needed for the networks.

Unconditional vs. Conditional Probability There are two types of probability we deal with. The first one is the *unconditional* probability. It expresses with which probability we expect a proposition to be true without any further information, or, before we know anything else. This is why it is also called the *prior* probability. For example we denote $P(Cavity) = 0.1$ for saying that the probability that a dentist's client is having a cavity is 10%.

As soon as we have some evidence about a proposition, we talk about *conditional* or *posterior* probability. We then write $P(Cavity|Toothache) = 0.8$ (read: "the probability of Cavity given Toothache ...") for saying that if we know that a dentist's patient has toothache, there is a 80% probability that he has a cavity. The *product rule* is a way to relate conditional to unconditional probabilities.

$$P(A \wedge B) = P(A|B)P(B) \tag{1}$$

It can be explained: For A and B to be true, we need B to be true and then A has to be true given B. This can also be applied the other way around:

$$P(A \wedge B) = P(B|A)P(A) \tag{2}$$

The Joint Probability Distribution It is called an *atomic event* when all the variables contained in a model are assigned values. The *joint probability distribution* (or "joint") is the collection of the probabilities of all the atomic events in a model. (Simple) joint probability distributions can be displayed in a table in which all the probabilities sum to 1 (table 2.1).

	Toothache	¬Toothache
Cavity	0.04	0.06
¬Cavity	0.01	0.89

Tab. 1: A joint probability distribution (joint)

Taking the probabilities from the joint, we can apply the product rule to our example and compute the probability of a cavity given toothache:

$$P(Cavity|Toothache) = \frac{P(Cavity \wedge Toothache)}{P(Toothache)} = \frac{0.04}{0.05} = 0.8 \tag{3}$$

Bayes' Rule *Bayes' Rule* can be derived from the two forms of the product rule and writes:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{4}$$

The rule enables us to calculate the probability of a hypothesis given some evidence, if we only know the probability with which an evidence follows a cause (the hypothesis).

Take a patient having a stiff neck (S). We would like to know the probability of this patient having Meningitis (M). The doctor knows that meningitis is causing a stiff neck in 50% of the cases. (Currently) the probability of meningitis is 1/50,000 and the probability of a patient having a stiff neck is 1/20. The probability of the patient having meningitis given the evidence that he has a stiff neck is then:

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{1/2 * 1/50,000}{1/20} = 0.000001 \tag{5}$$

Combining Evidence The problem with Bayes' rule is that if we want to combine evidence we need to estimate many conditional probabilities. Say we know $P(Cavity|Toothache)$ and $P(Cavity|Catch)$, but want to know the probability of a cavity if both pieces of evidence are collected:

$$P(Cavity|Toothache \wedge Catch) = \frac{P(Toothache \wedge Catch|Cavity)P(Cavity)}{P(Toothache \wedge Catch)} \tag{6}$$

Now we should know the probability of the two symptoms occurring in combination given there is a cavity. If we look at n symptoms we would have to look at n^2 conditional probabilities for two symptoms at once, and end up with an exponential increase of the number if we look at more of them in combination.

A method to reduce this complexity is to identify the *conditional independence* of variables. A cavity is the *direct cause* of a dentist catching this cavity using the steel probe as well as it is the direct cause of the patient having toothache. As soon as we know that there is a cavity, we do not need to base the probability of the probe catch on the patient's toothache anymore and vice versa. The catch and the toothache are then conditionally independent:

$$P(Catch|Cavity \wedge Toothache) = P(Catch|Cavity) \tag{7}$$

$$P(Toothache|Cavity \wedge Catch) = P(Toothache|Cavity) \tag{8}$$

or to generally say that X and Y are independent given Z:

$$P(X|Y, Z) = P(X|Z) \tag{9}$$

2.2 Bayesian Networks

Coming back to the Bayesian networks, it becomes clear why it is important to structure the reasoning knowledge. By every conditional independence we identify, the complexity of the reasoning is reduced.

Bayesian networks are directed acyclic graphs (DAGs) that connect random variables (the nodes) by influences (the directed links). For each node there is a *conditional probability table* (CPT) that contains the probability values for the *conditioning cases*, i.e. the combinations of states of the parent nodes[1]. For root nodes the prior probability is given.

The Joint Probability Distribution A Bayesian network can be seen as a representation of the joint probability distribution. All the entries in the joint can be calculated from what is given in the network. An entry can be calculated by the product of the probabilities of the events given in the conditional probabilities of the network. We use the following formula, describing that the probability of an event (i.e. the conjunction of particular assignments to each variable) is the product of the probability of each node given its parents:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Parents(X_i)) \tag{10}$$

In figure 1 we see an example for a Bayesian network (Russell & Norvig, 1995). The owner of a house has installed an alarm which is activated if there is a burglary or sometimes also if there is an earthquake. Two neighbors agreed on calling the house owner if they hear the alarm. However, for both of them there is a chance that they do not call if there is an alarm, and also a chance that they call if there is no alarm.

Now the owner can calculate the probability of an alarm with neither a burglary or an earthquake and both of the neighbors calling:

$$\begin{aligned} P(J \wedge M \wedge A \wedge \neg B \wedge \neg E) &= P(J|A)P(M|A)P(A|\neg B \wedge \neg E)P(\neg B)P(\neg E) && (11)\\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 && (12)\\ &= 0.000628 && (13) \end{aligned}$$

Dynamic Bayesian networks In a *dynamic Bayesian network* (DBN) the dimension of time is added to the Bayesian network. The nodes are partitioned into slices in different points of time. The structure of the network stays the same in the slices[2]. Because of that, the structure is recurrent and usually specified using two slices for t and t+1. Two simples example are shown in figures 2 and 3 (Darwiche, 2008).

2.3 Designing Bayesian Networks

For defining a Bayesian network, a definition of the structure and probabilities is needed, and an appropriate algorithm needs to be chosen.

[1] In the CPTs the entries of the rows sum to 1, so often one column of the table is left out because the value can be calculated using the given ones.

[2] Except in the first one possibly

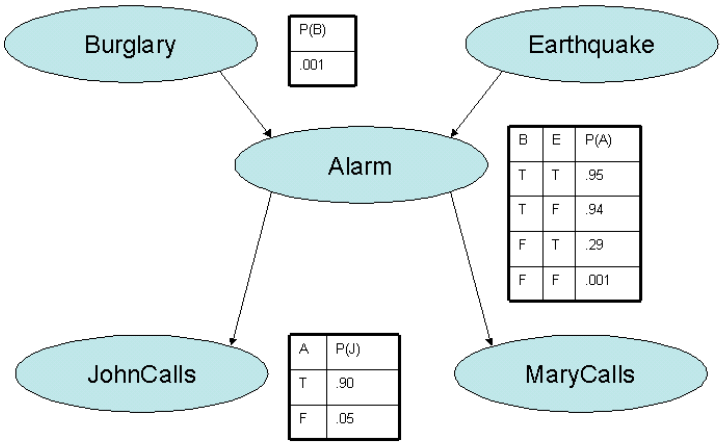

Fig. 1: A Bayesian network

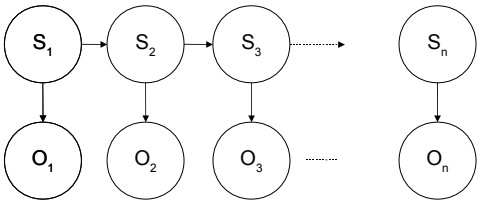

Fig. 2: A simple dynamic Bayesian network, repeating the structure.

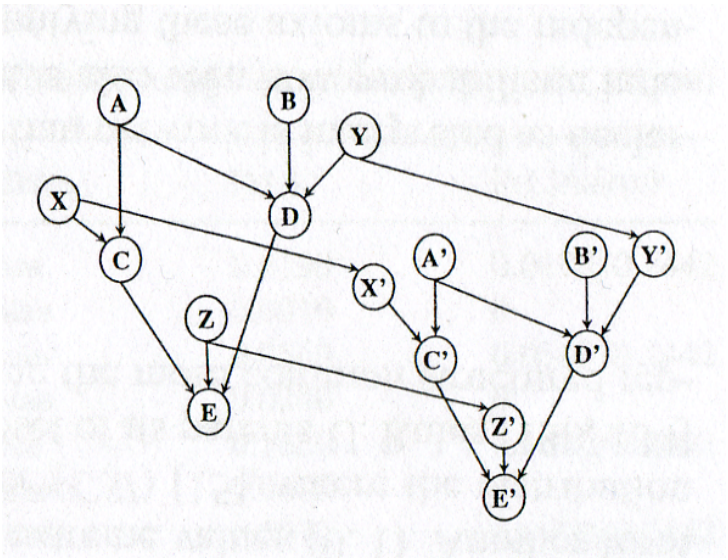

Fig. 3: Another simple dynamic Bayesian network, specified by two time slices.

Structure The structure of the network can be defined by an expert or produced by machine-learning. The structure of the network is important, because the structure determines the complexity of updating the probabilities for a given piece of evidence. Identifying independent sections of the network reduces the number of probabilities that have to be set for defining the joint probability distribution. Also, choosing an appropriate order for the nodes helps in coming up with the conditional probabilities. A causal model is recommended (Russell & Norvig, 1995).

Probabilities The (conditional) probabilities, i.e. the entries of the probability tables, can be estimated by an expert or machine-learned based on collected data (Charniak, 1991).

Algorithm for Computation The algorithms for computing the conditional probabilities given an evidence can be divided into two categories:

1. *Exact Solutions* are NP-hard. This method is usually only used for singly connected networks (i.e. "the underlying directed graph has no more than one path between any two nodes"). There are ways of turning a multiply connected network into a singly connected one (e.g. clustering). However this can lead to an "explosion of values" due to the combinations of values of nodes that have to be considered.

2. *Approximate Solutions* There are different methods for finding approximations of the conditional probabilities in the network. One of them is stating the initial values of some nodes and using them for deciding on the states of the other nodes. This is done for a certain number of times and the statistics are used to determine the needed probabilities. Some of the methods do not take evidence into account, some do.

Which algorithm suits best depends on the nature of the network. (Charniak, 1991)

3 Bayesian Networks for Learner Modeling

This section discusses examples of Bayesian networks for learner modeling. "Knowledge Tracing" as used in the ACT programming tutor of Corbett & Anderson (1995) is an approach to student modeling that uses Bayesian inference for tracking the learners' knowledge (3.1). It does not explicitly discuss the use of Bayesian networks but lays the foundation for later work that does so. Reye (2004) gives a detailed description of the theory underlying the design of a learner model based on a Bayesian Network (3.2). Baker, Corbett, & Aleven (2008) discuss the topic of estimating the slip and guess probability in a learner model based on Bayesian networks (3.3); Beck et al. (2008) use such a learner model for evaluation whether help really helps (3.4). Finally, we give a summary of the design issues found in the literature (3.5).

3.1 Knowledge Tracing

Corbett & Anderson (1995) discuss an approach for student modeling that they used in the ACT Programming Tutor (APT). The APT was used in introductory programming courses at high school and university level. It is an environment that allows students to practice writing short programs in Lisp, Prolog or Pascal. The student is presented a unit as a piece of text. This is followed by a number of exercises that are given to the student until the system believes that the student has reached mastery in all the skills that were introduced in the section. The user interface offers the student a list of templates that can be selected for use in the program and completed by typing in the identifiers and constants.

Every step the student does in the editor is compared to the *ideal student model*. The ideal student model is a set of several hundred language-specific rules for writing programs that form a "complete, executable model of procedural knowledge of the domain". Immediate feedback depending on the result of the comparison is given. If the student's action matches an applicable rule in the ideal student model, the action is accepted and the internal problem representation as well as the window are updated. If the action does not match an applicable rule, it is not accepted and a hint to the student is shown in the window. Because of that, "the student always stays on a recognizable solution path". In addition, the tutor shows the "skill meter" with bars that indicate the student's knowledge state, displaying the probability that the student has acquired the skills of that section. Check marks are used to signalize that the student has reached mastery in a skill.

Knowledge Tracing and Mastery Learning Knowledge tracing was introduced in order to support the students' mastery learning. A two-state model is used: Each rule is in the state of being "learned" or "unlearned" (Corbett & Anderson, 1995, i.e. known or not known). The transition from "learned" to "unlearned" can be done by reading a text or practice. There is no forgetting. There is a chance of guesses and slips. The four initial parameters used for knowledge tracing are:

1. G = the Guess parameter, i.e. the probability with which the student gives a correct response when she actually does not know it,
2. S = the Slip parameter, i.e. the probability with which the student gives a wrong response when she actually does know the correct one,
3. L_0 = the probability of the student knowing each skill in the beginning of using a tutor,
4. T = the probability of learning this skill, regardless whether the answer is correct, at each opportunity to practice a skill the student does not know.

The system's belief that the student knows a rule is calculated after an action (n) using the action as evidence for the posterior probability:

$$p(L_n|evidence) = p(L_{n-1}|evidence) + ((1 - p(L_{n-1}|evidence)) * P(T)) \qquad (14)$$

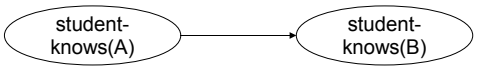

Fig. 4: Modeling the relationship between two topics

The student is given exercises for practicing a skill/acquiring a rule until a certain probability of mastery is reached. In the tutor the "mastery criterion" for a rule is a probability of 0.95.

3.2 Belief Networks for Modeling Student Knowledge

Reye (2004) discusses an approach of using belief networks for modeling student knowledge, taking into account that students should be considered as unreliable source of information and therefore dealing with uncertainty.

Structuring Knowledge In Reye's model, *relationships between topics* are formulated using conditional dependencies in order to be able to express how closely the topics are related, including both strict prerequisite relationships and weaker connections. Examples of relationships between the topics given in figure 4 are:

- knowing A is prerequisite of knowing B

$$p(\textit{student-knows}(A)|\textit{student-knows}(B)) = 1 \tag{15}$$

- if A and B are closely related (a student who knows A most likely also knows B)

$$p(\textit{student-knows}(B)|\textit{student-knows}(A)) = 0.95 \tag{16}$$

- the knowledge of A does not affect the knowledge of B and the prior probability of B is $p(B) = 0.01$ then

$$p(\textit{student-knows}(B)|\textit{student-knows}(A)) = p(B) = 0.01 \tag{17}$$

More than two topics can be connected. A belief (=Bayesian) network is a way to design and visualize the model.

For making the design easier, the direction of the arcs is chosen to be the same as the order in which the topics should be learned. The causal relationships (causal in the sense that not knowing A will cause not knowing B) also make understanding the belief network easier.

The belief network *supports gathering of information about the student's state of knowledge*. There is an analogy of the prerequisite structure and a digital circuit built of AND gates with inputs for the prerequisite topics and one output: each gate outputs 1 only if all the inputs are 1 (all the prerequisite topics are known) and it is functioning

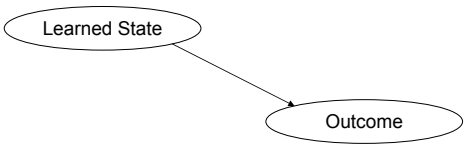

Fig. 5: The student's response as unreliable source of information

properly (the topic itself is known). If the student does not know one prerequisite topic, he will not know any of the topics building on that. Analogously the non-functioning of one gate disables all the following gates. Because of this analogy, for student modeling the GDE (General Diagnostic Engine; De Kleer & Williams, 1987) approach can be used for minimizing the number of measurements for finding a fault. The measurements are selected maximizing the expected amount of information gained by that measurement by minimizing the the entropy of the belief network after gathering that information.

The detection of the fault in a student's knowledge is less complex than in an electrical circuit. In the student model it is expected that if a prerequisite topic is not known all the following topics are not known, so the number of combinations of possible states of knowledge is less than in an electrical circuit (in a chain of items the growth of possible combinations is linear instead of exponential).

Students are unreliable sources of information because their responses do not always reflect their actual knowledge directly. There is the chance of lucky guesses and slips. Because of that evidence is collected from the student's responses (outcome) to compute the probability of him really knowing the topic (the learned state, see figure 5). The following probabilities are used to calculate the probability that the student has learned a topic:

1. $p(outcome = correct|learned)$ is the probability of a correct outcome if the student knows the topic

2. $p(outcome = correct|\neg learned)$ is the probability of a lucky guess, i.e. the outcome is correct, although the student has not learned it yet,

3. $p(learned)$ is the prior probability that the student knows the topic that is estimated before any information is collected.

Reye combines nodes of three types:

1. the "student-knows(topic)" nodes are linked in the *belief net backbone*, according to their prerequisite relationships,

2. each student-knows(topic) node is connected to supporting *local nodes* that are used for making inferences about the learned state from the outcome (e.g. "when-opportune-student-demonstrates-usage-of(topic)), comprising a *topic cluster*,

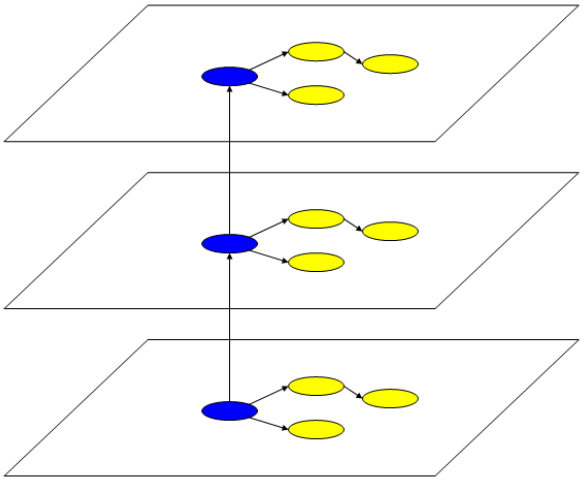

Fig. 6: A belief net backbone

3. the "global nodes" are keeping track of the student's overall characteristics, e.g. "student-is-bored()" and "student-overall-aptitude", and are used for fine-tuning the topic clusters.

Figure 6 shows the student-knows(topic) nodes (blue) and the local nodes (yellow) connected by the belief net backbone.

Updating Knowledge The student model is updated over time in steps according to the interactions with the system. For every interaction the update happens in *two phases* which are visualized in figure 7:

1. In phase one the (estimated) state of the student's knowledge before the interaction is updated,

2. In phase two the expected state of the student's knowledge after the interaction is calculated.

In phase one, i.e. for updating the system's belief in the student's knowledge before the interaction, the following parameters are used:

1. O_n a possible outcome of interaction n, i.e. a student's response like "correct" or "incorrect" or different levels of help,

2. L_{n-1} the system's belief that the student knows the topic prior to the n'th interaction,

3. $p(O_n|L_{n-1})$ the system's belief that outcome O_n will occur when the student already knows the topic (which is the slip probability if O_n is an incorrect outcome),

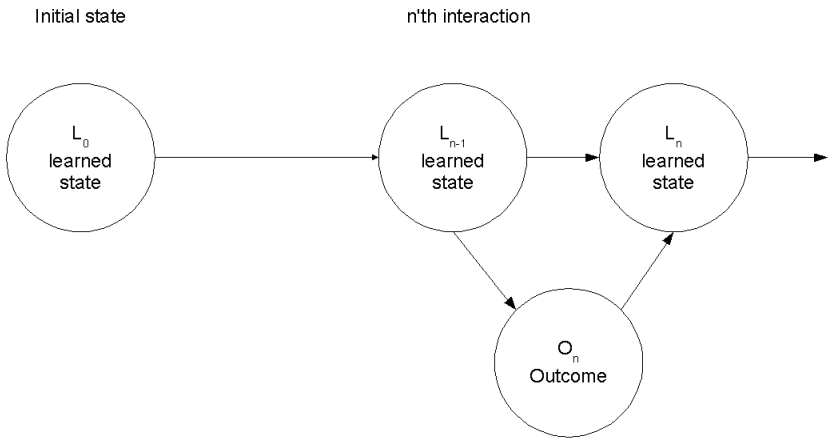

Fig. 7: Updating the student model in two phases

4. $p(O_n|\neg L_{n-1})$ the system's belief that outcome O_n will occur when the student does not know the topic yet (which is the guess probability if O_n is a correct outcome).

On the occurrence of outcome O_n the system's belief in the student's previous state of knowledge $p(L_{n-1}|O_n)$ is revised using Bayes' rule:

$$p(L_{n-1}|O_n) = \frac{p(O_n|L_{n-1})p(L_{n-1})}{p(O_n|L_{n-1})p(L_{n-1}) + p(O_n|\neg L_{n-1})p(\neg L_{n-1})} \tag{18}$$

This means that the system's belief that the student did already know a topic before the interaction given outcome of the interaction O_n, is the probability of the student knowing the topic and giving outcome O_n over the overall probability of outcome O_n.

For phase two, i.e. for calculating the expected state of knowledge after the interaction (the expected changes in the student's knowledge due to tutoring), the following two conditional probabilities are relevant. Suppose that O_n is correct. Then:

1. $p(L_n|L_{n-1}, O_n)$ is the rate of remembering/not forgetting, i.e. the student remembers (L_n) what was known previously (L_{n-1}), given outcome O_n of interaction n and

2. $p(L_n|\neg L_{n-1}, O_n)$ is the rate of learning given the outcome (learning from the interaction).

The belief about the student's knowledge after the interaction is computed by adding the probability of the student knowing something before the interaction and remembering it, and the probability of the student not knowing it before and learning it by the interaction.

$$p(L_n|O_n) = p(L_n|L_{n-1}, O_n)p(L_{n-1}|O_n) + p(L_n|\neg L_{n-1}, O_n)p(\neg L_{n-1}|O_n) \tag{19}$$

Reyes names as the essential parameters that have to be selected for each skill:

- $\gamma = \frac{p(O_n|L_{n-1})}{p(O_n|\neg L_{n-1})}$ which is the likelihood ratio of an outcome given that the student knows and of the outcome given the student does not know
- $\rho = p(L_n|L_{n-1}, O_n)$ which is the rate of remembering/not forgetting and
- $\lambda = p(L_n|\neg L_{n-1}, O_n)$ which is the rate of learning.

Depending on them, the system's belief of the student's learned state evolves.

Applicability Reye compares his approach to Corbett and Anderson's Knowledge Tracing (3.1). Corbett and Anderson's model also uses two states for representing the student's knowledge. Their equations for updating the model after a student's action can be derived from Reyes's approach if the assumptions of Corbett and Anderson are considered and parameters are chosen accordingly.

Moreover, Reye shows how his concept of using a belief network is suitable to Shute's SMART student modeling approach. Shute had developed functions for updating the student model by finding the best-fitting curves to points that were given by experts' opinions. Estimates of the final knowledge of students given by these functions were close to the results of post-tests of the students' knowledge and hence quite successful. The approach was used for problem solving in Shute's SMART system where the student can choose from different levels of help. Reye associates the level of help with the outcome and shows that if the parameters are chosen appropriately, his approach of belief nets produces nearly the same curves as Shute's approach.

Finally, Reye is discussing some issues on the computational complexity of his approach to student modeling. Available algorithms for singly connected networks cannot be used because the network is more complex: the prerequisite relationships can lead to more than one path between two topics and the topic clusters will be connected by the global nodes. Reye introduces the term *barren nodes* for nodes that no information is available for yet and because of which no inferences about the descendant nodes can be made yet. The barren nodes and the descendant parts of the network can be left out to aid computational efficiency.

3.3 Contextual Estimation of Slip and Guess Probability

In the work of Baker, Corbett, & Aleven (2008), Bayesian networks are used for modeling students' knowledge in an Intelligent Tutoring System (ITS). Based on the model, the student is given tasks for practicing skills until mastery is reached. The authors' work is based on Corbett and Anderson's *Bayesian Knowledge Tracing model* which constantly assesses the student's knowledge for every skill based on behavior (3.1). The four parameters used for the Knowledge Tracing are fit for each skill using data from students applying that skill within an intelligent tutor in order to find the combination that best predicts the pattern of correct and incorrect answers.

Problems with the Bayesian Knowledge Tracing Model The article at hand is addressing two statistical problems which the Knowledge Tracing models have been vulnerable to:

- 'Identifiability' = different combinations of the four parameters can fit the given data equally well, but produce different estimates of the students knowledge. Each combination leads to different assignments of exercises to the student which can result in under- or over-practice.
- 'Model degeneracy' = a model is theoretically degenerate when the Guess or Slip parameter is greater than 0.5. That means that the probability of a correct answer is higher if the student does not know the skill than if he does, or the probability of an incorrect answer is higher if the student knows the skill than if he does not, respectively. A model is empirically degenerate when after a number N of correct answers the belief that the student knows the skill has decreased (first test) and when after a number M of correct answers the student is not estimated to have reached mastery (second test).

Previous ways of fitting the parameters in the Bayesian Network to the skills in the tutoring system:

- The *baseline approach*, allowing the parameters taking any value between 0 and 1; the authors use the Bayes Toolkit-Student Modeling (BNT-SM) for it.
- The *bounded guess and slip approach*, bounding the guess and slip parameter to be in a certain range and making model degeneracy impossible; the authors use Microsoft Excel for it.
- The *Dirichlet Priors* approach, using a Gaussian probability distribution for parameter values across skills and constraining the values for the parameters for all skills and thus biasing the values of the single parameters to values that fit the whole dataset; the authors use the BNT-SM for it.

Contextual Estimation of Guess and Slip The *contextual estimation of guess and slip* is the authors' new approach to find the guess and slip parameters. The difference to the previous approaches is that the values of the parameters are not held constant for all situations but estimated for individual actions, i.e. related to the context.

- Log files from previous student interactions are used to estimate whether an answer was a guess or a slip, including subsequent steps.
- Machine-learning is used to identify features of an action that are independent from subsequent actions and characterize the guess/slip probability. They can be used for evaluating the guess/slip probability for an action immediately.
- $P(T)$ and $P(L_0)$ are fit for each skill, using curve-fitting.

Evaluation of the New Approach The approach is evaluated using the student log data from the ITS. The outcome of the evaluation is that using the contextual guess and slip model there are still cases of degeneracy, but the model is "substantially less degenerate" than the models used before. In terms of accuracy (comparison of the model's prediction with the answer of the student), the new model scores best as well.

3.4 Does Help Help?

The goal of Beck, Chang, Mostow, & Corbett (2008) is to measure the effectiveness of help in an ITS. Because a pre- and post-test experiment is often "impractical" and asking the students about their opinion only provides qualitative feedback, they seek for a way of deriving the help's influence from observational data.

Comparison to Other Approaches Before the new approach is introduced, two other approaches that could be applied to the data are discussed:

1. *Experimental Trials*, comparing two outcomes. It is difficult to decide what to compare. In the authors' example in a reading tutor, the accuracy of the reading of words for which a student asked for help and the accuracy of the reading of words which he did not ask for help are compared. The immediate effect of the help is positive, the student reads the words help was given for more accurately. If only the outcomes of the words that the student encounters a day later are compared in order to avoid memory effects, it shows that the student performs worse on words that help was given for.

2. *Learning Decomposition* "is a variant of learning curves". In this approach the relative value of different types of learning opportunities are estimated. It does not require the comparison of two data sets, but "computes the impact of help compared to reading the word", that is the value of help compared to one opportunity to practice. Examining the data with this approach leads to the result that receiving help actually causes students to perform worse.

For both of the approaches, the authors suspect that the apparent negative outcome which implies the help is rather inhibiting the students' learning rather than supporting it, results from the students asking for help when they are lacking knowledge. So the act of asking for help is not cause but evidence of a lack of knowledge.

New Approach: Bayesian Evaluation and Assessment For the new approach, the Bayesian Evaluation and Assessment, there are two analysis goals. Firstly, the student's knowledge should be assessed. This assessment of student knowledge is based on knowledge tracing (3.1). Secondly, as for the others, the effect of help is to be evaluated. Both of them are integrated in one model as displayed in figure 8. H_n is a binary variable that signals whether the help intervention was used or not. The model integrates the influence

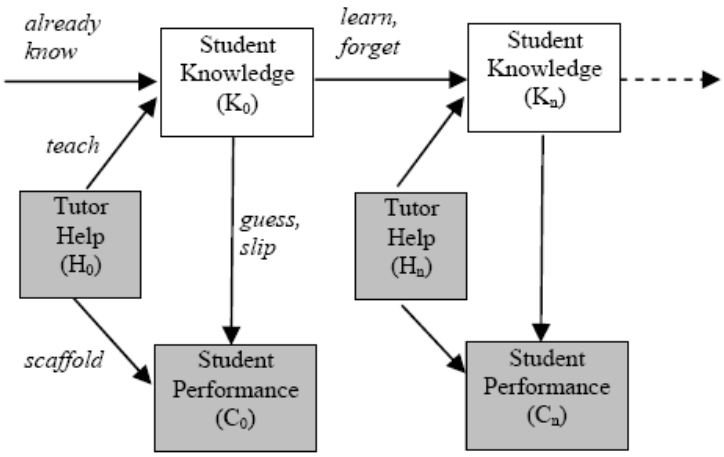

Fig. 8: Bayesian Evaluation and Assessment architecture

$$already\ know \mid help \equiv \Pr(K_0 = true, \mid H_0 = true)$$
$$already\ know \mid no\ help \equiv \Pr(K_0 = true, \mid H_0 = false)$$

$$learn \mid help \equiv \Pr(K_n = true \mid K_{n-1} = false, H_n = true)$$
$$learn \mid no\ help \equiv \Pr(K_n = true \mid K_{n-1} = false, H_n = false)$$

$$forget \mid help \equiv \Pr(K_n = false \mid K_{n-1} = true, H_n = true)$$
$$forget \mid no\ help \equiv \Pr(K_n = false \mid K_{n-1} = true, H_n = false)$$

$$guess \mid help \equiv \Pr(C_n = true \mid K_n = false, H_n = true)$$
$$guess \mid no\ help \equiv \Pr(C_n = true \mid K_n = false, H_n = false)$$

$$slip \mid help \equiv \Pr(C_n = false \mid K_n = true, H_n = true)$$
$$slip \mid no\ help \equiv \Pr(C_n = false \mid K_n = true, H_n = false)$$

Fig. 9: Equations for parameters in Bayesian Evaluation and Assessment model

of the occurrence of help both on the (current) student performance (C_n) and on the student's (long-term) knowledge (K_n). The effect of the help intervention can be calculated from the values that are learned by the model (see figure 9). E.g. the probability of a student learning a skill with help ($learn|help \equiv Pr(K_n = true|K_{n-1} = false, H_n = true)$) can be compared to the probability that the student will acquire the skill by merely practicing it ($learn|nohelp \equiv Pr(K_n = true|K_{n-1} = false, H_n = false)$).

Application The Bayesian Evaluation and Assessment approach was tested on data from a reading tutor. This tutor used speech recognition to determine the correctness of the student's reading word by word and accordingly set C_n, the student's performance. The students could ask for help for challenging words.

For the student modeling, a generic Bayes net toolkit (BNT-SM) was used. It takes as an input a data set and an XML specification of a student model and trains and tests the model.

	KT model	Help model	
		No Help Given	Help Given
Already know	0.618	0.660	0.278
Learn	0.077	0.083	0.088
Guess	0.689	0.655	0.944
Slip	0.056	0.058	0.009

Fig. 10: Parameters estimated by the models

Results Figure 10 shows the learned parameters for the Bayesian Evaluation and Assessment (Help) model that is considering the effects of help compared to a simpler knowledge tracing (KT) model.

The results are interpreted as follows:

- The probability of *already know* is higher if there is no help, so the tutor help is more likely to be asked if the student does not know the word yet.
- Learning is higher if there is help than if there is no help.
- The probability of a guess is higher if there is help; that means the student will more likely give the right response even if it is not really known yet; the help is providing scaffolding to the student's immediate performance; even if long-term knowledge is the higher goal, the positive effect of helping the student "to become unstuck" should not be ignored.

In summary, the model indicates a positive impact of help on students' learning. However, the new model was not more precise than the simpler knowledge tracing model in modeling student knowledge. This was tested on the training data, comparing the predictions of the models whether the student would read the next word correctly.

3.5 Design Issues

In the articles discussed, the following points were made about the design of the student model.

Structure An important relationship to be modeled is the causal relationships between behavior and knowledge. For that, a two-node Bayesian network can be used for modeling the student's knowledge of each skill, like in Corbett & Anderson's knowledge tracing (3.1). Other features that can be modeled in a learner model based on a Bayesian network are the influence of a tutor intervention and the dependencies between topics (the backbone). In the given examples the structure was pre-defined by the researcher.

Probabilities The probabilities are determined by machine learning using logs of previous uses of the ITS.

Algorithms for updating Which algorithms are used for updating the probabilities was not described in the given articles. However, Reye (2004) discusses an approach how to reduce the complexity of computations by not taking into account nodes for which there is no information, and their descendants (barren nodes).

4 A Learner Model for Garp3

The goal of this project is to develop a learner model that can serve as basis for "discussing" a model in Garp3 with a learner. During the dialog, elements the learner does not know should be emphasized in order to facilitate learning, and the learner should not be bored by questions about parts that are known. The dialog should run until confidence is reached that the learner has understood all parts of the model. For evaluating the approach, the learner model is coupled with the Quags question generator for Garp3 (Goddijn, 2002) and an example discourse is analyzed whether it reaches the goals set.

As basis for showing the integration of the learner model, we first discuss the embedding of the learner model in a tutoring system (4.1). Then, we give an overview of the "tree and shade" model in Garp3 which serves as running example for the subject matter input (4.2) and "Quags", the question generator for Garp3 (section 4.3). The design of the learner model based on a Bayesian network (BN) is described in section 4.4. Finally, the experimental setup for the evaluation (4.5) and the generated discourse (4.5) are explained.

4.1 Architecture

Figure 11 shows the the architecture of a tutoring system as proposed by de Koning et al. (2000) and discussed in Brielmann (2008). The red box highlights the parts that are particularly important in this project. The components can be grouped according to their responsibilities.

The Knowledge Base components are responsible for storing and generating knowledge. The *generic qualitative knowledge* is a collection of facts and rules in a form suitable as input for the qualitative simulator. The *qualitative simulation* produces the input for the *subject matter model* which is a collection of articulate simulation models that are prepared for the presentation to the learner.

Curriculum Generation is mainly handled by the *subject matter sequencing* module which generates the curriculum by putting the topics given in the subject matter model into a sensible sequence. The module is supplemented by the *question/assignment generation*, *prediction exercise generation* and *explanation generation* modules that convert the target topics given by the *subject matter sequencing* into the mode that is appropriate for the current learner interaction (question/assignment, explanation or exercise).

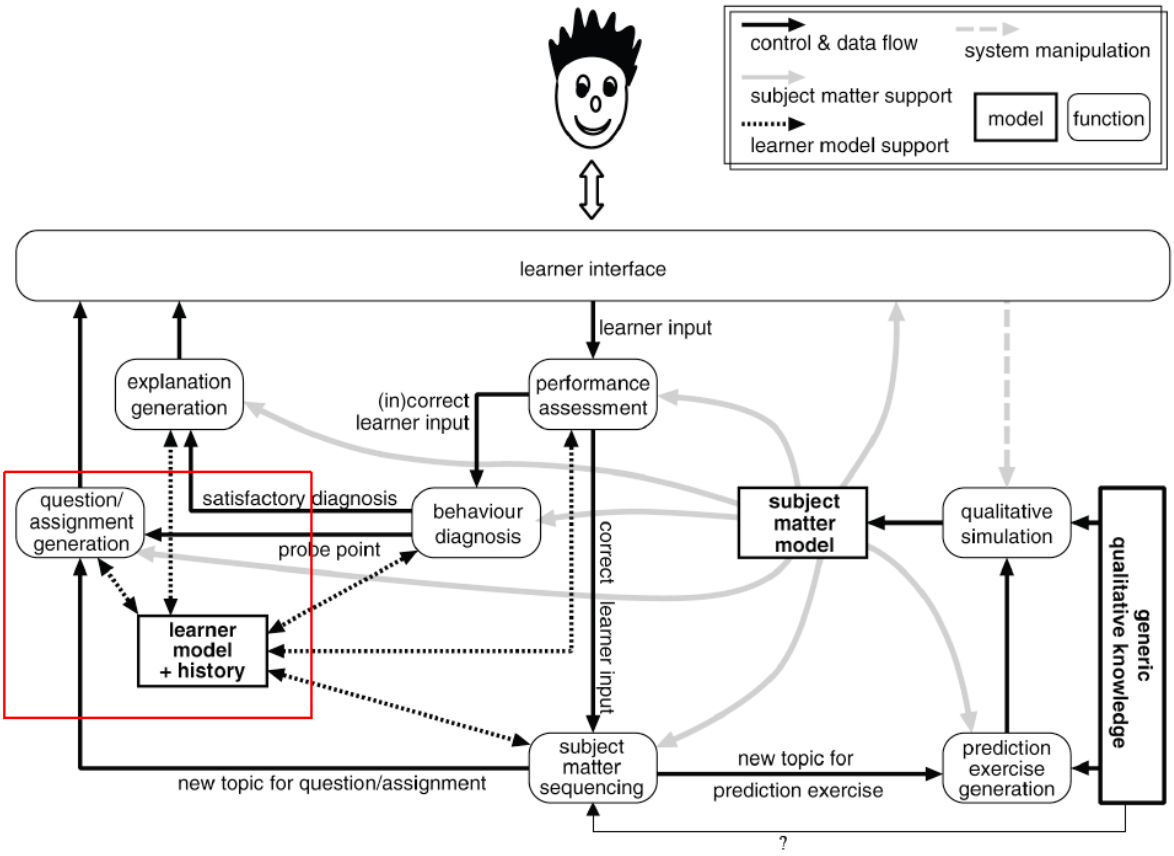

Fig. 11: The learner model embedded in the system

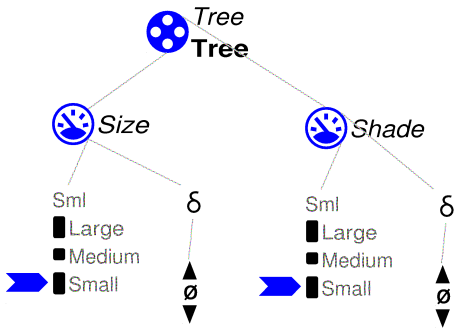

Fig. 12: Scenario "a small tree"

Learner Interaction is most obvious in the *learner interface*. Its responsibility is to pack the questions, explanations and exercises into sensible forms of communications and presentations for the learner. Other parts are the *performance assessment* module and the closely related *behaviour diagnosis* module. All relevant information about the learner, e.g. previously discussed topics and presentation preferences, are saved in the *learner model + history*.

In the system used for this project, the knowledge base is implemented in Garp3. We focus on the processing of one topic. Subject matter sequencing is not implemented. It is assumed that an appropriate topic is given as input which is then processed by Quags (question/assignment generation) together with the input from the learner model. In this project, we implement the learner model, using a Bayesian network.

4.2 The Tree and Shade Model in Garp3

As subject matter for our student model, we use the "tree and shade" model (Bredeweg et al., 2006). It is a simple example of a Garp3 model which models the growth of a tree and its shade and contains all ingredients of a typical model[3].

The starting scenario for the simulation describes a small tree with a small shade using the entity tree and the quantities size and shade (figure 12).

The static model fragment "tree with shade" (figure 13) describes the relation between the size of the tree and the shade. If the size is increasing, the shade is increasing as well (positive proportionality between size and shade). Size and shade have the same magnitudes during the simulation (quantity space correspondence between the quantity spaces of size and shade).

The process model fragment growth of the tree (figure 14) models the growth of the tree using the quantity growth rate, i.e. the rate at which the tree is growing. If there is growth, the size of the tree increases proportionally (positive influence of growth rate on size).

[3] We simplified the model by leaving out one proportionality described in the article because it is not needed for demonstration purposes.

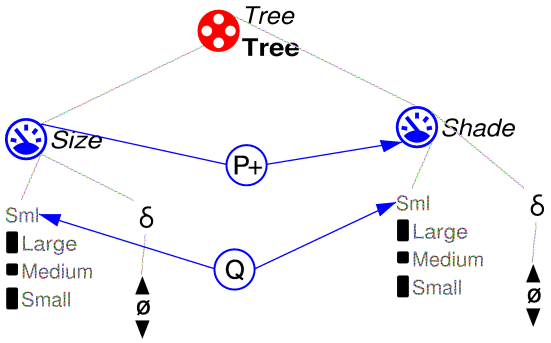

Fig. 13: Static model fragment "tree with shade"

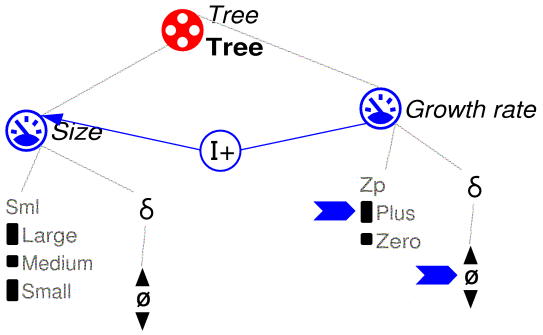

Fig. 14: Process model fragment "growth of tree"

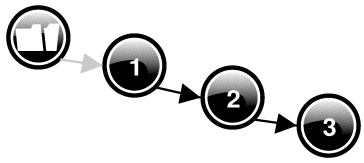

Fig. 15: Simulation with starting scenario "small tree"

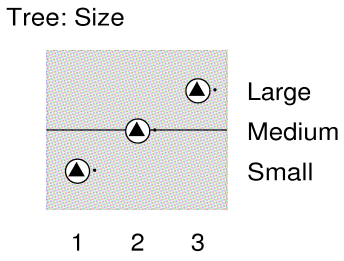

Fig. 16: Value history of size

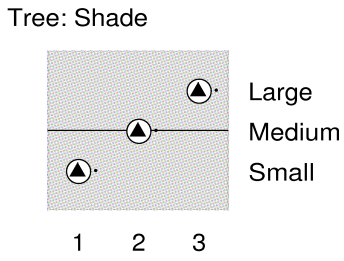

Fig. 17: Value history of shade

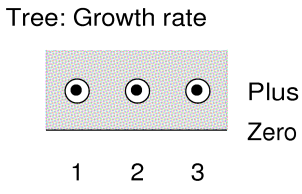

Fig. 18: Value history of growth rate

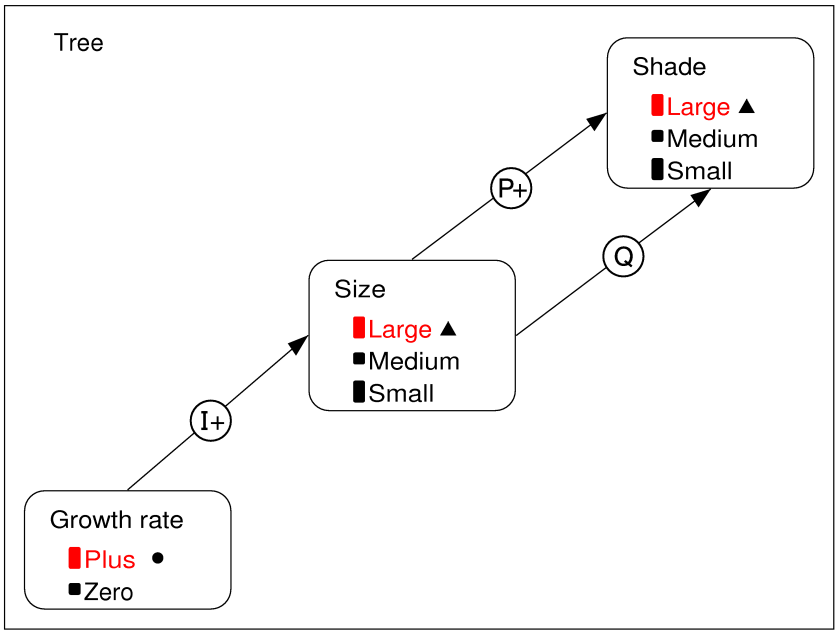

Fig. 19: Dependencies in all states

The simulation produces a graph with three states (figure 15). The tree is growing from small size in state 1 via medium size in state 2 to large size in state 3 (figure 16). The same happens for the shade (figure 17). The growth rate is constant (figure 18).

Because the model fragments are applicable in all states, the causal model stays the same during the simulation. It is an aggregation of the information from the model fragments (figure 19).

4.3 Question Generation

For testing the student model, "Quags" (Goddijn et al. 2003; Goddijn 2002; Bouwer 2005), the question generator for Garp3, is used. For showing how the student is guided through the parts of the model, the key point for the question generation is how to set the scope.

Quags uses a number of methods to reduce the total set of possible questions about a model to a set of sensible questions for the student. The approach is to first restrict the number of questions by certain criteria which can be set (e.g. by the user, the curriculum designer or another module), and then automatically select the most interesting ones according to built in heuristics.

The restriction methods that are used by QUAGS are:

1. Question type criteria

2. Selection of states

3. System scope (can be entities and quantities)
4. Included subject quantities
5. Forbidden subject quantities

The first restriction method is working on a "global" level, using a taxonomy for selecting the question templates for the admitted question types:

1. Perspective (changing values vs. reasons of change)
2. Concept (concept used, e.g. values, causal relations, inequality, correspondence, ...)
3. Behavior (real/all/submissive)
4. Infostate (info available in present state or not)
5. Answer method (strategy to find the answer, where is the information to be found)

The learner model contains information about combinations of quantities and concepts (e.g. the value of shade). The two possible options for setting the scope of the question generation are (1) to exclude elements of the model as soon as they are known and (2) to focus on the elements that are not known. We choose approach 2, in the first place because Quags does not offer the feature of excluding an element at the level of detail that is desired for the student model, while it does offer the possibility of including particular elements. This is needed for demonstrating the functionality of the learner model by selecting the elements with the lowest probability of knowledge. Moreover, focusing on these particular elements allows addressing the weak points of the learner specifically.

4.4 The Learner Model

Our learner model represents each primitive in the Garp3 model as a node in a Bayesian network (figure 20). The pieces of knowledge that are tracked in interaction with the learner are the magnitudes, derivatives, dependencies and correspondences. In the network, these "knowledge nodes" are the root nodes. From them, the knowledge about the quantities is derived which again can be used to derive the knowledge about the entities. An alternative approach for the tracking of knowledge would be to use the quantities as main pieces of knowledge and to derive the state of knowledge about the magnitudes, derivatives and quantities from them. However, the current approach is chosen over this, because we consider it as an advantage that the direction of dependency corresponds with the desired way of reasoning about the learner's knowledge in the form of "if the learner knows all the elements connected to a quantity, he knows the quantity". Only tracking the knowledge about quantities would not offer enough detail.

The observations from the interactions with the learner are connected as extra nodes to the knowledge nodes. This makes it possible to include the chance that the learner's answer is a guess or slip (also see the discussion of Reye, 2004 in 3.2). Hence, each "knowledge node" has an observation as a child node.

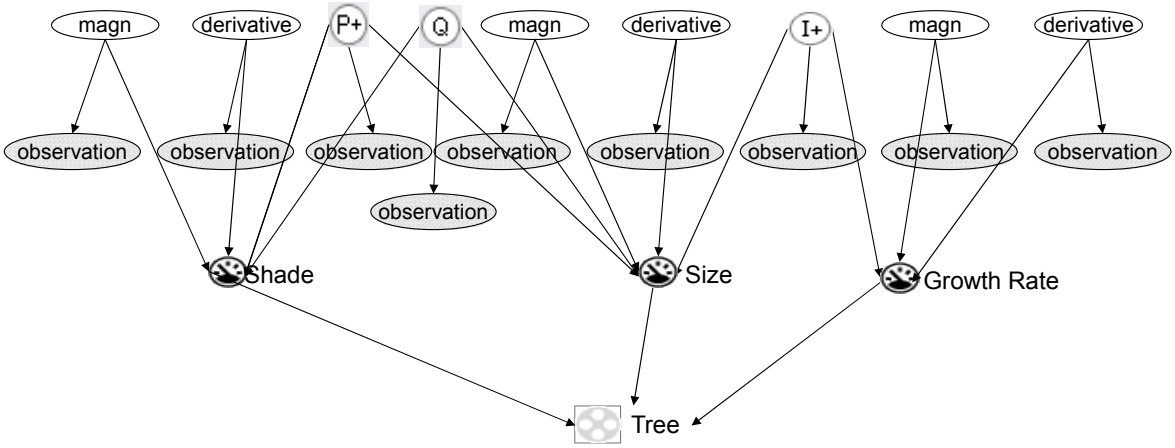

Fig. 20: A learner model network for a small growing tree, 1 time slice/interaction

Finally, the learners knowledge changes over time. This is why the learner model does not only consist of one set of nodes but is extended by another time slice after each interaction with the learner, as shown in figure 21 (the figure includes only the relevant nodes from the first one and leaves out the observation nodes for clarity). One "time slice" stands for one system-learner interaction. This facilitates the collection of more than one piece of evidence for a piece of knowledge. Moreover, the projection of the probabilities of the learner's knowledge provides the means to include the possibility that the learner learns or forgets something between interaction n and interaction n+1.

The implementation of the Bayesian network is based on the example of Bratko (2001) and can be found in appendix A. Dealing with time slices is added in order to make the network dynamic.

Selection of parameters At the moment, the parameters are set as follows:

- The prior probabilities of the knowledge nodes are set to 0.5, i.e. the probabilities that the learner knows or does not know a topic in advance are the same;
- The conditional probabilities for the quantities and the entity express a prerequisite relation: if all of the connected components are known the probability of knowing is 1, if not, the value is according to the ratio of known parents over the total number of parents;
- The probability of a guess (i.e. that the learner gives a correct answer although he does not know the topic) is 0.01; the probability of a slip (i.e. that the learner gives an incorrect answer although he knows the topic) is 0.1;
- Learn/forget are both 0.

The conditional probability tables are displayed in tables 2, 3, 4, 5, 6 and 7.

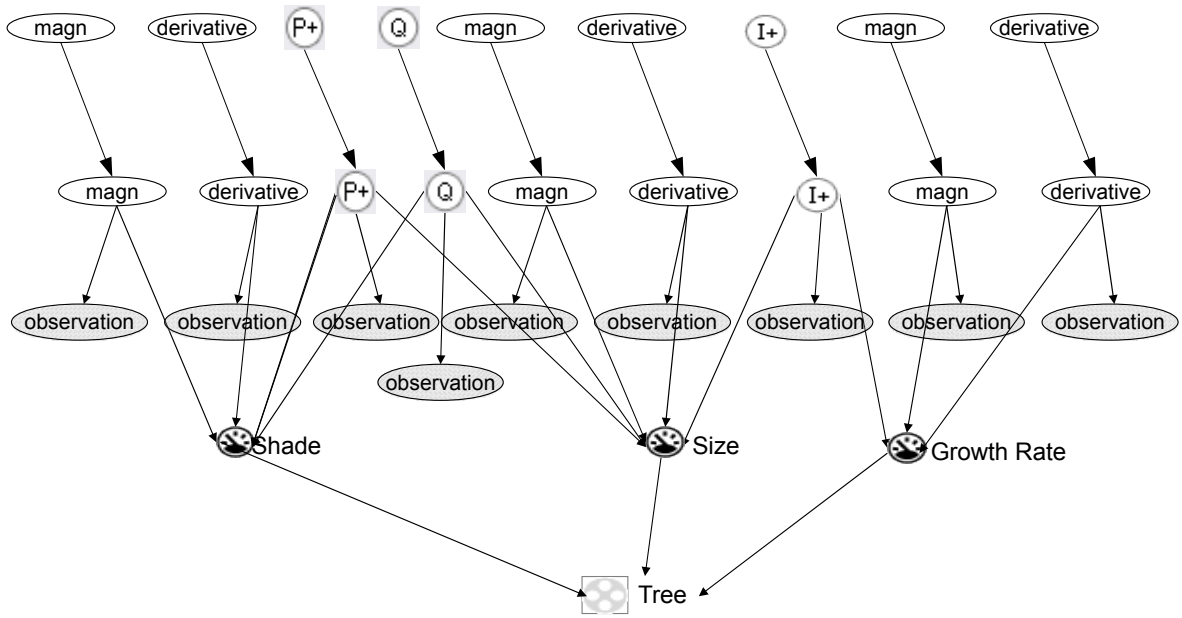

Fig. 21: A learner model network for a growing tree, 2 time slices/interactions

Tab. 2: The CPT for entity tree

size1/N	growth_rate1/N	shade1/N	$p(tree \mid size1/N, growth\_rate1/N, shade1/N)$
known	known	known	1
known	known	unknown	0.7
known	unknown	known	0.7
known	unknown	unknown	0.3
unknown	known	known	0.7
unknown	known	unknown	0.3
unknown	unknown	known	0.3
unknown	unknown	unknown	0

Tab. 3: The CPT for quantity size1

v_size1/N	d_size1/N	inf_pos_size1_-growth_rate1-/N	dir_q_-correspondence_size1_shade1/N	prop_pos_-shade1_size1/N	$p(size1/N \mid d\_size1/N, inf\_pos\_size1\_-growth\_rate1/N, dir\_q\_correspondence\_size1\_-shade1/N, prop\_pos\_shade1\_size1/N)$
known	known	known	known	known	1
known	known	unknown	known	known	0.8
known	unknown	known	known	known	0.8
known	unknown	unknown	known	known	0.6
unknown	known	known	known	known	0.8
unknown	known	unknown	known	known	0.6
unknown	unknown	known	known	known	0.6
unknown	unknown	unknown	known	known	0.4
known	known	known	unknown	known	0.8
known	known	unknown	unknown	known	0.6
known	unknown	known	unknown	known	0.6
known	unknown	unknown	unknown	known	0.4
unknown	known	known	unknown	known	0.6
unknown	known	unknown	unknown	known	0.4
unknown	unknown	known	unknown	known	0.4
unknown	unknown	unknown	unknown	known	0.2
known	known	known	known	unknown	0.8
known	known	unknown	known	unknown	0.6
known	unknown	known	known	unknown	0.6
known	unknown	unknown	known	unknown	0.4
unknown	known	known	known	unknown	0.6
unknown	known	unknown	known	unknown	0.4
unknown	unknown	known	known	unknown	0.4
unknown	unknown	unknown	known	unknown	0.2
known	known	known	unknown	unknown	0.6
known	known	unknown	unknown	unknown	0.4
known	unknown	known	unknown	unknown	0.4
known	unknown	unknown	unknown	unknown	0.2
unknown	known	known	unknown	unknown	0.4
unknown	known	unknown	unknown	unknown	0.2
unknown	unknown	known	unknown	unknown	0.2
unknown	unknown	unknown	unknown	unknown	0

Tab. 4: The CPT for quantity shade1

v_shade1/N	d_shade1/N	dir_q_-correspondence_-size1_shade1/N	prop_pos_-shade1_size1/N	$p(shade1/N \mid v\_shade1/N, d_s hade1/N, dir\_q\_correspondence\_-size1\_shade1/N, prop\_pos\_shade1\_size1/N)$
known	known	known	known	1
known	unknown	known	known	0.75
unknown	known	known	known	0.75
unknown	unknown	known	known	0.5
known	known	unknown	known	0.75
known	unknown	unknown	known	0.5
unknown	known	unknown	known	0.5
unknown	unknown	unknown	known	0.25
known	known	known	unknown	0.75
known	unknown	known	unknown	0.5
unknown	known	known	unknown	0.5
unknown	unknown	known	unknown	0.25
known	known	unknown	unknown	0.5
known	unknown	unknown	unknown	0.25
unknown	known	unknown	unknown	0.25
unknown	unknown	unknown	unknown	0

Tab. 5: The CPT for quantity growth_rate1

v_growth_rate/N	d_growth_rate/N	inf_pos_size1_-growth_rate/N	$p(growth\_rate1 \mid v\_growth\_rate/N, d\_growth\_rate/N, inf\_pos\_size1\_-growth\_rate/N)$
known	known	known	1
known	known	unknown	0.7
known	unknown	known	0.7
known	unknown	unknown	0.3
unknown	known	known	0.7
unknown	known	unknown	0.3
unknown	unknown	known	0.3
unknown	unknown	unknown	0

Tab. 6: The CPT for observations

Know/N	$p(Observation/N \mid Know/N)$
true	0.9
false	0.01

Tab. 7: The CPT for extension of the network (learning/forgetting)

Know/N	$p(Know/N+1 \mid Know/N)$
true	1
false	0

4.5 Experimental Setup

In 4.6, we describe a use case about the "tree and shade" model (4.2), using a simulation about the scenario "a small tree", with the question generation steered according to the state of the learner model.

The dialog listed is generated using a prototype user interface. This user interface controls the components involved and provides a simple interface for adding observations to the student model. Figure 22 gives an overview of the relevant parts. The Quags question generator is used as it is. The Bayesian network (BN) and the mappings connected to it are implemented in this project. The direct user interaction is realized as a mockup, with the question history and the heuristics for selecting a question being done manually. For the question history the listing of the dialog is used. The "heuristic" used for selecting the most appropriate question is to not ask a question two times in a row if alternative questions exist. If only one question was generated for the given restrictions, this question is repeated.

In the experiment, for each of the questions the following sequence is applied:

1. The user interface triggers Quags question generation with restriction to the quantity and concept described by the knowledge node with the lowest probability in the student model[4][5]. The quantity for the restriction is retrieved by looking up the child quantities of the knowledge node[6]. The concept to be applied for question generation is selected using a mapping of the knowledge nodes to the respective concepts which is given as part of the network definition.

2. Manually, one question from the generated list is selected according to the given heuristic and the history of questions that have been asked before. This question is also manually added to the "history".

3. In the listing of the use case (4.6), we state a possible system-learner interaction, including the system asking the question, the learner's answer and the system's

[4] The second sorting criterion is the name of the node. If there is a group of nodes with the same probability, the alphabetically first one is selected.

[5] For first question, question generation is run without restrictions.

[6] I.e. all child nodes which are not observations.

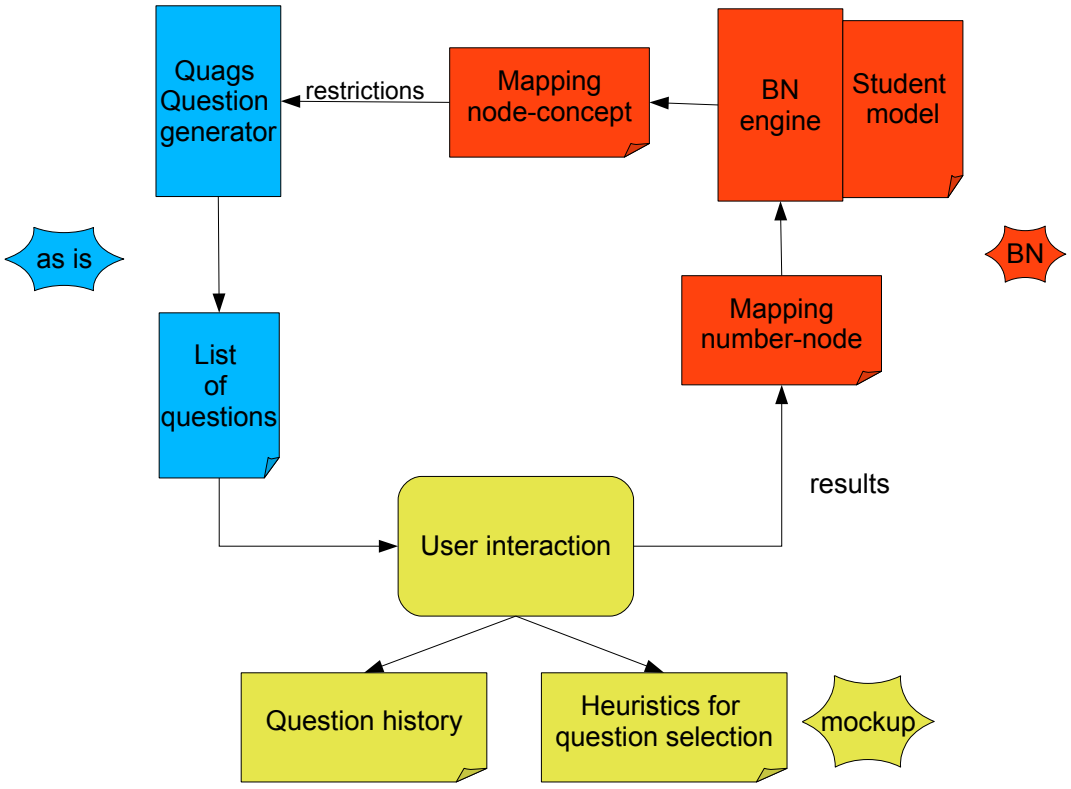

Fig. 22: The parts of the user interaction

feedback. This is not happening in the prototype system, but given in order to provide a better idea what the interaction can look like in a more complete tutoring system.

4. Using the user interface, we enter the learner's answer in the form of `[Node, Correct]`. `Node` is a number that encodes the knowledge node in the learner network which the question is about. `Correct` encodes whether the answer was right or wrong (`r` for right and `w` for wrong). The respective node is selected from a mapping of the number to the knowledge node which is included in the network definition and the evidence is added to the learner model. The network is extended by a new time slice. The root nodes are the (estimated) state of the students knowledge at time n+1 based on the probabilities given the previous evidence at time n.

4.6 Results

The main result of the experiment is the use case of a dialog which is produced as described in the previous section (4.5) and listed below. Figure 23 visualizes how the probabilities in the learner model network evolve for the conversation until all parts of the model have been discussed sufficiently. The probability numbers are given in tables 24 and 25.

In this example, the learner has difficulties with three concepts. We use these three episodes for showing how the sequence is generated and list questions until the end of the dialog.

4.6.1 Learner's difficulty with the value of the quantity shade1 (questions 1-4)

Question 1 In the beginning all of the probabilities have the same starting value (0.5):

obs_d_growth_rate1	0.455
obs_d_shade1	0.455
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.455
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.500
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500

v_growth_rate1	0.500
v_shade1	0.500
v_size1	0.500
growth_rate1	0.500
shade1	0.500
size1	0.500
tree	0.500

The Quags question generation is called without any restrictions. The resulting questions are:

```
Now look at state 1
question(1, What is the value of shade1 of entity tree?, The value of
   shade1 is small)
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
question(1, What is the value of size1 of entity tree?, The value of
   size1 is small)
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(8, Why does shade1 of entity tree have the value small?, size1
   of entity tree has the value small and there is a quantity
   correspondence between shade1 and size1)
question(11, How does growth_rate1 of entity tree influence shade1 of
   entity tree?, growth_rate1 has a positive influence on size1 which
   propagates its change to shade1)
question(5, Why does shade1 of entity tree increase?, size1 of entity
   tree increases and changes in size1 are followed by changes in shade1)
question(5, Why does size1 of entity tree increase?, growth_rate1 of
   entity tree is positive and growth_rate1 has a positive influence on
   size1)
```

We select the first question, as there are no questions in the learner history.
A possible system-learner interaction:

System: What is the value of shade1 of entity tree?
Learner: Medium.
System: This is not true.

The learner's answer is wrong, we enter `[1, w].` The evidence is added to the learner model, the probability of v_shade1 decreases from 0.500 to 0.092, and the model is extended by one time slice.

Question 2 Following the wrong answer, the knowledge node for the value of shade1 (v_shade1) has the lowest probability of being known. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.455
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.092
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.500
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.092
v_size1	0.500
growth_rate1	0.500
shade1	0.398
size1	0.500
tree	0.465

By looking up the children of the node with the lowest probability in the Bayesian network (`v_shade1`) the quantity `shade1` is identified as the next focus of the dialog. The mapping of the knowledge nodes to concepts given with the network definition determines that the concept to be used is `val` (value). Accordingly, the Quags question generation is called with restriction to quantity `shade1` and concept `val` (value). The resulting set of questions is:

```
Now look at state 1
question(17, Which values can shade1 of entity tree adopt?, small, and
   point(medium(shade1)), and large.)
question(1, What is the value of shade1 of entity tree?, The value of
   shade1 is small)
```

```
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
```

We select the first question because it has not been asked yet.
A possible system-learner interaction:

System: Which values can shade1 of entity tree adopt?
Learner: Small and large.
System: This is only part of the answer.

The learner's answer is wrong, we enter `[1, w].` The evidence is added to the learner model, the probability of v_shade1 decreases from 0.092 to 0.010, and the model is extended by one time slice.

Question 3 Following the wrong answer, the knowledge node for the value of shade1 has an even lower and thus still the lowest probability of being known. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.455
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.019
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.500
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.010
v_size1	0.500
growth_rate1	0.500
shade1	0.378
size1	0.500
tree	0.458

Again, the Quags question generation is called with restriction to quantity shade1 and concept value. The resulting set of questions is the same:

```
Now look at state 1
question(17, Which values can shade1 of entity tree adopt?, small, and
   point(medium(shade1)), and large.)
question(1, What is the value of shade1 of entity tree?, The value of
   shade1 is small)
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
```

We select the second question, because the first one was asked in the previous step. A possible system-learner interaction:

System: What is the value of shade1 of entity tree?
Student: The value of shade1 is small.
System: True.

The learner's answer is right, we enter `[1, r].` The evidence is added to the learner model, the probability of v_shade1 increases from 0.010 to 0.479, and the model is extended by one time slice.

Question 4 Because of the correct answer, the probability for the knowledge node for the value of shade1 was increased. However, it is still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.455
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.436
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.500
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.479
v_size1	0.500

growth_rate1	0.500
shade1	0.495
size1	0.500
tree	0.498

Again, the Quags question generation is called with restriction to quantity shade1 and concept value. The resulting set of questions is the same:

```
Now look at state 1
question(17, Which values can shade1 of entity tree adopt?, small, and
   point(medium(shade1)), and large.)
question(1, What is the value of shade1 of entity tree?, The value of
   shade1 is small)
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
```

We select the first question again, because it was not asked in the step before.
A possible system-learner interaction:

System:	Which values can shade1 of entity tree adopt?
Student:	small, and point(medium(shade1)), and large.
System:	Indeed.

The learner's answer is right, we enter `[1, r].` The evidence is added to the learner model, the probability of v_shade1 increases from 0.479 to 0.988, and the model is extended by one time slice.

4.6.2 Learner's difficulty with the derivative of the quantity shade1 (questions 5-8)

Question 5 Now, the value of the shade is more probable to be known than the rest of the items. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.455
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455

d_growth_rate1	0.500
d_shade1	0.500
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500
shade1	0.622
size1	0.500
tree	0.542

The interaction moves on with the next item, the derivative of growth_rate1, (`d_growth_rate1`) which is alphabetically the first one in the list of nodes with the lowest probability. The child node of `d_growth_rate1` determines that the next focus of the dialog is the quantity `growth_rate1`. The mapping of knowledge nodes to concepts given with the network definition returns `der` (derivative) as corresponding concept. The Quags question generation is called with restriction to quantity `growth_rate1` and concept `der` (derivative). For this knowledge node no questions can be generated.

Thus, the generation is re-run for the next piece of knowledge, i.e. the node `d_shade1`. Its child node in the network gives as the next focus quantity `shade1`, and the mapping in the network definition returns `der` (derivative) as corresponding concept. Hence, the Quags question generation is called with restriction to quantity `shade1` and concept `der` (derivative). The resulting question is:

```
Now look at state 1
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
```

The only question is selected and posed to the learner.
A possible system-learner interaction:

System:	What is going to happen with shade1 of entity tree during the simulation?
Student:	shade1 will stay the same.
System:	This is not correct.

The learner's answer is wrong, we enter `[2, w].` The evidence is added to the learner model, the probability of d_shade1 decreases from 0.500 to 0.092, and the model is extended by one time slice.

Question 6 Following the wrong answer, the knowledge node for the derivative of shade1 has the lowest probability of being known. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.092
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.092
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500
shade1	0.520
size1	0.500
tree	0.507

Accordingly, the Quags question generation is called with restriction to quantity shade1 and concept derivative. The resulting question is:

```
Now look at state 1
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
```

The only question is selected and posed to the learner.
A possible system-learner interaction:

System:	What is going to happen with shade1 of entity tree during the simulation?
Student:	shade1 will rise.
System:	This is only part of the answer.

The learner's answer is wrong, we enter `[2, w].` The evidence is added to the learner

model, the probability of d_shade1 decreases from 0.092 to 0.010, and the model is extended by one time slice.

Question 7 Following the wrong answer, the probability of the knowledge node for the derivative of shade1 is even lower and thus still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.019
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.010
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500
shade1	0.500
size1	0.500
tree	0.500

Again, the Quags question generation is called with restriction to quantity shade1 and concept derivative. The resulting question is the same:

```
Now look at state 1
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
```

There is only one question to select.

A possible system-learner interaction:

System: What is going to happen with shade1 of entity tree during the simulation?
Student: shade1 will rise and reach its maximum value, ending up in state 3 via state 2.
System: Correct.

This time, the learner's answer is right, maybe somebody explained the topic in the meantime. We enter `[2, r].` The evidence is added to the learner model, the probability of d_shade1 increases from 0.010 to 0.479, and the model is extended by one time slice.

Question 8 Because of the correct answer, the probability for the knowledge node for the derivative of shade1 was increased. However, it is still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.436
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.479
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500
shade1	0.617
size1	0.500
tree	0.540

Again, the Quags question generation is called with restriction to quantity shade1 and concept derivative. The resulting question is the same:

```
Now look at state 1
question(4, What is going to happen with shade1 of entity tree during the
```

```
 simulation?, shade1 will rise and reach its maximum value, ending up
in state 3 via state 2)
```

There is only one question to select.
A possible system-learner interaction:

System: What is going to happen with shade1 of entity tree during the simulation?
Student: shade1 will rise and reach its maximum value, ending up in state 3 via state 2.
System: Good.

The learner's answer is still right. We enter `[2, r].` The evidence is added to the learner model, the probability of d_shade1 increases from 0.479 to 0.988, and the model is extended by one time slice.

Question 9 The probability of the node for the derivative of the shade is now amongst the highest ones in the list. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.455
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.500
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500
shade1	0.744
size1	0.500
tree	0.585

The interaction moves on with the next item, `d_size1`. Its child quantity, and hence the quantity to be focused on next, is `size1`. The network definition maps knowledge node to the concept `der` (derivative). The Quags question generation is called with restriction to quantity `size1` and concept `der` (derivative). The resulting question is:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
```

We select the only question.
A possible system-learner interaction:

System:	What is going to happen with size1 of entity tree during the simulation?
Student:	size1 will rise and reach its maximum value, ending up in state 3 via state 2.
System:	Indeed.

The learner's answer is right. We enter `[6, r].` The evidence is added to the learner model, the probability of d_size1 increases from 0.500 to 0.989, and the model is extended by one time slice.

Question 10 The sorted list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.455
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.500
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500

shade1	0.744
size1	0.598
tree	0.619

The next node in the list is `dir_q_correspondence_size1_shade1` with child quantities `size1` and `shade1`, and mapped concept `corr` (correspondence). The Quags question generation is called with restriction to quantities `size1` and `shade1` and concept `corr` correspondence. The resulting set of questions is:

```
Now look at state 1
question(8, Why does shade1 of entity tree have the value small?, size1
   of entity tree has the value small and there is a quantity
   correspondence between shade1 and size1)
question(4, What is going to happen with shade1 of entity tree during the
    simulation?, shade1 will rise and reach its maximum value, ending up
   in state 3 via state 2)
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
```

We select the first question, because it has not been asked yet.
A possible system-learner interaction:

System:	Why does shade1 of entity tree have the value small?
Student:	size1 of entity tree has the value small and there is a quantity correspondence between shade1 and size1.
System:	Yes.

The learner's answer is right. We enter `[4, r].` The evidence is added to the learner model, the probability of dir_q_correspondence_size1_shade1 increases from 0.500 to 0.989, and the model is extended by one time slice.

4.6.3 Learner's difficulty with the positive influence of the quantity growth_rate1 on the quantity size1 (questions 11-17)

Question 11 The next item questions are generated for is the influence of growth_rate1 on size1. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.455
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455

obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.500
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.500
shade1	0.866
size1	0.696
tree	0.695

The next node in the group of nodes with the lowest probability is `inf_pos_size1_growth_rate1` (positive influence of size1 on growth_rate1) with child quantities `size1` and `growth_rate1`, and mapped concept `crel` (causal relation). The Quags question generation is called with restriction to quantities `size1` and `growth_rate1` and concept `crel` (causal relation). The resulting set of questions is:

```
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
   size1 of entity tree., growth_rate1 has a positive influence on size1)
```

We select the first question.

A possible system-learner interaction:

System:	What is going to happen with size1 of entity tree during the simulation?
Student:	size1 is going to increase.
System:	This is only part of the answer.

The learner's answer is wrong. We enter `[7, w].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 decreases from 0.500 to 0.092, and the model is extended by one time slice.

Question 12 Following the wrong answer, the knowledge node for the influence of growth_rate1 on size1 has the lowest probability of being known. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.092
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.092
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.357
shade1	0.866
size1	0.614
tree	0.619

Accordingly, the Quags question generation is called with restriction to quantities growth_rate1 and size1 and concept causal relation. The resulting set of questions is:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
   size1 of entity tree., growth_rate1 has a positive influence on size1)
```

We select the second question because the first one was asked in the previous step. A possible system-learner interaction:

System: Describe the influence of growth_rate1 of entity tree on size1 of entity tree.
Student: growth_rate1 has no influence on size1.
System: This is not true.

The learner's answer is wrong. We enter `[7, w].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 decreases from 0.092 to 0.010, and the model is extended by one time slice.

Question 13 Following another wrong answer, the probability of the knowledge node for the influence of growth_rate1 on size1 is even lower and thus still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.019
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.010
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.329
shade1	0.866
size1	0.598
tree	0.604

Again, the Quags question generation is called with restriction to quantities growth_rate1 and size1 and concept causal relation. The resulting set of questions is the same:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
   size1 of entity tree., growth_rate1 has a positive influence on size1)
```

The first question is asked again.

A possible system-learner interaction:

System: What is going to happen with size1 of entity tree during the simulation?
Student: size1 is going to increase.
System: This is only part of the answer.

The learner's answer is wrong. We enter `[7, w].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 decreases from 0.010 to 0.001, and the model is extended by one time slice.

Question 14 The probability of the knowledge node for the influence of growth_rate1 on size1 is still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.011
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.001
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.325
shade1	0.866
size1	0.596
tree	0.602

The restrictions for Quags and the resulting set of questions are the same:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
```

`size1 of entity tree., growth_rate1 has a positive influence on size1)`

We select the second question, because the first one was just asked.
A possible system-learner interaction:

System: Describe the influence of growth_rate1 of entity tree on size1 of entity tree.
Student: growth_rate1 has a positive influence on size1.
System: Indeed.

The learner's answer is right. Maybe somebody has given an explanation in the meantime. We enter `[7, r].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 increases from 0.001 to 0.085, and the model is extended by one time slice.

Question 15 The probability of the knowledge node for the influence of growth_rate1 on size1 increased but it is still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.086
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.085
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.355
shade1	0.866
size1	0.613
tree	0.618

Thus, the restrictions for Quags and the resulting set of questions are the same:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
   size1 of entity tree., growth_rate1 has a positive influence on size1)
```

We select the first question, because the second one was just asked.
A possible system-learner interaction:

System:	What is going to happen with size1 of entity tree during the simulation?
Student:	size1 is going to increase.
System:	This is only part of the answer.

The learner's answer is wrong. Maybe the learner has not understood what influence is doing. We enter `[7, w].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 decreases from 0.085 to 0.009, and the model is extended by one time slice.

Question 16 The probability of the knowledge node for the influence of growth_rate1 on size1 is still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.018
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.009
v_growth_rate1	0.500
v_shade1	0.988
v_size1	0.500
growth_rate1	0.328
shade1	0.866

size1	0.597
tree	0.604

Thus, the restrictions for Quags and the resulting set of questions are the same:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
   size1 of entity tree., growth_rate1 has a positive influence on size1)
```

We select the second question, because the first one was just asked.
A possible system-learner interaction:

System:	Describe the influence of growth_rate1 of entity tree on size1 of entity tree.
Student:	growth_rate1 has a positive influence on size1.
System:	Good.

The learner's answer is right. We enter `[7, r].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 increases from 0.009 to 0.457, and the model is extended by one time slice.

Question 17 The probability of the knowledge node for the influence of growth_rate1 on size1 increased but it is still the lowest. The list of probabilities is:

obs_d_growth_rate1	0.455
obs_d_shade1	0.889
obs_d_size1	0.890
obs_dir_q_correspondence_size1_shade1	0.890
obs_inf_pos_size1_growth_rate1	0.417
obs_prop_pos_shade1_size1	0.455
obs_v_growth_rate1	0.455
obs_v_shade1	0.889
obs_v_size1	0.455
d_growth_rate1	0.500
d_shade1	0.988
d_size1	0.989
dir_q_correspondence_size1_shade1	0.989
prop_pos_shade1_size1	0.500
inf_pos_size1_growth_rate1	0.457
v_growth_rate1	0.500
v_shade1	0.988

v_size1	0.500
growth_rate1	0.485
shade1	0.866
size1	0.687
tree	0.687

Thus, the restrictions for Quags and the resulting set of questions are the same:

```
Now look at state 1
question(4, What is going to happen with size1 of entity tree during the
   simulation?, size1 will rise and reach its maximum value, ending up in
    state 3 via state 2)
question(20, Describe the influence of growth_rate1 of entity tree on
   size1 of entity tree., growth_rate1 has a positive influence on size1)
```

We select the first question, because the first one was just asked.
A possible system-learner interaction:

System:	What is going to happen with size1 of entity tree during the simulation?
Student:	size1 will rise and reach its maximum value, ending up in state 3 via state 2.
System:	Correct.

The learner's answer is right. We enter `[7, r].` The evidence is added to the learner model, the probability of inf_pos_size1_growth_rate1 increases from 0.457 to 0.987, and the model is extended by one time slice.

4.6.4 Finishing the Dialog

The dialog moves on to the next knowledge node, and in the same manner as shown in the three episodes, the process of question generation and updating the learner model is repeated until the lowest probability of a knowledge nodes that questions can be generated for is sufficiently close to 100%.

In the example used in figures 23, 24 and 25, the sequel consists of questions 18 to 28 which are all answered correctly by the learner. The sequence of questions which are generated and selected as described above is:

(18) What is going to happen with shade1 of entity tree during the simulation?
[shade1 will rise and reach its maximum value, ending up in state 3 via state 2.]

(19) Which values can growth_rate1 of entity tree adopt?
[point(zero), and plus.]

(20) Which values can size1 of entity tree adopt?
[small, and point(medium(size1)), and large.]

(21) What is going to happen with size1 of entity tree during the simulation?
[size1 will rise and reach its maximum value, ending up in state 3 via state 2.]

(22) What is going to happen with shade1 of entity tree during the simulation?
[shade1 will rise and reach its maximum value, ending up in state 3 via state 2.]

(23) Which values can shade1 of entity tree adopt?
[small, and point(medium(shade1)), and large.]

(24) What is going to happen with size1 of entity tree during the simulation?
[size1 will rise and reach its maximum value, ending up in state 3 via state 2.]

(25) Why does shade1 of entity tree have the value small?
[size1 of entity tree has the value small and there is a quantity correspondence between shade1 and size1.]

(26) What is going to happen with size1 of entity tree during the simulation?
[size1 will rise and reach its maximum value, ending up in state 3 via state 2.]

(27) Which values can growth_rate1 of entity tree adopt?
[point(zero), and plus.]

(28) Which values can size1 of entity tree adopt?
[small, and point(medium(size1)), and large.]

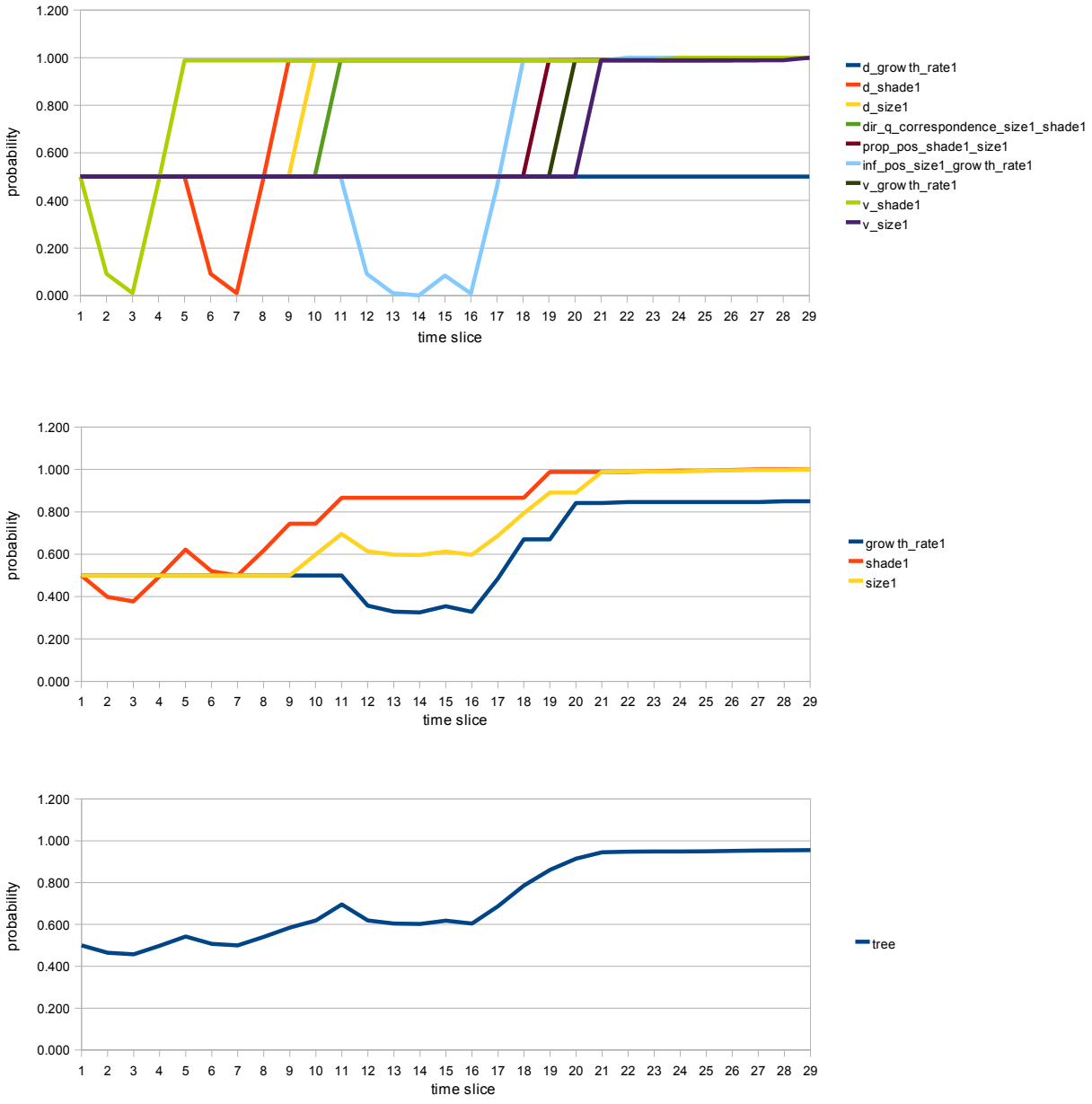

Fig. 23: The probabilities that the learner knows certain pieces of knowledge (graphs)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
obs_d_growth_rate1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455
obs_d_shade1	0.455	0.455	0.455	0.455	0.455	0.092	0.019	0.436	0.889	0.889	0.889	0.889	0.889	0.889	0.889
obs_d_size1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.890	0.890	0.890	0.890	0.890	0.890
obs_dir_q_correspondence_size1_shade1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.890	0.890	0.890	0.890	0.890
obs_inf_pos_size1_growth_rate1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.092	0.019	0.011	0.086
obs_prop_pos_shade1_size1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455
obs_v_growth_rate1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455
obs_v_shade1	0.455	0.092	0.019	0.436	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889
obs_v_size1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455
d_growth_rate1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
d_shade1	0.500	0.500	0.500	0.500	0.500	0.092	0.010	0.479	0.988	0.988	0.988	0.988	0.988	0.988	0.988
d_size1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.989	0.989	0.989	0.989	0.989	0.989
dir_q_correspondence_size1_shade1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.989	0.989	0.989	0.989	0.989
prop_pos_shade1_size1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
inf_pos_size1_growth_rate1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.092	0.010	0.001	0.085
v_growth_rate1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
v_shade1	0.500	0.092	0.010	0.479	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988
v_size1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
growth_rate1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.357	0.329	0.325	0.355
shade1	0.500	0.398	0.378	0.495	0.622	0.520	0.500	0.617	0.744	0.744	0.866	0.866	0.866	0.866	0.866
size1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.598	0.696	0.614	0.598	0.596	0.613
tree	0.500	0.465	0.458	0.498	0.542	0.507	0.500	0.540	0.585	0.619	0.695	0.619	0.604	0.602	0.618

Fig. 24: The probabilities that the learner knows certain pieces of knowledge (numbers, part 1; changing values marked yellow)

	15	16	17	18	19	20	21	22	23	24	25	26	27	28
obs_d_growth_rate1	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455	0.455
obs_d_shade1	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.900	0.900	0.900	0.900	0.900	0.900	0.900
obs_d_size1	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.900	0.900	0.900	0.900	0.900
obs_dir_q_correspondence_size1_shade1	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.900	0.900	0.900	0.900
obs_inf_pos_size1_growth_rate1	0.018	0.417	0.888	0.888	0.888	0.888	0.900	0.900	0.900	0.900	0.900	0.900	0.900	0.900
obs_prop_pos_shade1_size1	0.455	0.455	0.455	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.900	0.900	0.900
obs_v_growth_rate1	0.455	0.455	0.455	0.455	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.900	0.900
obs_v_shade1	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.889	0.900	0.900	0.900	0.900	0.900	0.900
obs_v_size1	0.455	0.455	0.455	0.455	0.455	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.890	0.900
d_growth_rate1	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
d_shade1	0.988	0.988	0.988	0.988	0.988	0.988	0.988	1.000	1.000	1.000	1.000	1.000	1.000	1.000
d_size1	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	1.000	1.000	1.000	1.000	1.000
dir_q_correspondence_size1_shade1	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	1.000	1.000	1.000	1.000
prop_pos_shade1_size1	0.500	0.500	0.500	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	1.000	1.000	1.000
inf_pos_size1_growth_rate1	0.009	0.457	0.987	0.987	0.987	0.987	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
v_growth_rate1	0.500	0.500	0.500	0.500	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	1.000	1.000
v_shade1	0.988	0.988	0.988	0.988	0.988	0.988	0.988	0.988	1.000	1.000	1.000	1.000	1.000	1.000
v_size1	0.500	0.500	0.500	0.500	0.500	0.989	0.989	0.989	0.989	0.989	0.989	0.989	0.989	1.000
growth_rate1	0.328	0.485	0.670	0.670	0.842	0.842	0.846	0.846	0.846	0.846	0.846	0.846	0.850	0.850
shade1	0.866	0.866	0.866	0.989	0.989	0.989	0.989	0.991	0.994	0.994	0.997	1.000	1.000	1.000
size1	0.597	0.687	0.793	0.891	0.891	0.989	0.991	0.991	0.991	0.993	0.996	0.998	0.998	1.000
tree	0.604	0.687	0.786	0.861	0.914	0.945	0.947	0.948	0.949	0.950	0.951	0.953	0.954	0.955

Fig. 25: The probabilities that the learner knows certain pieces of knowledge (numbers, part 2; changing values marked yellow)

5 Discussion

The generated dialog demonstrates how weak points of the learner are identified and selected as subjects of the discourse. Moreover, figures 23, 24 and 25 show that the implementation of the learner model can serve for guiding the system-learner-dialog through all parts of the model until all probabilities are close to 100%[7]. Not only the graphs of the probabilities of the "knowledge nodes", but also the graphs of the according quantities and entity increase in the course of the dialog. This indicates that controlling the dialog according to the knowledge about values, derivatives etc., the "knowledge nodes", can be used to derive the knowledge about the sub systems of the model. The discourse does reach the goals set and the model fulfills its purpose as it was planned.

A disadvantage of the approach as it is now is the size of the conditional probability tables. Whenever a parent node is added, the number of entries in the probability table doubles, for example when a dependency is added to a quantity. This can result in big conditional probability tables that are difficult to manage. Automatically generating the conditional probability tables could be a way of eliminating these unmanageable tables.

As next step for further work, we propose the automatic generation of the Bayesian network as the basis of the student model including the automatic generation of the conditional probability tables. The "tree and shade" model which is used for our feasibility study contains all typical ingredients of a Garp3 model. The automatic generation of the Bayesian network should be realizable by a generalization of the given network based on the types and numbers of the model components and their connections.

Moreover, a topic for further work is the refinement of network parameters, i.e. the probabilities. A user study should be done in order to obtain more precise values. A decision to be made as part of the refinement is whether the guess and slip as well as the learn and forget parameters are the same for all pieces of knowledge or if they should be distinguished, e.g. for each node or different concepts.

Finally, we recommend to investigate the role of the simulation's states for representing learners' knowledge. One way how states could influence the learner model would be in the choice of parameters. They could be set in a way that the probability of knowing something increases more or less fast depending on the complexity of the state graph, so the learner is asked a suitable number of questions until it is decided that a topic is known. For example, in a state graph with three states it might be enough to ask about one item three times to consider it as known; in a state graph with 20 states, a higher number should be required.

We conclude that the discussed approach of a learner model based on a Bayesian network can be used as basis for further developing a learner model for Garp3.

[7] With the exception of the derivative of growth_rate1, because questions about the values of derivatives are not implemented in Quags.

A Implementation

A.1 The Tree and Shade User Model

```
% A Bayesian network for the tree and shade model
:- [bayes_if].

:- dynamic
        parent/2,
        p/2,
        p/3.

:- retractall(parent(_, _)),
        retractall(p(_, _)),
        retractall(p(_, _, _)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define the network structure %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% atoms replace the structures for initial testing of the network


% shade(tree, shade1, continuous, sml)
parent(shade1/N, tree/N).
% size(tree, size1, continuous, sml)
parent(size1/N, tree/N).
% growth_rate(tree, growth_rate1, continuous, zp)
parent(growth_rate1/N, tree/N).

% value(shade1, unk, small, plus)
parent(v_shade1/N, shade1/N).
parent(d_shade1/N, shade1/N).

% value(size1, unk, small, plus)
parent(v_size1/N, size1/N).
parent(d_size1/N, size1/N).

% value(growth_rate1, unk, plus, zero)
parent(v_growth_rate1/N, growth_rate1/N).
parent(d_growth_rate1/N, growth_rate1/N).

% prop_pos(shade1, size1)
parent(prop_pos_shade1_size1/N, shade1/N).
parent(prop_pos_shade1_size1/N, size1/N).

% inf_pos_by(size1, growth_rate1)
```

```
parent(inf_pos_size1_growth_rate1/N, growth_rate1/N).
parent(inf_pos_size1_growth_rate1/N, size1/N).

% dir_q_correspondence(shade1, size1)
parent(dir_q_correspondence_size1_shade1/N, shade1/N).
parent(dir_q_correspondence_size1_shade1/N, size1/N).

% observations are child nodes of hidden knowledge , i.e. latent nodes
% observation(Interaction/Result, Knowledge)
parent(Knowledge, Observation) :-
       observation(Observation, Knowledge).


observation(obs_v_shade1/N, v_shade1/N).
observation(obs_d_shade1/N, d_shade1/N).

observation(obs_v_size1/N, v_size1/N).
observation(obs_d_size1/N, d_size1/N).

observation(obs_v_growth_rate1/N, v_growth_rate1/N).
observation(obs_d_growth_rate1/N, d_growth_rate1/N).

observation(obs_prop_pos_shade1_size1/N, prop_pos_shade1_size1/N).
observation(obs_inf_pos_size1_growth_rate1/N, inf_pos_size1_growth_rate1/N).
observation(obs_dir_q_correspondence_size1_shade1/N,
   dir_q_correspondence_size1_shade1/N).

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define the probabilities %
%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Prior probabilities

% prior prob of knowing an observed node in the beginning
p(Know/0, 0.5) :-
       observation(_, Know/0).

% Conditional probabilities
%
% quantities
p(tree/N, [size1/N, growth_rate1/N, shade1/N], 1).
p(tree/N, [size1/N, growth_rate1/N, \+shade1/N], 0.7).
p(tree/N, [size1/N, \+growth_rate1/N, shade1/N], 0.7).
p(tree/N, [size1/N, \+growth_rate1/N, \+shade1/N], 0.3).
p(tree/N, [\+size1/N, growth_rate1/N, shade1/N], 0.7).
p(tree/N, [\+size1/N, growth_rate1/N, \+shade1/N], 0.3).
```

```
p(tree/N, [\+size1/N, \+growth_rate1/N, shade1/N], 0.3).
p(tree/N, [\+size1/N, \+growth_rate1/N, \+shade1/N], 0).

% size
p(size1/N, [v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 1).
p(size1/N, [v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.8).
p(size1/N, [v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.8).
p(size1/N, [v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.6).
p(size1/N, [\+v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.8).
p(size1/N, [\+v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.6).
p(size1/N, [\+v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.6).
p(size1/N, [\+v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.4).

p(size1/N, [v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.8).
p(size1/N, [v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.6).
p(size1/N, [v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.6).
p(size1/N, [v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.4).
p(size1/N, [\+v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.6).
p(size1/N, [\+v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.4).
p(size1/N, [\+v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.4).
p(size1/N, [\+v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, prop_pos_shade1_size1/N], 0.2).

p(size1/N, [v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.8).
p(size1/N, [v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.6).
p(size1/N, [v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.6).
p(size1/N, [v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.4).
```

```
p(size1/N, [\+v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.6).
p(size1/N, [\+v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.4).
p(size1/N, [\+v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.4).
p(size1/N, [\+v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N,
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.2).

p(size1/N, [v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.6).
p(size1/N, [v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.4).
p(size1/N, [v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.4).
p(size1/N, [v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.2).
p(size1/N, [\+v_size1/N, d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.4).
p(size1/N, [\+v_size1/N, d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.2).
p(size1/N, [\+v_size1/N, \+d_size1/N, inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0.2).
p(size1/N, [\+v_size1/N, \+d_size1/N, \+inf_pos_size1_growth_rate1/N, \+
    dir_q_correspondence_size1_shade1/N, \+prop_pos_shade1_size1/N], 0).


% shade
p(shade1/N, [v_shade1/N, d_shade1/N, dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 1).
p(shade1/N, [v_shade1/N, \+d_shade1/N, dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 0.75).
p(shade1/N, [\+v_shade1/N, d_shade1/N, dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 0.75).
p(shade1/N, [\+v_shade1/N, \+d_shade1/N, dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 0.5).

p(shade1/N, [v_shade1/N, d_shade1/N, \+dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 0.75).
p(shade1/N, [v_shade1/N, \+d_shade1/N, \+dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 0.5).
p(shade1/N, [\+v_shade1/N, d_shade1/N, \+dir_q_correspondence_size1_shade1/N,
    prop_pos_shade1_size1/N], 0.5).
p(shade1/N, [\+v_shade1/N, \+d_shade1/N, \+dir_q_correspondence_size1_shade1/N
    , prop_pos_shade1_size1/N], 0.25).
```

```
p(shade1/N, [v_shade1/N, d_shade1/N, dir_q_correspondence_size1_shade1/N, \+
    prop_pos_shade1_size1/N], 0.75).
p(shade1/N, [v_shade1/N, \+d_shade1/N, dir_q_correspondence_size1_shade1/N, \+
    prop_pos_shade1_size1/N], 0.5).
p(shade1/N, [\+v_shade1/N, d_shade1/N, dir_q_correspondence_size1_shade1/N, \+
    prop_pos_shade1_size1/N], 0.5).
p(shade1/N, [\+v_shade1/N, \+d_shade1/N, dir_q_correspondence_size1_shade1/N,
    \+prop_pos_shade1_size1/N], 0.25).

p(shade1/N, [v_shade1/N, d_shade1/N, \+dir_q_correspondence_size1_shade1/N, \+
    prop_pos_shade1_size1/N], 0.5).
p(shade1/N, [v_shade1/N, \+d_shade1/N, \+dir_q_correspondence_size1_shade1/N,
    \+prop_pos_shade1_size1/N], 0.25).
p(shade1/N, [\+v_shade1/N, d_shade1/N, \+dir_q_correspondence_size1_shade1/N,
    \+prop_pos_shade1_size1/N], 0.25).
p(shade1/N, [\+v_shade1/N, \+d_shade1/N, \+dir_q_correspondence_size1_shade1/N
    , \+prop_pos_shade1_size1/N], 0).


% growth_rate1
p(growth_rate1/N, [v_growth_rate1/N, d_growth_rate1/N,
    inf_pos_size1_growth_rate1/N], 1).
p(growth_rate1/N, [v_growth_rate1/N, d_growth_rate1/N, \+
    inf_pos_size1_growth_rate1/N], 0.7).
p(growth_rate1/N, [v_growth_rate1/N, \+d_growth_rate1/N,
    inf_pos_size1_growth_rate1/N], 0.7).
p(growth_rate1/N, [v_growth_rate1/N, \+d_growth_rate1/N, \+
    inf_pos_size1_growth_rate1/N], 0.3).
p(growth_rate1/N, [\+v_growth_rate1/N, d_growth_rate1/N,
    inf_pos_size1_growth_rate1/N], 0.7).
p(growth_rate1/N, [\+v_growth_rate1/N, d_growth_rate1/N, \+
    inf_pos_size1_growth_rate1/N], 0.3).
p(growth_rate1/N, [\+v_growth_rate1/N, \+d_growth_rate1/N,
    inf_pos_size1_growth_rate1/N], 0.3).
p(growth_rate1/N, [\+v_growth_rate1/N, \+d_growth_rate1/N, \+
    inf_pos_size1_growth_rate1/N], 0).


% Observations:
%
% The probabilites for the observations can be set separately, but
% for now all p(slip) = 1 - 0.9 = 0.1
p(Obs/N, [Know/N], Prob) :-
       observation(Obs/N, Know/N),
       Prob = 0.9.
```

```
% for now all p(guess) = 0.01
p(Obs/N, [\+Know/N], Prob) :-
        observation(Obs/N, Know/N),
        Prob = 0.01.

% Connection between two time slices
% For now, students are not learning and not forgetting
% Remember (i.e. not forget)
p(X/N, [X/M], 1) :-
        parent(X/M, X/N).

% Learn
p(X/N, [\+X/M], 0) :-
        parent(X/M, X/N).

% mapping the knowledge nodes to their information types
% do automatically in the future?
% types to be found in inputorganizer:
% val, rel, der, ineq, corr, calc

type(Node/_, Type) :-
        type(Node, Type).

type(\+Node/_, Type) :-
        type(Node, Type).

type(v_shade1, val).
type(d_shade1, der).

type(v_size1, val).
type(d_size1, der).

type(v_growth_rate1, val).
type(d_growth_rate1, der).

type(prop_pos_shade1_size1, rel).

type(inf_pos_size1_growth_rate1, rel).

type(dir_q_correspondence_size1_shade1, corr).



number(1, obs_v_shade1).
number(2, obs_d_shade1).
```

```
number(3, obs_prop_pos_shade1_size1).
number(4, obs_dir_q_correspondence_size1_shade1).

number(5, obs_v_size1).
number(6, obs_d_size1).

number(7, obs_inf_pos_size1_growth_rate1).

number(8, obs_v_growth_rate1).
number(9, obs_d_growth_rate1).
```

A.2 The Implementation of the Bayesian Network

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Computing the probabilities in a Bayesian network
% (adapted from Bratko's code, p. 370)
% Miriam Brielmann, 18.02.2009
%
% Computing the probabilities in a Bayesian network:
%
% Belief network is represented by relations:
%  parent( ParentNode, Node)
%  p( Node, ParentStates, Prob)
%   where Prob is conditional probability of Node given
%   values of parent variables ParentStates, for example:
%   p( alarm, [ burglary, \+earthquake], 0.99)
%  p( Node, Prob)
%   probability of node without parents
%
% prob( Event, Condition, P):
%  probability of Event, given Cond, is P;
%  Event is a variable, its negotiation, or a list
%  of simple events representing their conjunction

% probT is used to cache the probabilities of one time slice
:- dynamic
       probT/3.

:- retractall(probT(_, _, _)).

% bayes(+Px, +PyGivenX, +Py, -PxGivenY)
%
% Bayes rule for Y being a child node of X
% p(X|Y) = p(X) * p(Y|X) / p(Y)
% in general, with Cond0 = Y and Cond:
% p(X|Cond0) = p(X|Cond) * p(X and Cond) / p(Y|Cond)
%
%
bayes(Px, PyGivenX, Py, PxGivenY) :-
       PxGivenY is Px * PyGivenX / Py.

% prob(+X, +Cond, -Prob)
% with X the node the probability should be calculated for,
% Cond a list of conditions,
% Prob the resulting probability

% The probability is already given by the combination of the parent nodes
```

```
prob(X, Cond, Prob) :-
       p(X, Cond, Prob),
%      permutation(Cond, Given),
       %writef('prob of %w given %w is %w\n', [X, Cond, Prob]),
       !.

% The probability is already given (has been computed)
prob(X, Cond, Prob) :-
       probT(X, Cond, Prob),
       !.


% Probability of one node
prob([X], Cond, P) :-
       !,
       prob(X, Cond, P).

% Probability of conjunction (of X and the other Xs)
% p(X1 and X2 | Cond) = p(X1 | Cond) * p(X2 | X1 and Cond)

% if the negation of X is member of the conditions the result is 0
prob( [(\+X)|Rest], Cond, 0) :-
       member(X, Cond),
       !,
       asserta(probT([(\+X)|Rest], Cond, 0):-!).
       %writef('prob of %w given %w is %w\n', [[X|Xs], Cond, P]).

% if the negation of X is member of the conditions the result is 0
prob( [X|Rest], Cond, 0) :-
       member((\+X), Cond),
       !,
       asserta(probT([X|Rest], Cond, 0):-!).
       %writef('prob of %w given %w is %w\n', [[X|Xs], Cond, P]).

% if X is member of the conditions, it must not be added to the list of
% conditions! the prob of the first term is one, can be skipped
prob( [X|Xs], Cond, P) :-
       member(X, Cond),
       !,
       prob( Xs, Cond, P),
       %writef('prob of %w given %w is %w\n', [[X|Xs], Cond, P]).
       asserta(probT([X|Xs], Cond, P):-!).

prob( [X|Xs], Cond, P) :-
       !,
       prob( X, Cond, Px),
```

```
        prob( Xs, [X|Cond], PRest),
        P is Px * PRest,
        %writef('prob of %w given %w is %w\n', [[X|Xs], Cond, P]).
        asserta(probT([X|Xs], Cond, P):-!).


% Empty conjunction
prob( [], _, 1) :-
        writef('Empty conjunction!\n'),
        !.

% Condition implies X (X is part of the evidence set)
prob( X, Cond, 1) :-
        member( X, Cond),
        %writef('prob of %w given %w is %w\n', [X, Cond, 1]),
        !.

% Condition implies X is false
prob( X, Cond, 0) :-
        member( (\+X), Cond),
        %writef('prob of %w given %w is %w\n', [X, Cond, 0]),
        !.

% Probability of negation is 1 - probability of event
prob( (\+X), Cond, P) :-
        !,
        prob( X, Cond, P0),
        P is 1 - P0,
        asserta(probT((\+X), Cond, P):-!).
        %writef('prob of (\+%w)\%w given %w is %w (1)\n', [X, N, Cond, P]).


%%%%

% Use Bayes rule if condition involves a descendant of X
prob( X, Cond0, P) :-
        % produce a list of conditions without Y
        delete_element( Y, Cond0, Cond),
        % if Y is a descendant of X
        predecessor( X, Y),
%       writef('descendant involved: p(%w|%w)\n', [X, Cond0]),
        !,
        prob( X, Cond, Px),
        prob( Y, [X|Cond], PyGivenX),
        prob( Y, Cond, Py),
        %writef('prob of Y (%w) given %w is %w\n', [Y, Cond, Py]),
```

```
        bayes(Px, PyGivenX, Py, P),
        %writef('prob of X (%w) given %w is %w\n', [X, Cond0, P]),
        asserta(probT(X, Cond0, P):-!).

% Cases when condition does not involve a descendant
% a) X a root cause - its probability given
prob( X, _, P) :-
        p( X, P),
        %writef('%w is a root cause, no descendants involved.\n', [X]),
        %writef('prob of %w is %w\n', [X, P]),
        !.

% b) X has parents: P(X|Cond) = sum(p(X|S)*p(S|Cond))
% with S representing all possible states of the parent nodes
prob( X, Cond, P) :-
        %writef('check for parents of %w with condition %w\n', [X, Cond]),
        % Conditions on parents
        % the probabilities of X given the parent states
        findall((CONDi, Pi), p(X, CONDi, Pi), CPlist),
        %writef('parents: %w\n', [CPlist]),
        % only do it if the node is connected to the network!
        \+ CPlist = [],
        !,
        sum_probs( CPlist, Cond, P),
        %writef('prob of X (%w) given %w is %w\n', [X, Cond, P]).
        asserta(probT(X, Cond, P):-!).

prob( X, _, _) :-
        writef('no prob could be calculated for %w\n', [X]),
        fail.

% sum_probs( CondsProbs, Cond, WeightedSum)
%  CondsProbs is a list of conditions and corresponding probabilities,
%  WeightedSum is weighted sum of probabilities of Conds given Cond

sum_probs([], _, 0).

% S is COND1, P1 is p(X|S)
sum_probs([(COND1, P1)|CondsProbs], COND, P) :-
        %writef('sum_probs\n',[]),
        % p(S|Cond)
        prob( COND1, COND, PC1),
        %writef('p(%w|%w)=%w\n',[COND1, COND, PC1]),
        sum_probs( CondsProbs, COND, PRest),
        P is P1 * PC1 + PRest.
```

```
predecessor( X, Y) :-
       parent( X, Y).


predecessor( X, (\+Y)) :-
       parent( X, Y),
       !.

predecessor( (\+X), Y) :-
       parent( X, Y),
       !.

predecessor( X, Z) :-
       parent( X, Y),
       predecessor( Y, Z).


% delete_element(X, OldList, NewList)
% needed for deleting an element from a list
% because the standard built-in delete is only dealing with lists of
% elements to be deleted
delete_element( X, [X|L], L).

delete_element( X, [Y|L], [Y|L2]) :-
       delete_element( X, L, L2).
```

A.3 Learner Model Specific Interface for Bayesian Network

```
% Miriam Brielmann, 18.02.2009
%
% Interface for manipulating a Bayesian network
%

:- [bayesian].

:- dynamic
        interaction/1,
        evidence/2.

% filter(+Threshold, -Filtered)
% Creates a list of all nodes that have a probability above the
% threshold for the current time slice
filter(Thresh, Filtered) :-
        interaction(Int),
        get_all_nodes(Int, Nodes),
        filter(Thresh, Nodes, Filtered).

% filter(+Threshold, +Nodes, -Filtered)
% Filters the input list (Nodes) and returns all nodes above the
% threshold in the list Filtered (without time information)
%
% nothing left
filter(_, [], []) :-
        !.

filter(Thresh, [Node1/N|Rest1], [Node1|Rest2]) :-
        evidence(N, Ev),
        prob(Node1/N, Ev, Prob),
        Prob > Thresh,
        !,
        filter(Thresh, Rest1, Rest2).

filter(Thresh, [_|Rest1], Rest2) :-
        filter(Thresh, Rest1, Rest2).


% lowest(-LowestProb/Node)
% Returns the node with the lowest probability in the current time slice
lowest(Lowest) :-
        interaction(Int),
        get_know_nodes(Int, Nodes),
        lowest(1, Nodes, Lowest).
```

```
% lowest(+Number, +AllNodes, -LowestNodes)
% Returns the node with the Number lowest probabilities
%
lowest(1, AllNodes, Lowest) :-
       interaction(Int),
       evidence(Int, Ev),
       get_probs(AllNodes, Ev, List),
       sort(List, [Lowest|Sorted]),
       write([Lowest|Sorted]),nl.

lowest(Number, AllNodes, Lowest) :-
       interaction(Int),
       evidence(Int, Ev),
       get_probs(AllNodes, Ev, List),
       sort(List, Sorted),
       write(Sorted),nl,
       LP1 is Number-1,
       length(Part1, LP1),
       append(Part1, [Lowest|_], Sorted).



% shows all the probabilities for the current time slice
show :-
%      profile(x).

%x :-
       interaction(Int),
       write('Interaction: '),
       write(Int), nl,
       show(Int).

% shows the probabilities for a given time slice
show(Int) :-
       evidence(Int, Evidence),
       write('Evidence: '),
       write(Evidence), nl,
       get_all_nodes(Int, Nodes),
       print_probs(Nodes, Evidence).


% get_all_nodes(+Interaction, -Nodes)
% creates a list with all the nodes for the relevant time slice
% (interaction)
get_all_nodes(Int, Nodes) :-
```

```
        setof(Node,
              get_node(Int, Node),
              Nodes).

% get_node(+Interaction, -Node)
% gets a node in the net for the current time slice
%
% gets a parent node
get_node(Int, A/Int):-
        parent(A/Int, _/Int).

% gets a child node
get_node(Int, B/Int):-
        parent(_/Int, B/Int).


% shows all the probabilities for the current time slice
show_know :-
        interaction(Int),
        write('Interaction: '),
        write(Int), nl,
        show_know(Int).

% shows the probabilities for a given time slice
show_know(Int) :-
        evidence(Int, Evidence),
        write('Evidence: '),
        write(Evidence), nl,
        get_know_nodes(Int, Nodes),
        print_probs(Nodes, Evidence).

% get_know_nodes(+Interaction, -Nodes)
% creates a list with the knowledge nodes for the relevant time slice
% (interaction)
get_know_nodes(Int, Nodes) :-
        findall(Node/Int,
              observation(_, Node/Int),
              Nodes).

% get_probs(+Nodes, +Evidence -Probabilities)
% creates a list of probabilities (format: Prob/Node) for a list of
% nodes
get_probs([], _, []) :-
        !.

get_probs([Node1|Rest], Evidence, [Prob/Node1|RestProbs]) :-
```

```
        prob(Node1, Evidence, Prob),
        get_probs(Rest, Evidence, RestProbs).


print_probs([], _) :-
        !.

print_probs([Node1|Rest], Evidence) :-
        prob(Node1, Evidence, Prob),
        write(Node1/Prob), nl,
        print_probs(Rest, Evidence).


% curr_prob(+Node, -Probability)
% returns the probability for one node with given evidence
% in the current time slice
curr_prob(Node, Prob) :-
        interaction(Int),
        evidence(Int, Ev),
        prob(Node/Int, Ev, Prob).


% returns all the current child nodes of Node together with their
    probabilities
% in the list Probs
child_nodes(Node, Probs) :-
        interaction(Int),
        evidence(Int, Ev),
        setof(N/Int/P, (predecessor(Node/Int, N), prob(N, Ev, P)), Probs).


% initialize_net(-Knodes)
% calculate probabilities for the knowledge nodes in state 0,
% i.e. without any evidence given,
% based on the prior probability of the root node
% (for that, the probabilities of the parent nodes are also calculated
% and asserted)
initialize_net(Knodes) :-
        %make sure no interaction or has been counted yet
        retractall(interaction(_)),
        retractall(evidence(_, _)),
        assert(interaction(0)),
        assert(evidence(0, [])),
        % find probabilities for all observed nodes for time slice 0
        findall(Know/0/Prob,
                (
```

```
                observation(_, Know/0),
                prob(Know/0, [], Prob)
                ),
                Knodes
               ).

% input(+Evidence, -Knodes, -Int)
% enters a series (list) of evidence
%
% only one piece left
input([Ev], Knodes, Int) :-
        !,
        update(Ev, _, _),
        extend(Knodes, Int).

input([Ev1|RestEv], Knodes, Int) :-
        update(Ev1, _, _),
        extend(_, _),
        input(RestEv, Knodes, Int).

% update(+Evidence, -Knodes, -Int)
% add evidence to list of evidence and return the updated list of
% knowledge nodes and the number of the current interaction
%
% for a list of evidence:
% empty list = finished
update([], _, _) :-
        !.

update([Ev1|Rest], _, _) :-
        interaction(Int),
        add_evidence(Ev1, Int, _),
        update(Rest, _, _),
        !.

update(Evidence, Knodes, Int) :-
        interaction(Int),
        add_evidence(Evidence, Int, NewEv),
        findall(Know/Int/Prob,
                (
                observation(_, Know/Int),
                prob(Know/Int, NewEv, Prob)
                ),
                Knodes
        ).
```

```
% add_evidence(+NewEvidence, +State, -EvidenceList)
% adds a new piece of evidence (NewEvidence) to the internal list of
% evidence and returns this list (EvidenceList)
add_evidence([], State, PrevEv) :-
       !,
       evidence(State, PrevEv).

% Fail if the negation of the evidence is contained in the list
add_evidence(\+NewEvidence, N, PrevEv) :-
       evidence(N, PrevEv),
       member(NewEvidence/N, PrevEv),
       !,
       write('Evidence could not be added because the negation is part of the
          list.'), nl,
       fail.

% Check: Should not be necessary!
% If the evidence is not known yet, add it to the list
add_evidence(\+NewEvidence, N, EvidenceList) :-
       evidence(N, PrevEv),
       \+member(\+NewEvidence/N, PrevEv),
       !,
       append([\+NewEvidence/N], PrevEv, EvidenceList),
       retract(evidence(N, _)),
       asserta(evidence(N, EvidenceList)).


% Fail if the negation of the evidence is contained in the list
add_evidence(NewEvidence, N, PrevEv) :-
       evidence(N, PrevEv),
       member(\+NewEvidence/N, PrevEv),
       !,
       write('Evidence could not be added because the negation is part of the
          list.'), nl,
       fail.

% If the evidence is not known yet, add it to the list
add_evidence(NewEvidence, N, EvidenceList) :-
       evidence(N, PrevEv),
       \+member(NewEvidence/N, PrevEv),
       !,
       append([NewEvidence/N], PrevEv, EvidenceList),
       retract(evidence(N, _)),
       asserta(evidence(N, EvidenceList)).
```

```
% Otherwise return the same list
add_evidence(_, N, PrevEv) :-
        evidence(N, PrevEv).

% Calls extend/2, ignores results
extend :-
        extend(_, _).

% extend(-Knodes, -NextInt)
% extends the net with one time slice, i.e. prepares the input for the
% next interaction, returns the observed nodes with their probabilities
% and the number of the next interaction
extend(Knodes, NextInt) :-
        interaction(CurInt),
        NextInt is CurInt + 1,
        evidence(CurInt, Evidence),
        findall(Know/NextInt/Prob,
                (
                % the root nodes for the current time slice
                p(Know/CurInt, _),
                    % connect the node in the new slice with the old one
                    assert(parent(Know/CurInt, Know/NextInt)),
                    prob(Know/NextInt, Evidence, Prob),
                    % save the prior probability of the node
                    assert(p(Know/NextInt, Prob)),
                    % in the next time slice we are not interested in
                    % the previous anymore, so remove the connection
                    retract(parent(Know/CurInt, Know/NextInt))
                ),
                Knodes),
        retractall(probT(_, _, _)),
        retract(interaction(_)),
        assert(interaction(NextInt)),
        assert(evidence(NextInt, [])).
```

A.4 User Interface

```
% Miriam Brielmann, 19.03.2009
% test interaction Bayes - Quags
:- [usermodel/tree_and_shade].

test :-
       initialize_net(_),
       % generate all questions
       wizigarp:question_generator(_, criteria(_, _, _, _, _), _, _, _, _),
       %writef('Questions: %w\n', Q),
       read(A),
       do(A).


do(stop) :-
       !.

% do([+Number, +Correct])
% Updates the model with the input for the node of the given number
% and the correctness (r=right, w=wrong)
% and generates the next set of questions
do([Number, Correct]) :-
       !,
       in(Number, Correct, Int),
       out(Int, 1),
       read(A),
       do(A).

do(_) :-
       writef('Please repeat input:\n'),
       read(A),
       do(A).

% in(+Number, +Correct, -Int)
in(Number, r, Int) :-
       !,
       number(Number, InNode),
       input([InNode], _, Int).


in(Number, w, Int) :-
       number(Number, InNode),
       input([\+InNode], _, Int).

% out(+Int, +Number)
```

```
% Generates a list of questions for interaction Int and Number of
% probabilities. If no questions can be generated, the number of
% probabilities taken into consideration is increased and the generation
% repeated.
out(Int, Number) :-
       % get the probabilities for all nodes
       get_know_nodes(Int, Nodes),
       % get the node with the lowest probability
       % print_probs(Nodes, Ev),
       lowest(Number, Nodes, Prob/(Lowest/_)),
       process_node(Lowest, Prob, Int, Number).

process_node(_, Prob, _, _) :-
       Prob > 0.99,
       !,
       writef('Congratulations, you have mastered the entire model!\n').

process_node(Lowest, _, Int, Number) :-
       % find the information type for the node
       findall(Type, type(Lowest, Type), Types),
       % get the (direct) children (= relevant quantities for the node)
       findall(Child, (parent(Lowest/Int, Child/Int),\+observation(Child/Int,
          Lowest/Int)), Quantities),
       % generate the relevant questions
       wizigarp:question_generator(_, criteria(_, _, _, _, Types), _,
          Quantities, _, Q),
       %writef('Questions: %w\n', Q).

       again(Q, Int, Number).

again(Q, Int, Number) :-
       var(Q),
       writef('no questions generated'),
       Next is Number+1,
       out(Int, Next),
       !.

again(_, _, _).
```

References

Baker, R. S., Corbett, A. T., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. *ITS 2008*, (pp. 406–415).

Beck, J. E., Chang, K., Mostow, J., & Corbett, A. (2008). Does help help: Introducing the bayesian evaluation and assessment methodology. *ITS 2008*, (pp. 383–394).

Bouwer, A. (2005). *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*. SIKS; Universiteit van Amsterdam [Host].

Bratko, I. (2001). *Prolog Programming for Artificial Intelligence*. Addison Wesley.

Bredeweg, B., Liem, J., Bouwer, A., & Salles, P. (2006). D6.9.1 - curriculum for learning about qr modelling.

Brielmann, M. (2008). Considerations about a system for knowledge communication. Final paper for the course "Knowledge Communication".

Charniak, E. (1991). Bayesian networks without tears. *AI Magazine*, *12*, 50–63.

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

Darwiche, A. (2008). *Handbook of Knowledge Representation*, chap. 11. Bayesian Networks. Elsevier B.V.

de Kleer, J., & Williams, B. (1987). Diagnosing multiple faults. *Artificial Intelligence*, *32*(1), 97–130.

de Koning, K., Bredeweg, B., Breuker, J., & Wielinga, B. (2000). Model-based reasoning about learner behaviour. *Artificial Intelligence*, *117*(2), 173–229.

Goddijn, F. (2002). *Automatische vraaggeneratie bij kwalitatieve simulaties*. Master's thesis, Universiteit van Amsterdam.

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

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

Russell, S. J., & Norvig, P. (1995). *Artificial Intelligence – A Modern Approach*. Prentice Hall.