Chapter 14
Bayesian Student Modeling

Cristina Conati

Department of Computer Science, University of British Columbia,
2366 Main Mall, Vancouver, BC, V6G3C1
conati@cs.ubc.ca

Abstract. Bayesian networks are a formalism for reasoning under uncertainty that has been widely adopted in Artificial Intelligence (AI). Student modeling, i.e., the process of having an ITS build a model of relevant student's traits/states during interaction, is a task permeated with uncertainty, which naturally calls for probabilistic approaches. In this chapter, I will describe techniques and issues involved in building probabilistic student models based on Bayesian networks and their extensions. I will describe pros and cons of this approach, and discuss examples from existing Intelligent Tutoring Systems that rely on Bayesian student models

14.1 Introduction

One of the distinguishing features of an Intelligent Tutoring System (ITS) is that it is capable of adapting its instruction to the specific needs of each individual student, as good human tutors do. Adaptation can be performed at different levels of sophistication, from responding to student observable performance (e.g., errors), to targeting student assessed knowledge (or lack thereof), to helping students achieve specific goals (e.g., generate a given portion of a problem solution), to reacting to student emotions, to scaffolding meta-cognitive abilities (e.g., self-monitoring).

The more an ITS needs to know about its student to provide the desired level of adaptation, the more challenging it is for the ITS to build an accurate *student model* (see chapter by Beverly Woolf) based on the information explicitly available during interaction, because this information usually provides only a partial window on the desired student states. In other words, student modeling can be plagued by a great deal of uncertainty. In this chapter, I will illustrate an approach to handle this uncertainty that relies on the sound foundations of probability theory: Bayesian networks (Pearl 1988). Since the late eighties, Bayesian networks have been arguably the most successful approach for reasoning under uncertainty in AI, and have been widely used for both user modeling and student modeling. The rest of this chapter starts by providing some basic definitions. Next, it introduces *Dynamic Bayesian networks*, an extension of Bayesian networks to handle temporal information, and provides case studies to illustrate when and how to use static vs. dynamic networks in student modeling. The last part of the chapter discusses two

R. Nkambou et al. (Eds.): Advances in Intelligent Tutoring Systems, SCI 308, pp. 281–299.
springerlink.com

main challenges in using Bayesian networks in practice: how to choose the network structure and how to specify the network parameters. For each of these challenges, the chapter illustrates a variety of solutions and provides examples of how they have been used in applications to student modeling.

14.2 Bayesian Networks in a Nutshell

Bayesian networks are graphical models designed to explicitly represent conditional independence among random variables of interest, and exploit this information to reduce the complexity of probabilistic inference (Pearl 1988). Formally, a Bayesian network is a directed acyclic graph where nodes represent random variables and links represent direct dependencies among these variables. If we associate to each node X_i in the network a conditional probability table (CPT) that specifies the probability distribution of the associated random variable given its immediate parent nodes *parents(X_i),* then the Bayesian network provides a compact representation of the Joint Probability Distribution (JPD) over all the variables in the network.

$$\mathbf{P}(X_1, \ldots, X_n) = \prod\nolimits^{n}_{i=1} \boldsymbol{P}\,(X_i \mid Parents(X_i)) \tag{1}$$

This equation holds assuming that the network has been constructed so that each node is conditionally independent of all its non-descendant nodes given its parents (see (Russel and Norvig 2010) for more details).

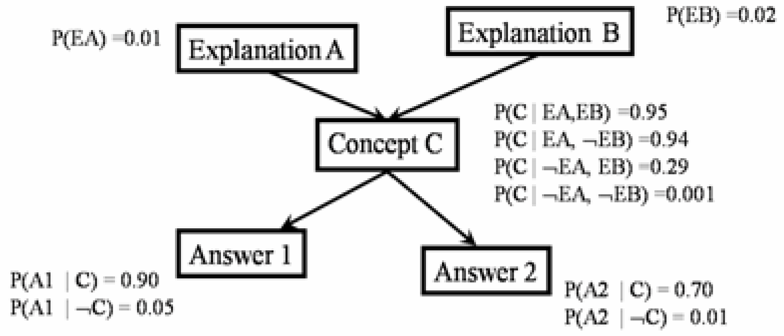

Fig. 14.1 Sample Bayesian network

Figure 14.1 shows a simple Bayesian network representing the following domain: the nodes *Explanation A* and *Explanation B* (indicated as *EA* and *EB* in the relevant CPTs) are binary variables each representing the probability that a student receives a corresponding explanation of concept C. The two explanations are provided independently, e.g., one at school by a teacher and one at home by a parent. The node *Concept C* (indicated as *C* in the relevant CPTs) is a binary variable representing the probability that a student understands the corresponding concept. The nodes *Answer 1* and *Answer 2* (indicated as *A1* and *A2* in the relevant CPTs) are binary variables each representing the probability that a student responds correctly to two different test questions related to concept C. The links and conditional probabilities in the network represent the probabilistic dependencies between receiving each of the two possible explanations for the concept, understanding it and then being able to answer related test questions correctly.

14.3 Static vs. Dynamic Bayesian Networks

The Bayesian network in Fig. 14.1 is *static*, i.e., it is suitable to perform probabilistic inference over variables with values that don't change over time. What changes and is tracked by a static Bayesian network is the belief over the state of these variables as new evidence is collected, i.e., the posterior probability distribution of the variables given the evidence.

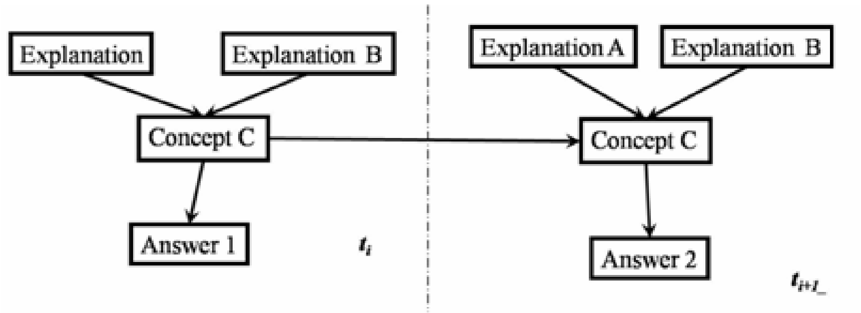

Fig. 14.2 Example DBN

Dynamic Bayesian networks (Dean and Kanazawa 1989), on the other hand, track the posterior probability of variables whose value change overtime given sequences of relevant observations. A Dynamic Bayesian networks (DBN from now on) consists of *time slices* representing relevant temporal states in the process to be modelled. For instance, Fig. 14.2 shows two time slices of a dynamic version of the network in Fig. 14.1. The first slice to the left represents the state of the variables *Concept C*, *Explanation A* and *Explanation B* from Fig. 14.1 after observing a student's answer to the first test at a given time t_i. The second slice represents the state of the same variables after observing a student's answer to the second test at a successive time t_{i+1}. The link between the variables for *Concept C* at times t_i and t_{i+1} models the influence of time on knowledge of this concept. It can be used, for instance, to model forgetting by adding to the CPT for *Concept C* at time t_{i+1} a non-zero probability that the student does not know *concept C* at that time given that she knew it at time t_i.

A key difference between the static network in Fig. 14.1and the dynamic network in Fig. 14.2 is in how evidence on student test answers is taken into account to update the posterior probability of *Concept C*. In Fig. 14.1, two subsequent observations on *Answer 1* and *Answer 2* would have the same weight in updating the probability of

Concept C, which makes sense if the true value of that variable does not change as the observations are gathered. In Fig. 14.2, the effect of having observed *Answer 1* at t_i on the probability of *Concept C* at t_{i+1} is mediated by the probability of *Concept C* at time t_i, while having observed *Answer 2* at t_{i+1} has a direct effect. This makes sense if the true value of *Concept C* can change overtime, because more recent observations are better reflections of the current state of a dynamic process than older ones.

14.3.1 Sample Applications to Student Modelling

Static networks can be used in student modeling as assessment tools under the assumption that the variables to be assessed (e.g., knowledge) are not changing as new evidence (e.g., test results) comes in. For instance, Mislevy (1995) describes a Bayesian student model used by the HYDRIVE tutoring system to assess a variety of skills and knowledge related to troubleshooting an aircraft hydraulics system. Martin and Vanlehn (1995) use a Bayesian student model for off-line assessment of student physics knowledge from evidence on completed problem solutions. Arroyo and Woolf (2005) describe a Bayesian network that assesses student attitudes toward learning with Wayang Outpost, an ITS for math (e.g., whether the student liked the system, found it helpful, learned from it) from statistics on the student interaction with the system (e.g., time spent per problem, time spent per action, average incorrect actions).

One of the first examples of using DBNs in student modeling is the *knowledge tracing* mechanism implement in the CMU Cognitive Tutors (Corbett and Anderson 1995). This mechanism uses Bayes theorem to compute the probability of mastering a rule at time t_{i+1} as a function of both the probability of knowing the rule at time t_i and observations of student problem solving steps pertaining to that rule at time t_{i+1}. While the original formulation of this mechanism was not in terms of DBNs, Reye (Reye 1998)has shown that it can be formulated as a DBN with the same basic behavior. One limitation of this *knowledge tracing* mechanism is that it requires knowing exactly which domain rule the current student solution step refers to. In order to eliminate possible ambiguities in mapping student solution steps with domain rules, this approach requires that students follow one specific solution defined a priory in the student model. For the same reason, it requires that students explicitly show all their solutions steps, i.e., it does not allow students to combine solution steps in their heads and generate actions that are the results of these mental computations. These requirements result in fairly constrained interaction that may become frustrating for some students. Finally, in *knowledge tracing* probabilistic update is limited to one rule at the time, i.e., this mechanism does not exploit the dependencies among the different rules involved in creating a complete problem solution.

The student model of the Andes tutoring system for physics (Conati et al. 2002) extends the approach proposed by Martin and Vanlehn (1995) to address the above limitations of knowledge tracing. For each problem solved by a student, Andes builds a static Bayesian network whose nodes and links represent how the various steps in a problem solution derive from previous steps and physics rules

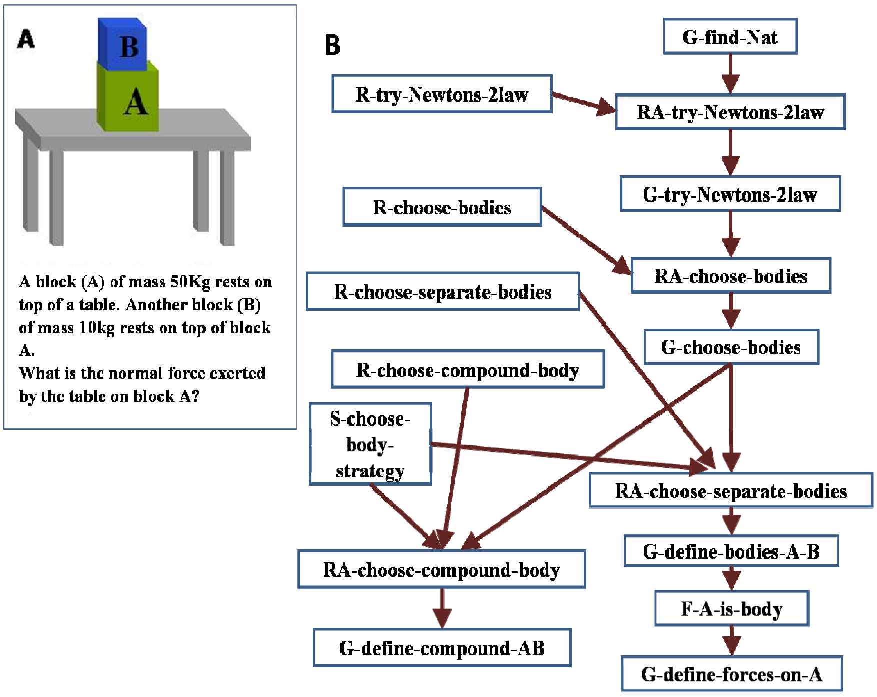

Fig. 14.3 A physics problem and a segment of the corresponding Bayesian network in the Andes tutoring system

(*task-specific network* from now on). For instance, Fig. 14.3B shows a (simplified) section of the task specific network for the problem in Fig. 14.3A, involving the application of Newton's second law to find the value of a normal force. Nodes in this network represent (i) facts corresponding to explicit solution steps (nodes labeled with a *F-* prefix in, Fig. 14.3B); (ii) problem solving goals (nodes labeled with a *G-* prefix); (iii) physics rules (nodes labeled with a *R-* prefix) that generate these facts and goals when applied to preceding facts and goals in the solution. Specific rule applications are indicated by nodes labeled with a *RA-* prefix in Fig. 14.3B. Alternative ways to solve a problem are represented as alternative paths to one or more solution steps in the network. Students can perform problem solving steps in their heads as they desire. When a problem solving step is entered in the Andes interface, Andes retrieves the corresponding fact node in the current task-specific Bayesian network, sets its value to *true* and computes the posterior probability of the other nodes in the network given this new evidence. All nodes involved in generating this step (i.e., all ancestors of the corresponding fact node) may be influenced by this update, with strength dictated by the probabilistic dependencies defined in the network's CPTs. Essentially, the task-specific Bayesian network allows Andes to guess which implicit reasoning has generated a given step, with the accuracy of the guess being influenced by how many steps the student has kept in her head and how many alternative ways to generate each step are represented in the network.

It should be noted that the task-specific network that Andes uses to track how a student solves a specific problem is not dynamic. Instead, Andes uses a form of dynamic network to track the evolution of student knowledge from one solved problem to the next. In particular, Andes maintains a *long-term student model* that encodes the posterior probability of each physics rule known by the system given all the solutions that a student has generated to so far. When the student starts a new problem, Andes generates the task-specific network for that problem as in Fig. 14.3, and initializes the prior probabilities of the rule nodes in the network using the posterior probabilities of the corresponding rules in the long-term model. As soon as the student terminates the problem, Andes discards its task-specific network, but saves the posterior probability of each of the network's rule nodes in the domain-general student model. This probability will then become the prior of the rule node in the task-specific network for the next problem that uses that rule. This process essentially corresponds to having a DBN where each time slice contains a rule node for each rule in the Andes' knowledge base; a new time slice is created when the student opens a new problem, and spans the time it takes the student to terminate problem solving. Removing a time slice when problem solving is over and saving rule posteriors to be used as priors in the next time slice is a form of *recursive filtering* (or *roll-up*). This process allows for maintaining at most two time slices in memory, as opposed to all the time slices tracked (Russel and Norvig 2010).

An alternative to the approach used in Andes is to create a new time slice every time a student generates a new action. We did not adopt this approach in Andes because the roll-up mechanism can be computationally expensive when performed after every student action on networks as large as Andes'. While this approximation may prevent Andes from precisely tracking learning that happens in between solution steps, it did not prevent Andes and its student model to perform well in empirical evaluations (Conati et al. 2002). Because, in the worst-case scenario, probabilistic update in Bayesian networks is intractable, simplifications like the one discussed here must often be made to ensure that the networks are usable in practice, and their impact/acceptability must be verified empirically. (Murray et al. 2004) describe an approach that does create a new slice after every student actions in networks comparable to Andes'. Despite adopting techniques to make network structure and CPTs more compact, performance testing based on simulated student actions showed that exact inference on the resulting models was not feasible. Using algorithms for approximate inference (Russel and Norvig 2010) improved performance, but still resulted in delayed response times on the larger networks tested.

An example of a DBN-based student model that creates time slices after every action and that has been used is practice is found in Prime Climb, an educational game to help students learn number factorization.

In Prime Climb students in 6th and 7th grade practice number factorization by pairing up to climb a series of mountains. Each mountain is divided into numbered sectors (see Figure 4), and each player can only move to a number that does not

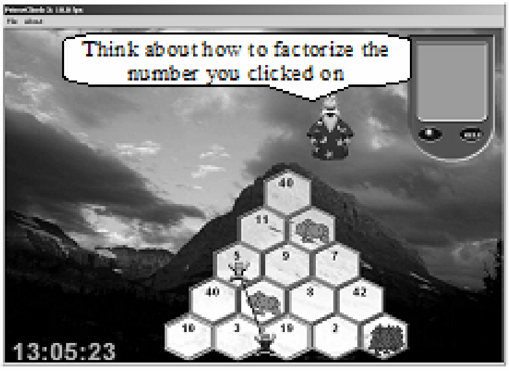

Fig. 14.4 The Prime Climb Interface

share any common factors with her partner's number, otherwise s/he falls. To help students with the climbing task, Prime Climb includes a pedagogical agent (see Figure 4) for each player, that provides individualized support, both on demand and unsolicited, when the student does not seem to be learning from the game. To provide well-timed and appropriate interventions, the agent must have an accurate model of student learning, but maintaining such model is hard because performance tends to be a fairly unreliable reflection of student knowledge in educational games. PrimeClimb uses DBNs to handle the uncertainty involved in this modelling task. More specifically, there is a DBN for each mountain that a student climbs (the *short-term student model*). This DBN assesses the evolution of a student's number factorization knowledge during game play, based on the student's game actions. Each time slice in the DBN includes a *Factorization node* F_x for each number that is relevant to make correct moves on the current mountain (i.e., there is a node for each number on the mountain and for each of its factors). Each of these factorization nodes represents whether the student has mastered the factorization of that number. A new time slice is created after every new student action, e.g., after the student clicks on a number *x* to move there. The one-slice-per-action approach is feasible in Prime Climb because each time slice rarely contains more than a few dozens nodes. We will provide more details of the nature of the Prime Climb's DBNs in a later section.

14.4 Using Bayesian Networks in Practice

There are several advantages in using Bayesian networks for reasoning under uncertainty in general, and for student modeling in particular.

- They provide a more compact representation of the joint probability distribution (JPD) over the variables of interest. To fully specify the JDP $P(X_1, \ldots, X_n)$ over variables $X_1, \ldots, X_n$, it is necessary to specify the probability of each possible combination of variable's values (e.g., m^n numbers in the case of *n* *m*-valued variables). To fully specify the same distribution expressed via a

Bayesian network, it is sufficient to specify for each node with k parents the m^k entries of the associated CPT. If $k << n$, i.e., if the variables to be represented are sparsely connected, then the Bayesian network brings a substantial saving in the number of parameters that need to be specified.

- Algorithms have been developed that exploit the network's structure for computing the posterior probability of a variable given the available evidence on any other variable in the network. While the worse case complexity of probabilistic inference in Bayesian networks is still exponential in the number of nodes, in practice it is often possible to obtain performances that are suitable for real-world applications.
- The intuitive nature of the graphical representation facilitates knowledge engineering. It helps developers focus on identifying and characterizing the dependencies that are important to represent in the target domain. Even when dependencies are left out to reduce computational complexity, these decisions are easy to track, record and revise based on network structure, facilitating an iterative design-and-evaluation approach to model construction.
- Similarly, the underlying network structure facilitates the process of generating automatic explanations of the results of probabilistic inference, making Bayesian networks very well suited for applications in which it is important that the user understands the rational underling the system behavior, as it is often the case for Intelligent Tutoring systems (e.g., Zapata-Rivera and Greer 2004).
- Finally, Bayesian networks lend themselves well to support decision making approaches that rely on the sound foundations of decision theory. This means that selection of tutorial actions can be formalized as finding the action with maximum expected utility given a probability distributions over the outcomes of each possible action and a function describing the utility (desirability) of these outcomes (e.g., Murray et al. 2004; Mayo and Mitrovic 2001).

As is the case for any representation and reasoning paradigm, however, the benefits brought by Bayesian networks come with challenges. The two that arguably have the highest impact on the effort required by adopting this technology are: how to select a suitable structure and how to set the necessary network parameters. The next section discusses these two challenges and solutions proposed in the context of using Bayesian networks in student modeling.

14.5 Choosing Network Structure and Parameters: Examples from Student Modeling

14.5.1 Network Structure

14.5.1.1 Structure Defined Based on Knowledge

One common misconception related to structure definition in Bayesian networks is that the direction of the link between two variables must represent causality. In reality, the only constraint on structure is that every variable be (or can be reasonably assumed to be) independent of all its non-descendant nodes in the network, given its parent nodes. What is true is that structuring the network in the direction of causality makes it easier to satisfy the above constraint, because effects are independent of any previous influence given their immediate causes. In the domain represented in Fig. 14.1, for instance, whether the student understands or not concept C fully defines the probability that the student be able to answer questions about that concept, regardless of which explanation, if any, the student received.

Furthermore, defining links in the causal direction generally results in a more sparsely connected network. In our example, because understanding the concept fully specifies the probability of each answer, there is no direct dependency between the answers and thus there is no need for a link between the corresponding nodes. There is also no need for a direct link between the two *explanation* nodes, because we said they are provided independently.

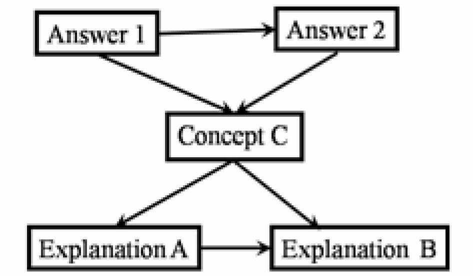

Fig. 14.5 Alternative structure for the Bayesian network in Figure 1

On the other hand, if we define the network as in Fig. 14.5, things change. We need a direct link between the two answer nodes because, given no other information, the belief that a student can generate a correct answer to a test is affected by whether or not the student can generate a correct answer to a different test that taps the same knowledge. Similarly, we need a direct link between the two explanation nodes because they are dependent if we know the true state of the student understanding of concept B. For instance, knowing that the student understands the concept and did not receive explanation A should increase the probability that the student received explanation B. This relationship between explanation A, explanation B and the understanding of concept C is fully captured by the structure in Fig. 14.1, but needs the extra arc between EA and EB in Fig 14.5. Still, the two networks in Fig 14.1 and Fig. 14.5 are equivalent if their CPTs are specified so that they represent the same JPD over the five variables involved. Which structure to select depends mostly on how much effort is required to specify the necessary network parameters (i.e., probabilities in the CPTs). Sparser networks include fewer parameters, but it is also important to consider how easy it is to quantify the needed probabilities.

In Andes, for instance, network structure is purely causal, capturing the following basic relation between knowledge of physics principles and problem solving steps: in order to perform a given problem solving step, a student needs to know the related physics rule and the preconditions for applying the rule. If a step can be derived from different rules, the student needs to apply at least one of them. As we will see in more detail in the next section, this causal structure yields very intuitive CPTs that can be specified via a limited number of parameters.

Matters are bit more complicated with the student model for the Prime Climb educational game. As we mentioned in a previous section, the student's progress on a Prime Climb mountain is tracked by a DBN that includes factorization nodes F_x for all the numbers on that mountain and their factors. Click nodes C_x are introduced in the model when the corresponding actions occur, and are set to either *true* or *false* depending upon whether the move was correct or not. Fig. 14.6 illustrates the structure used in the model to represent the relations between factorization and click nodes. The action of clicking on number *x* when the partner is on number *k* is represented by adding a click node C_x with parent nodes F_x and F_k (see Fig. 14.6b).

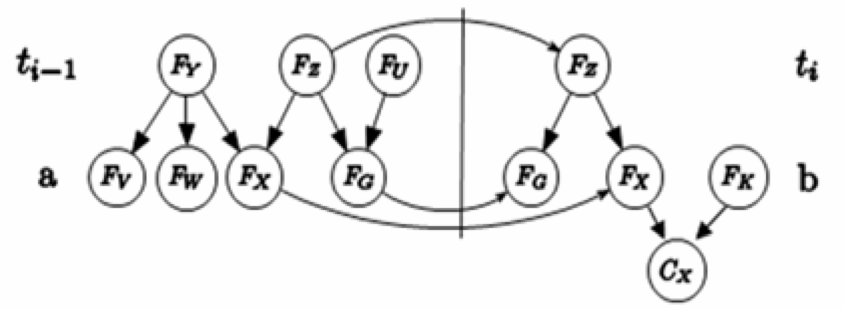

Fig. 14.6 Factorization nodes in the Prime Climb student model, where Z=X*G and Y=V*W*X; b: Click action

This structure represents the causal relationship between factorization knowledge and game actions that depend on it, which is intuitive to formalize: the correctness of a click is influenced by whether the student knows the factorization of the two numbers involved. The probability should be very high if the student knows both numbers, lower if the student knows only one number, and close to 0 if the student knows neither. Less obvious is how to choose the structure that represents the relationship between the factorization knowledge of a number and the factorization knowledge of its factors, because this relationship is not strictly causal. The rationale underlying the structure that was chosen for the Prime Climb network was derived based on discussion with mathematics teachers: knowing the prime factorization of a number influences the probability of knowing the factorization of its factors, while the opposite is not true. It is hard to predict if a student knows a number's factorization given that s/he knows how to factorize its non-prime factors. To represent this rationale, factorization nodes are linked as parents to nodes representing their non-prime factors. The conditional probability table (CPT) for each non-root factorization node (e.g. F_x in

Fig. 14.6a) is defined so that the probability of the node being known is high when all parent factorization nodes are true, and decreases proportionally with the number of unknown parents.

14.5.1.2 Structure Defined Based on Data

So far we have discussed how to define network structure based on existing knowledge of the dependencies among the relevant variables, but this approach is not feasible when the variables involved are not as clearly related as the ones in Andes and Prime Climb. The alternative is to define the structure based on data. Existing algorithms (e.g., Buntine 1996; Moore and Wong 2003) perform some form of heuristics search over the space of possible structures. The heuristics used to evaluate points in the search space generally rely on either statistical measures of correlation to verify whether the dependencies implicit in a given structure reflect the dependencies in the data, or measures related to the model's log likelihood *P(data|model)*, i.e., how well a given model explains the available data. These algorithms, however, require substantial amount of data to learn complex networks, which has limited their adoption in student modelling so far. To deal with limited data availability, existing work on learning the structure of Bayesian student models has combined ideas from these algorithms with heuristics based on knowledge of the target domain. Zhou and Conati (2003) for instance, have used a data-based approach to define the structure of a Bayesian student model that combines information on student personality and interaction patterns to assess student goals while playing Prime Climb. Using expert knowledge to define the structure of this DBN was not possible. While there are theories in psychology that can be used to relate personality to goals users may be pursuing while playing an educational game (e.g,, learn vs. having fun), these theories are too high-level to allow defining specific dependencies among these variables (see for instance Costa and McRae (1992)). Similarly, while it is intuitive that interaction behaviours should are in general affected by user goals, there is limited knowledge on how goals actually impact interaction behaviours in novel environments such as Prime Climb.

To learn the structure of the goal assessment network from data, Zhou and Conati (2003) run a user study during which the interaction patterns of students playing Prime Climb were logged and questionnaires were used to collect data on user personality and interaction goals. Because the amount of data collected was not sufficient to reliably apply existing algorithms to learn the complete network structure, this work used a greedy variation that separately builds and then combines different subparts of the network. The dependencies to be represented in each subpart are selected by running a correlation analysis over the relevant variables and choosing only those correlations that are statistically significant and above a given threshold for strength. The choice among the alternative structures that can represent the selected dependencies is made based on measures of log likelihood, and by using intuition to choose between structures with similar scores. Although this approach is not sound because the log marginal likelihood measure is not additive over network subparts, the resulting network (shown in Fig. 14.7) showed to be effective in assessing student goals when inserted in a larger model

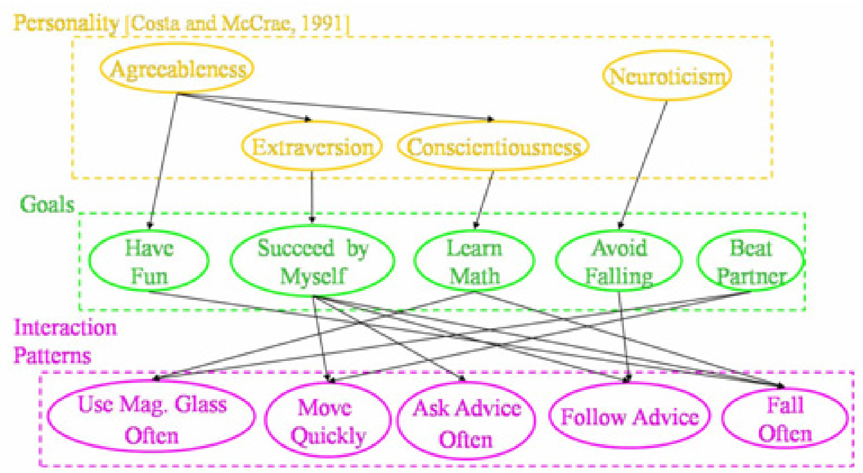

Fig. 14.7 Fragment of the goal assessment network in (Zhou and Conati 2003)

that relies on these goals as one of the elements to infer student emotions (Conati and Maclaren 2009). Arroyo and Wolf (2005) use a similar approach to learn the structure of the Bayesian network that relates interaction behaviors to user attitudes, mentioned in section 14.3.1.

14.5.2 Network Parameters

"Where do the parameters come from?" is arguably the first and most common objection that is raised in research that applies Bayesian networks to real world problems. As is the case for structure, the two main approaches to parameter specification are learning the parameters from data, or relying on domain experts to estimate them. Relying on expert judgment is costly and error prone. It is difficult for humans to commit to numbers their intuitions over given probabilistic dependencies. There has been substantial research on techniques that support the probability elicitation process (e.g., Keeney and von Winterfeldt 1991), but these techniques usually involve rather lengthy elicitation procedures and thus tend to be impractical when expert availability is limited. Still, when data is not available, relying on experts is the only viable approach and having conditional probabilities that are intuitive to specify can greatly facilitate parameter elicitation. In this section, will discuss one technique that can facilitate parameter specification by reducing the number of parameters to be specified, and two techniques for learning parameters from data

14.5.2.1 Parameters Reduction

One approach that can help reduce the effort of parameter specification is to reduce the number of parameters by approximating the necessary conditional probabilities via probabilistic variations of standard logic gates. This is the approach used by Andes to define the conditional probabilities in its task-specific networks.

Recall from section 14.3.1 that a task-specific network in Andes represents one or more solutions to a problem in terms of how each solution element derives from a physics rule and from the solution elements that are preconditions for rule application. Solution elements are either physics facts or problem solving goals, (collectively identified for convenience as *propositions* nodes PROP- in Fig. 14.8). Specific rule applications are represented in the network by rule application nodes (Rule-Appl nodes in Fig. 14.8).

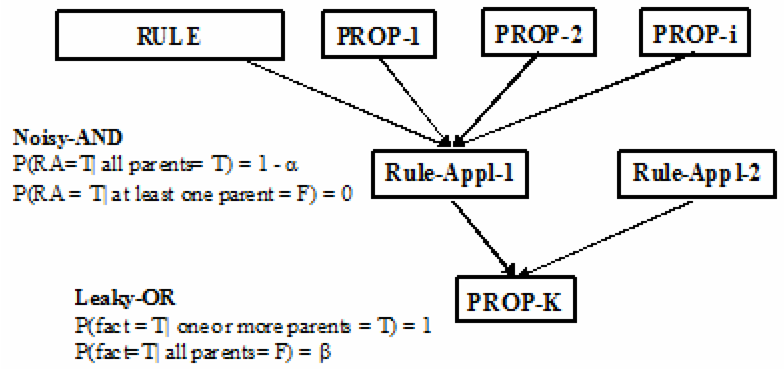
RULE
PROP-1
PROP-2
PROP-i
Noisy-AND
P(RA=T| all parents= T) = 1 - α
P(RA = T| at least one parent = F) = 0
Rule-Appl-1
Rule-Appl-2
Leaky-OR
P(fact = T| one or more parents = T) = 1
P(fact=T| all parents= F) = β
PROP-K

Fig. 14.8 probabilistic relations among rules, rule applications and their effects in Andes's task specific network

The parents of each Rule-application node include exactly rule, and a number of Proposition nodes corresponding to the rule's preconditions (see Fig. 14.8). A Rule-application node's value is *true* if the student has applied or can apply the corresponding rule to the propositions representing its preconditions, *false* otherwise. The probabilistic relationship between a Rule-application node and its parents is a Noisy-AND probabilistic gate (Henrion 1989). Here the Noisy-AND models the assumption that, in order to apply a rule, a student needs to know the rule and all its preconditions, although there is a non-zero probability α (the *noise* in the Noisy-AND), that the student will fail to apply the rule when s/he can, because of an error of distraction or some other form of *slip*. Thus, the α in Andes' Noisy-AND gates is an estimate of how likely it is that a student commits a slip, and it is the only parameter that needs to be specified to define the CPTs of rule-application nodes, regardless of how many parents they have.

Proposition nodes have as many parents (rule-application nodes) as there are ways to derive them. Thus, if there are two different rule applications that lead to the same solution element, then the corresponding Proposition node will have two parents (see Fig. 14.8). In Andes, the conditional probabilities between Proposition nodes and their parents are described by a Leaky-OR relationship (Henrion 1989), as shown in the lower part of Fig. 14.8. In a Leaky-OR relationship, a node is true if at least one of its parents is true, although there is a non-zero probability β of a "leak," that the node is true even when all the parents are false. This leak represents in Andes the probability that a student can derive a step via guessing or in some other way not represented in the network, and it is the only parameter that needs to be specified to define the CPTs of proposition nodes, regardless of how many alternative ways to derive a step the network encodes.

While the use of probabilistic logic gates in Andes greatly reduces the number of parameters that need to be specified, assessing the probability of a slip for each rule application and the probability of a guess for each solution element can still be a daunting task. The approach used in Andes follows a strategy that is often helpful when using Bayesian networks: make one or more simplifying assumptions that facilitate model definition and verify empirically whether the resulting model still yields an acceptable performance. The simplifying assumption made in Andes with respect to network parameters is that all slip and guess parameters are the same in the task-specific networks. Model adequacy was verified indirectly via empirical evaluations of the complete Andes system. The most extensive evaluation involved an experimental condition with 140 students using Andes for homework activities over the course of several weeks, and a control condition with 135 students doing homework without Andes. Students in the Andes condition scored significantly higher on a midterm exam covering relevant material. The accuracy of the Andes model was also analyzed directly by studying its performance in assessing the knowledge profile of simulated student (VanLehn and Niu 2001). This evaluation focused on performing *sensitivity analysis* on the Andes models to identify the factors that most impact model performance. The analysis revealed that the factor with the highest impact is, not surprisingly, the number of solution steps available as evidence to the model. I contrast, varying slip and guess parameters showed to have little effect on accuracy, confirming that the assumption of uniform slip and guess parameters was an acceptable one to make in light of the savings that it brings in effort for model specification.

14.5.2.2 Learning Parameters from Data

When all nodes in a Bayesian networks are observable, the entries for the network's CPTs can be learned via maximum-likelihood parameter estimation from frequency data (Russel and Norvig 2010). Unfortunately, in student modelling it is often the case that the variables of interest are not observable (e.g. student knowledge). Even when the variables are in theory observable (e.g., student goals, emotional states), in practice it can be very difficult to collect data on them,. Still, learning parameters from data is desirable because it eliminates the need to resort to the subjective judgment of experts. This judgement is not only hard to obtain and possibly fallacious, it can also be altogether unavailable when trying to model novel phenomena such as the relationships between student interaction with an ITS and student emotional states.

For this reason, there has been increasing interest in investigating how to exploit data-based techniques for parameters definition in student modeling. One approach, pioneered by Mayo and Mitrovic (2001), is to include in the student model only variables that are easily observable from interaction events with the tutoring system. In (Mayo and Mitrovic 2001) these variables model success or failure in using a variety of skills involved in correctly punctuating sentences. In particular, for each relevant skill, the Bayesian network in (Mayo and Mitrovic 2001) includes a variable representing the probability that a student will apply the skill correctly the next time it is relevant, given the outcome of the student's last attempt to apply the skill. The CPTs in the network were learned from log files of students solving punctuation problems in CAPIT, a tutoring system to help students practice punctuation skills. The network predictions are then used by CAPIT to automatically select new exercises for students, based on the criterion that a good exercise should involve several skills that the student has mastered and one or two skills that the student may still apply incorrectly. The idea of including in the student model only variables that are easily observable from interaction events obviously constraints the depth and sophistication of the inferences that an ITS can make about its students. However, Mayo and Mitrovic (2001) show that this approach is suitable and effective when the target instructional domain and interactions are of limited complexity.

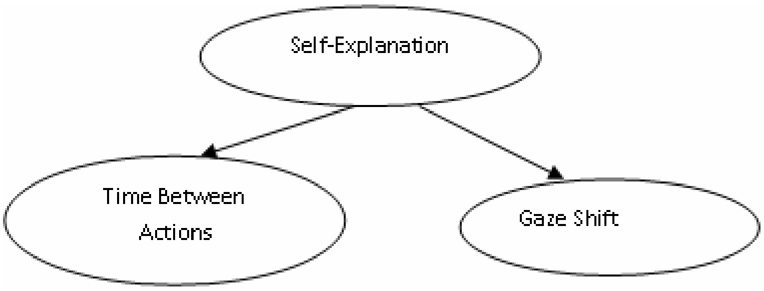

Fig. 14.9 Simple Bayesian network to predict self-explanation from action latency and gaze patterns in ACE

A second approach to learning the parameters of a student model from data relies on conducting empirical studies designed ad hoc to collect data on variables not observable from basic interaction events (we'll call these variables "hard-to-observe", to distinguish them from truly unobservable variables such as knowledge). For instance, Conati et al. (2005) conducted a study to collect data for a DBN that assesses student self-explanation behaviour from action latency and gaze patterns while the student is using an interactive simulation of mathematical functions. Self-explanation is the process of clarifying and elaborating instructional material to oneself, and it generally has a strong impact on learning (Chi 2000). In the context of studying interactive simulations, self-explanation relates to the effort a student makes to explain the effects of the manipulations performed on simulation parameters. The ACE system (Bunt et al. 2001; Conati and Merten 2007) aims to track student effort in self-explanation and provide adaptive interventions to increase this effort when needed. The study in (Conati et al. 2005) collected verbal protocols of students interacting with ACE, and analyzed these protocols to identify both episodes in which students generated self-explanations as well as episodes in which students failed to reason about the behaviour of the interactive simulation. These episodes where then matched to both log data on latency between student actions, as well as attention patterns over salient elements on the ACE interface, tracked via an eye-tracker. Frequencies from this dataset where then used to set the CPT of the simple Bayesian network shown in Fig. 14.9 (also known as a Naive Bayes classifier). A follow-up study showed that, when added to a more complex model of student learning, the network in Fig. 14.9 reliably supports the assessment of both student self-explanation and learning during interaction with ACE (Conati and Merten 2007). D'Mello et al. (2008) and Conati and Maclaren (2009) have adopted similar approaches relying on sophisticated data collection to build student models that assess student emotions from a variety of evidence sources.

In the research described above, it was not known upfront which observable factors could be good predictors of the hard-to-observe variables. Under these circumstances, in order to create a Bayesian student model researchers need to first find these predictors, which requires setting up experiments to collect data on the hard-to-observe variables. Once the data is collected and predictors are identified, everything is in place to apply standard maximum-likelihood parameter estimation. On the other hand, if there is an established dependency between the target hard-to-observe variables and a set of observable predictors, then the network parameters can be learned using EM (Dempster, et al. 1977). EM (which stands for Expectation-Maximization) is a class of algorithms that learn the parameters of a model with hidden variables by successive approximations based on two steps: the *expectation* step generates expected values of hidden variables from the current version of the model with approximated parameters; the *maximization* step refines the model parameters by performing maximum-likelihood parameter estimation using the expected values as if they were observed values. Thus, using EM removes the need for setting up complex studies to get values for hard-to-observe variables, when the dependency structure between these variables and a battery of observable variables is already known. Fergusson et al. (2006), for instance, used EM to learn the parameters of a Bayesian network that models knowledge of 12 geometry skills. In particular, EM was used to learn the dependencies between the variables representing this knowledge, and observable variables representing test questions designed specifically to assess the 12 target skills. The data for this work comes from a test that students in a Massachusetts high school had to take as part of a field study to evaluate the Wayang Outpost ITS for math.

Collecting sufficient amounts of data is the bottleneck in using any form of machine learning to specify a student model. It often requires setting up strong relationships with schools so that the necessary data can be collected as part of school activities involving whole classrooms. This process is generally very laborious. Schools, however, are becoming more and more willing to participate in these initiatives as the ITS field matures and produces concrete evidence of the benefits of having intelligent tutors available in the classroom, as it is shown by the increasing number of large scale school studies reported in ITS-related publications.

14.6 Discussion and Conclusions

Building a reliable picture of a student's relevant cognitive and affective states during learning is a task permeated with uncertainty that can be challenging even for experienced human tutors. Bayesian networks is a formalism for reasoning under uncertainty that has been successfully used in many AI applications, and that has been extensively used in student modeling, and user modeling in general. Critics of this approach mention the difficulty of reliably defining the model parameters (conditional probabilities) as one of its main drawbacks. An alternative approach for building a model of relevant student states would be to specify heuristic rules to define how available evidence should be integrated to assess the states. Defining these rules, however, still requires quantifying at some point complex probabilistic dependencies, because not explicitly using probabilities does not magically get rid of the uncertainty inherent to the modeling task. The advantage of a formal probabilistic approach is that the model only needs to quantify local dependencies among variables. The sound foundations of probability theory define how these dependencies are processed and affect the other variables in the model. In contrast, heuristic approaches require defining both the dependencies and ways to process them. This task is not necessarily simpler that defining conditional probabilities and entails a higher risk of building a model that generates unsound inferences. Furthermore, the Bayesian network graphical representation provides a compact and clear description of *all* the dependencies that exist in the domain, given the direct conditional dependencies encoded in the model. This helps to both verify that the postulated conditional dependencies define a coherent model, as well as debug the model when it generates inaccurate assessments. Similarly, the underlying network structure facilitates the process of generating automatic explanations of the results of probabilistic inference, making Bayesian networks very well suited for applications in which it is important that the user understands the rational underling the system behavior, as it is often the case for ITS (e.g., Zapata-Rivera and Greer 2004). Finally, Bayesian networks lend themselves well to support decision making approaches that rely on the sound foundations of decision theory. While decision theoretic approaches can still be too computationally expensive for handling complex tutorial interactions (e.g., Murray et al. 2004), researchers have shown their feasibility for dealing with particular pedagogical decisions in simpler domains, such as problem selection in sentence punctuation tasks (Mayo and Mitrovic 2001). Furthermore, continuous advances in reseach on decision theoretic planning suggest that more and more real world problems will be solvable with these approaches (see for instance the proceedings of *POMDP Practitioners Workshop: solving real-world* at http://users.isr.ist.utl.pt/~mtjspaan/POMDPPractioners/), including problems related to complex student modeling.

References

Arroyo, I., Woolf, B.: Inferring learning and attitudes from a Bayesian Network of log file data. In: 12th International Conference on Artificial Intelligence in Education, AIED 2005 (2005)

Bunt, A., Conati, C., Hugget, M., Muldner, K.: On Improving the Effectiveness of Open Learning Environments through Tailored Support for Exploration. In: 10th World Conference of Artificial Intelligence and Education, AIED 2001 (2001)

Buntine, W.: A Guide to the Literature on Learning Probabilistic Networks from Data. IEEE Transactions on Knowledge and Data Engineering 8(2), 195–210 (1996)

Chi, M.: Self-explaining: The dual processes of generating inference and repairing mental models. In: Glaser, R. (ed.) Advances in instructional psychology: Educational design and cognitive science, vol. (5), pp. 161–238. Lawrence Erlbaum Associates, Mahwah (2000)

Conati, C., Maclaren, H.: Empirically Building and Evaluating a Probabilistic Model of User Affect. Modeling and User-Adapted Interaction 19(3), 267–303 (2009)

Conati, C., Merten, C.: Eye-Tracking for User Modeling in Exploratory Learning Environments: an Empirical Evaluation. Knowledge Based Systems 20(6), 557–574 (2007)

Conati, C., Gertner, A., VanLehn, K.: Using Bayesian Networks to Manage Uncertainty in Student Modeling. Journal of User Modeling and User-Adapted Interaction 12(4), 371–417 (2002)

Conati, C., Merten, C., Muldner, K., Ternes, D.: Exploring Eye Tracking to Increase Bandwidth in User Modeling. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 357–366. Springer, Heidelberg (2005)

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

Costa, P., McRae, R.: Four ways five factors are. Personality and Individual Differences 13, 653–665 (1992)

Dean, T., Kanazawa, K.: A Model for REasoning About Persistence and Causation. Computational Intelligence 5(3), 142–150 (1989)

Dempster, A., Laird, N., Rubin, D.: Maximization-likelihood from Incomplete Data via the EM Algorithm. Journal of Royal Statistical Society, Series B (1977)

D'Mello, S., Craig, S., Witherspoon, A., McDaniel, B., Graesser, A.: Automatic detection of learner's affect from conversational cues. User Modeling and User-Adapted Interaction, 45–80 (2008)

Ferguson, K., Arroyo, Y., Mahadevan, S., Park Woolf, B., Barto, A.: Improving Intelligent Tutoring Systems: Using Expectation Maximization to Learn Student Skill Levels. In: Ikeda, M., Ashley, K.D., Chan, T.-W. (eds.) ITS 2006. LNCS, vol. 4053, pp. 453–462. Springer, Heidelberg (2006)

Henrion, M.: Some practical issues in constructing belief networks. In: 3rd Conference on Uncertainty in Artificial Intelligence, pp. 161–173 (1989)

Keeney, R.L., von Winterfeldt, D.: Eliciting probabilities from experts in complex technical problems. IEEE Transactions on Engineering Management 38, 191–201 (1991)

Martin, J., VanLehn, K.: Student assessment using Bayesian nets. International Journal of Human-Computer Studies 42, 575–591 (1995)

Mayo, M., Mitrovic, T.: Optimising ITS Behaviour with Bayesian Networks and Decision Theory. International Journal of Artificial Intelligence in Education 12, 124–153 (2001)

Mislevy, R.: Probability-based inference in cognitive diagnosis. In: Nichols, P., Chipman, S., Brennan, R. (eds.) Cognitive Diagnostic Assessment, pp. 43–71. Erlbaum, Hillsdale (1995)

Moore, A., Wong, W.: Optimal Reinsertion: A New Search Operator for Accelerated and More Accurate Bayesian Network Structure Learning. In: ICML 2003, pp. 552–559 (2003)

Murray, C., VanLehn, K., Mostov, J.: Looking Ahead to Select Tutorial Actions: A Decision-Theoretic Approach. International Journal of Artificial Intelligence in Education 14(3-4), 235–278 (2004)

Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo (1988)

Reye, J.: Two-phase updating of student models based on dynamic belief networks. In: Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V.J. (eds.) ITS 1998. LNCS, vol. 1452, pp. 274–283. Springer, Heidelberg (1998)

Russel, S., Norvig, P.: Artificial Intelligence - A Modern Approach, 3rd edn. Prentice Hall, Englewood Cliffs (2010)

VanLehn, K., Niu, Z.: Bayesian student modeling, user interfaces and feedback: A sensitivity analysis. International Journal of Artificial Intelligence in Education 12, 154–184 (2001)

Zapata-Rivera, D., Greer, J.: Interacting with Inspectable Bayesian Student Models. International Journal of Artificial Intelligence in Education 14(2), 127–163 (2004)

Zhou, X., Conati, C.: Inferring User Goals from Personality and Behavior in a Causal Model of User Affect. In: UI 2003, International Conference on Intelligent User Interfaces, pp. 211–281. ACM Press, New York (2003)