VanLehn, K. (2008). Intelligent tutoring systems for continuous, embedded assessment. In C. Dwyer (Ed.) The future of assessment: Shaping teaching and learning. Mahwah, NJ: Erbaum. pp. 113-138

# 5

# Intelligent Tutoring Systems for Continuous, Embedded Assessment

*Kurt VanLehn*
University of Pittsburgh

This chapter describes how intelligent tutoring systems (ITS) can be used for assessment. It begins by describing intelligent tutoring systems and distinguishing them from an older, more widely available technology. It argues that this newer technology has some intrinsic benefits as a tool for assessment. It then describes a particular type of assessment widely used in the ITS community. The assessment technique is based on a mathematically sound inference method, Bayesian belief networks. The basic technique, an adaptation of Knowledge Tracing (Corbett & Anderson, 1995), is described along with several recent extensions.

## An Introduction to Intelligent Tutoring Systems

There are many types of tutoring systems. The oldest and most commonly available technology has students enter the final answer to a question or problem, and gives them feedback and hints based on the answer (Dick & Carey, 1990). For instance, such a system might assign an algebra word problem and expect the student to solve it on scratch paper and enter a numerical answer. If the answer is correct, the system says so; otherwise, it may give a hint. This kind of tutoring system is often called computer-aided instruction (CAI), computer-based training (CBT), or more recently, Web-based homework (WBH).

Perhaps the second largest category of tutoring system has students enter steps leading up to the solution of a problem, and it can give feedback and hints on those steps as well as on the final answer

(VanLehn, 2006; Corbett, Koedinger, & Anderson, 1997; Rickel, 1989; Shute & Psotka, 1996). For instance, after assigning an algebra word problem, such a system would expect the student to enter a sequence of lines, each consisting of a variable definition or an equation. That is, the work that would otherwise be done on scratch paper is instead done on the computer. The system can give the student feedback and hints on the intermediate steps leading up to the final answer. These systems are called intelligent tutoring systems (ITS).

Although the defining feature of this kind of ITS is that it gives feedback and hints on steps as well as final answers, *when* it gives feedback and hints varies. Some ITS (e.g., Mitrovic & Ohlsson, 1999) delay giving feedback and hints on steps until the student has indicated that the solution is complete; this allows the student to practice finding his/her own errors and correcting them. However, many ITS are intended for novices who are still struggling to find any solution at all, so these systems (e.g., Anderson, Corbett, Koedinger, & Pelletier, 1995) give feedback on a step just as soon as it is entered; this prevents long, unproductive trial-and-error searches. In short, the distinguishing feature is not when the feedback and hints are given, but whether the system addresses only the final answer or the intermediate steps as well.

The term "ITS" encompasses systems that don't easily fit the description above, such as a tutoring system that converses with a student in spoken natural language (e.g., Litman et al., in press). However, the field does not yet have an accepted nomenclature for distinguishing the various subtypes of intelligent tutoring systems, so this chapter will use "ITS" to refer only to tutoring systems where a student receives either immediate or delayed feedback and hints on multiple steps that comprise a solution to a problem.

The term "step" will be used for the user interface entries that students are expected to make. Steps can take many forms. For illustration, consider a simulated laboratory task: determining if a particular chemical solution changes its acidity as it is heated. The student's steps might be to (1) get a clean test tube from the (simulated) storage cabinet, (2) decant some of the solution into it, (3) heat it to a specific temperature, (4) measure its acidity (pH), (5) create a blank line graph, (6) label the x-axis "temperature," (7) label the y-axis "acidity," (8) plot a point on the graph, (9) heat the solution some more, (10) plot a second point on the graph, and (11) enter "true" as the final answer. Each of these 11 user

interface gestures is a step. Let us suppose that the tutoring system is a delayed feedback system, so as soon as the student has entered "true," it gives feedback by suggesting that more than two data points are needed for reliability, and that "pH" would be a more accurate label for the y-axis than "acidity." Thus, its feedback and hints are at the level of steps.

For most ITS, the user interface is designed so that steps are entered frequently. That is, the student only has to think for a few seconds or perhaps a minute at most to decide how to do the next step. If a step is so complex that it tempts the student to use scratch paper, then the ITS designers might redesign the user interface to break down that step into substeps. However, the size of steps is an ITS design variable just like immediate/delayed feedback. More competent students should probably be given a user interface that requires larger, more complex steps, whereas novices probably learn best from a user interface with frequent, small, simple steps (Greer & McCalla, 1989).

Based on studies of human tutors (Douglas, 1991; Fox, 1993; Hume, Michael, Rovick, & Evens, 1996; Lepper, Woolverton, Mumme, & Gurtner, 1993; McArthur, Stasz, & Zmuidzinas, 1990; Merrill, Reiser, Ranney, & Trafton, 1992; VanLehn, Siler, Murray, Yamauchi, & Baggett, 2003; Wood & Middleton, 1975), ITS give feedback and hints in sequences of increasing specificity. Suppose a student enters an incorrect step into a tutoring system that gives feedback and hints immediately. The minimal-information feedback is simply to say, "that's incorrect." Systems often do this by turning the step red or beeping. This minimal-information response comprises the first "hint" in the system's sequence. If the student either asks for a hint or makes a second incorrect attempt at the step, the system gives the next hint in the sequence, which is usually designed to jog the student's memory or to point out some critical feature of the problem that the student may have overlooked. If the student again asks for a hint or enters the step incorrectly, the system gives the third hint in the sequence, which reveals further information. This continues until the student either enters the step correctly or the system reaches the last hint in its sequence. The last hint, which is called the "bottom out hint," tells the student exactly what to enter. Although most ITS use hint sequences exclusively, some use even more complicated hint dialogues for some steps (e.g., VanLehn, Lynch, Schultz, et al., 2005).

Some basic concepts of ITS have been introduced: A *task* is solved by entering many *steps* leading up to a *final answer.* The student can get *feedback* and *hints* on individual steps either *immediately* or *delayed* until they have submitted their final answer. The feedback and hints are often given in a *hint sequence* that ends with a *bottom-out hint*.

## *An Illustrative ITS*

In order to illustrate these concepts in an integrated fashion, let us use a college physics tutoring system, Andes (VanLehn, Lynch, Schultz, et al., 2005). Andes is in regular use at the U.S. Naval Academy (USNA). It is used by students in their dorm rooms to help them do their assigned homework. When Andes was adopted, all the other instructional activities in the course were almost entirely unaffected. In particular, the instructors continued to use the same textbooks, lectures, labs, and exams.

Indeed, the USNA course is taught by a team of instructors, and only some of them have adopted Andes. The other instructors' students still do homework on paper. This has allowed us to compare students who do homework on Andes with those who do homework on paper while holding all other instructional activities constant. The Andes students learned significantly more (effect size, .61).

Figure 5.1 shows the Andes screen, which is divided into four windows. (1) The upper left window shows the homework problem, which is often illustrated with a picture. A box for entering the final answer is also displayed. Below the picture is some white space in which the student can draw diagrams. In this case, the student has drawn a diagram with a body (the dot), a Cartesian coordinate system (tilted slightly), a vector pointing downward representing the weight force *Fw* on the car, and a vector pointing leftward representing the displacement *d* of the car. (2) The upper right window lists the variables the student has defined so far. Most were defined as side effects of drawing vectors, etc. (3) The lower right window is a list of equations that the student has defined so far. (4) The lower left window is for hint dialogues between the tutoring system ("T:…") and the student ("S:…").

There are several different kinds of steps. Entering an equation is a step, and it consists of clicking in a box in the lower right window,

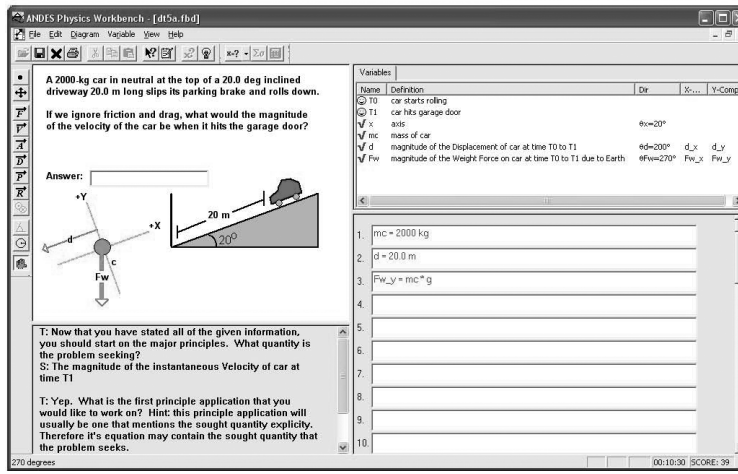


**Figure 5.1** User interface of the Andes physics tutoring system.

typing the equation, and pressing the Enter key. Defining a scalar variable is a step, and it is done by pulling down a menu, selecting the type of variable, and filling out a dialogue box that defines the variable precisely and names it. Drawing a vector is a step, and it is done by clicking on the vertical menu bar on the left side of the screen, dragging out a vector, then filling out a dialogue box to complete a precise definition of the vector. There are several other kinds of steps as well.

Whenever the student enters a step, it turns green if it is correct and red if incorrect. In the figure, only ~~one equation~~ is incorrect; all the other steps are correct. Andes will give hints only if asked, and students ask for a hint by clicking on a button in the top menu bar. In most cases, Andes uses a hint sequence. On a few especially important cases, it uses a more complex hint dialogue, wherein it asks the students a question and provides menus for answering it. A fragment of such a hint dialogue is shown in the lower left window of the figure.

## Why Use an ITS for Assessment?

There are several reasons for using ITS for assessment. This section lists them, dividing them into those that flow from the intrinsic operation of ITS, and those that arise from their routine use in instruction.

*Benefits Due to the Multistep Nature of ITS*

First, an ITS allows somewhat more authentic tasks to be used for assessment, and that may increase the validity of the assessment. Conventional assessment methods are like CAI tutoring systems in that they judge the student's work based only on the final answer. In order to get enough data on a student during a testing session, tasks need to be simple enough that many of them can be solved in an hour. An ITS gathers data on intermediate steps as well as the final answer, so it can get just as much data on a student by assigning just one or two tasks to be completed in an hour. It is often thought that short, simple tasks are not as authentic as complex ones, and that this harms their validity. Granted, a single student working with a computer for an hour or so may not be as authentic as a multi-person team working on a project over a semester, but moving from a 30-second task to a 30-minute task is at least a step toward authenticity and increased validity.

Second, an ITS can provide strategic and meta-cognitive assessments. Because an ITS monitors the steps leading up to a solution, an ITS can observe a student's problem-solving strategy, which is difficult or impossible to observe when only the final answer is entered. If the ITS includes reference information, such as a glossary, worked example problems, principles, or even the textbook itself, then it can monitor the student's use of this information as the student solves problems. This can reveal the student's meta-cognitive strategies for learning. Even the student's use of the ITS's hints can reveal their meta-cognitive strategies (Aleven & Koedinger, 2000). This allows an assessment to measure not only what a student knows, but how they use it and how they learn it.

Third, the consequential validity (Messick, 1994) of using an ITS for assessment is probably positive. It has been argued that conventional assessments based on final answers to many relatively short, inauthentic problems have caused students and instructors to focus too much on becoming proficient in this kind of problem solving. That is, the societal consequences of the assessment are negative. On the other hand, when students and instructors adapt their instruction in order to raise ITS scores, they are probably improving the student's meta-cognitive and problem-solving strategies. Thus, the consequential validity of an ITS assessment is probably more positive than that of conventional assessment.

### *Benefits of Embedded Assessment*

With a few exceptions, ITS have *not* been used to take over or replace classroom instruction. As mentioned earlier, the Andes physics tutoring system (VanLehn et al., 2005) only helps students do their homework; all other instructional activities are carried out by the instructors as usual. As another example, Carnegie Learning's popular Cognitive Algebra Tutor (www.carnegielearning.com) is used by high school students 2 days a week in computer labs; the other 3 days are regular algebra classes.

Let us assume that our assessment ITS is used only to coach students as they solve complex tasks, and that the course contains many other instructional activities besides problem solving. To put it in a slightly old fashioned way, we are assuming that the ITS grades students' homework and seatwork. This saves the instructor a great deal of time and effort, but can it really change classroom practices very much?

Unlike a human grader, the ITS is objective, indefatigable, and completely reliable. It is reliable in the sense that given the same log of student actions (i.e., all user interface events, in chronological order, with the same pauses), it will generate exactly the same assessment.[1]

Moreover, the assessment can be standardized, in that the same measures can be used in all classes. The assessment can be nationally or internationally normed. Norming is particularly easy because the ITS is viable as an instructional tool, which allows one to collect calibration data on student behavior over extended periods of time from many sites prior to releasing it as a normed assessment tool. Lastly, the ITS can send its assessments over a secure connection to an impartial judge, such as a testing company, as well as to the instructor and/or student. These are some of the same benefits that standardized testing achieves.

At first glance, it might appear that using an ITS for high-stakes assessment would require the same kind of authentication used at testing centers. When an ITS is used for homework or seatwork, the ITS cannot really tell who is at the keyboard. A student could log in and then have a friend sit down and solve the problems. However, even if an ITS is used for high-stakes decision making, it would be used to gather data over a whole year or semester. Only rarely would someone be willing to do a whole year of homework and/or seatwork for their friend and not get caught.

A much more important issue is that conventional testing is usually sequestered problem solving (Bransford & Schwartz, 1999), but ITS problem solving is not. That is, conventional testing is done in a situation where students cannot receive help, do not receive feedback on their answers, and in general are prevented from learning. However, an ITS has exactly the opposite goal—it tries to get students to learn. As will be described in the subsequent section, algorithms have been developed that can track students' evolving competence even when the tutoring system is giving them hints and helping them learn. However, the ITS cannot detect when the student is receiving help from a human. Clearly, we do not want to prevent ITS students from obtaining help from instructors, friends, and family whenever they need it. Thus, we need to know how often such help is likely to occur and how much impact it might have on the accuracy of the ITS assessment. Let us consider seatwork first, then homework.

Ethnographic studies of the Cognitive Tutors indicate that even though students doing seatwork get most of their help from the tutoring system, they still ask for help from the instructor (Schofield, 1995). However, the instructors and students report that their conversations are often mathematically deeper than they were before the tutoring system was adopted. The tutoring system takes care of the simple help requests, leaving the instructor to deal with the complex ones. Both instructors and students prefer this mode of interaction.

When an ITS is used for homework, it is harder to conduct the appropriate ethnographic studies to determine how it is really used, but it appears that ITS homework is like ITS seatwork in that the ITS deepens human help but does not eliminate it. The Andes instructors still get many visits from students during office hours, and we can draw a few inferences from their experiences (D. Treacy, personal communication, 2000–2005). The instructors report that Andes makes it much easier for them to diagnose the student's difficulties. When a student reports having trouble with a particular homework problem, the instructor downloads the student's (partial) solution from a server. If the partial solution doesn't immediately identify the student's difficulty, the instructor can replay the log data in order to find out what kinds of mistakes were made and erased, what kind of hints were given by Andes, etc. The instructor can also examine Andes's analysis of the student's work. For diagnosing the student's misconceptions, these new sources of data seem much more effective than the student's self-reports.

Although more ethnographic studies are clearly needed, it appears that doing seatwork and homework with an ITS deepens human help but certainly does not eliminate it. Thus, an issue for future research, which can probably only be addressed by field studies, is determining if the existing ITS assessment algorithms gracefully accept sharp changes in competence due to human help.

Assuming that the existing assessment algorithms suffice (or that new ones do), we have a mode of assessment that does not require sequestered problem solving. Students are expected and encouraged to learn as they are being assessed, and they are working on complex, more authentic problems. Moreover, the assessments can be standardized, normed, and secured just like conventional tests are now. This suggests that standardized tests may be superfluous if an ITS is used to grade seatwork and homework. If so, then a major advantage of ITS assessment is that it would eliminate sequestered testing.

Elimination of sequestered, standardized testing in favor of continuous, embedded assessment could bring several practical advantages.

- It would free up the time currently used on taking tests that are deliberately designed to prevent learning during the test.
- Students could not be absent on the day of the test, because with ITS assessment, almost every day is a "test day."
- Students' assessed performance would not be atypically higher or lower than their "ordinary" performance, because the ITS measures their ordinary performance. It would no longer make sense to say that a student "doesn't test well" or "is good at tests, but nothing else."
- Test anxiety would disappear (or be indistinguishable from general anxiety).
- Cramming for the test no longer makes sense.
- Instead of measuring students' competence a few times a year, instructors could track students' knowledge growth continuously. For instance, if the instructor wants to create pairs comprised of a knowledgeable student and less knowledgeable student for tomorrow's discussions, the instructor can read the student's current state of knowledge today. The instructor can also detect more rapidly those students who are falling behind and those that are not sufficiently challenged.

However, before deciding to eliminate conventional testing, it needs to be repeated that an ITS measures performance during only part of the student's instructional activities. Students may also participate in

class discussions, engage in group projects, write essays, or give presentations. The assessment derived from an ITS should be combined with other measures in order to get a complete and well-rounded picture of the student's evolving strengths and weaknesses.

## Bayesian Assessment of Knowledge Component Mastery

Having described ITS and their potential benefits as assessments, this section defines a particular kind of assessment, reflected in the section's title, and argues for its utility. In order to define this kind of assessment, a few technical terms need to be introduced.

The first technical term is "knowledge component," which has its roots deep in the history of ITS and artificial intelligence (AI) more generally. After a few decades of working with relatively general reasoning methods, AI researchers realized that effective problem solving requires large quantities of domain-specific knowledge. Knowledge was represented in various notations, including production rules, frames, Prolog clauses, and so on. The knowledge was operational, in that it was specific enough to be used to solve problems. It was also represented in many small pieces, such as rules, frames, etc. The term *knowledge component* has emerged as a notation-neutral term for a small piece of operational domain knowledge.

When used in the context of an ITS, knowledge component denotes a small piece of domain knowledge *that the student should learn*. For instance, in late 2005, Andes contained 308 knowledge components and covered about 75% of the AP Physics B curriculum. A typical textbook for that curriculum has about 35 chapters, so this works out to about 12 knowledge components per chapter. Most of the knowledge components are well-known principles, such as Newton's second law or Ohm's law. However, some knowledge components are needed for reasoning but do not have names. For instance, in electrical circuits, there is a knowledge component that says that the current through a switch is zero if the switch is open, and the voltage drop across a switch is zero if the switch is closed. This knowledge component is mentioned in the textbook, but is not treated as a "named" principle like Ohm's law.

The second technical term that needs to be defined is the set of knowledge components that are *relevant* to a step. Suppose a student is solving a problem and takes a correct step. The knowledge components relevant

to the step are those that an ideally competent student would need to apply in order to generate that step in that context. For instance, suppose a physics problem is partially solved, as in Figure 5.1, and the student draws a zero-length vector representing the car's initial velocity. There is just one relevant knowledge component: "If an object is at rest at a certain time, then its instantaneous velocity at that time is zero." When an ITS is intended to be used by novice learners, then the number of relevant knowledge components per step is usually small—under five, typically, and often just one. When the ITS is intended for use by more competent students, then the number of relevant knowledge components per step is often somewhat larger—perhaps five or 10. As mentioned earlier, the average step size is an important ITS design parameter, similar to immediate vs. delayed feedback.

The third technical term that needs definition is "step attempt history." Because the ITS can give feedback and hints on steps, students sometimes make several attempts before successfully entering a step. In the earlier illustration of a simulated chemistry lab, instead of simply labeling the x-axis with "temperature," the following events might occur:

1. The student labels the x-axis "acidity."
2. The system's speech generator says, "OK, but I wouldn't choose axes that way."
3. The student clicks on the Hint button.
4. The system's speech generator says, "The question asks whether acidity varies as temperature is varied. So temperature is the independent variable."
5. The student clicks on the Hint button again.
6. The system says, "It is conventional to put the independent variable on the x-axis, so I would advise putting 'temperature' on the x-axis and 'acidity' on the y-axis."
7. The student enters "temperature" as the x-axis label.

In other words, it took seven user interface events to get this step entered correctly. When tutoring systems are used for assessment, it matters whether a step was entered correctly on the first try or only after receiving several hints. For future reference, let us use "step attempt history" to refer to the sequence of user interface events leading up to the student's final entry of the step.[2]

The last technical term that needs definition is *mastery* of a knowledge component. ITS assessment was often developed with mastery

learning in mind, and some ITS actually do implement mastery learning (Bloom, 1984). That is, they choose the tasks that the student should do, and they keep assigning tasks from an instructional unit until the student has mastered all the unit's knowledge components. However, mastery learning causes students to proceed at different paces. After a few weeks, some students may be in chapter 10 while others are still struggling with chapter 3. Thus, instructors often turn mastery learning off, and assign all students the same tasks. Nonetheless, the concept "mastery of a knowledge component" has been retained and is the last of our technical terms.

The concept is actually quite subtle. Intuitively, mastery means that the knowledge component is known so well that the instructor and ITS no longer need to focus on teaching it. However, this intuitive definition refers to instructional policy (how the mastery judgment is used) rather than the evidence used to define or infer mastery. As evidence of mastery, one intuitively expects the following:

- The knowledge component is always applied in situations where it should be applied.
- The knowledge component is never applied in situations where it should not be applied.
- When the knowledge component is applied, the application is done rapidly and without reference to external representations of the knowledge component (e.g., textbook, friend).
- The student can explain the knowledge component in general terms, and can explain why a particular application (or non-application) was justified.

Even an ITS doesn't have enough information to assess all these characteristics, so it must estimate mastery from the information that it has. Thus, an ITS should calculate only a probability of mastery—given the evidence observed by the ITS, what are the chances that this student has mastered this knowledge component?

It makes sense to adopt a Bayesian approach to assessment of mastery of knowledge components. Many courses build on prior knowledge, so some of the knowledge components used in the course may be mastered by students before they take the course. Of course, the ITS cannot just assume that all students will have mastered all the prerequisites. Some students must learn the prerequisites as they learn everything else. Thus, the ITS must estimate mastery of the prerequisite knowledge components along with the others. Its estimates can converge more rapidly on

accurate values if it assumes a fairly high prior probability of mastery for prerequisite knowledge components, and a fairly low prior probability of mastery for knowledge components that are rarely taught in preceding courses. If it has information about a specific student's background (e.g., courses taken), then the student's prior probabilities can perhaps be made more accurate.

At this point, the assessment problem can be stated: Given a set of knowledge components, each with a prior probability of mastery, and given the step attempt history for each step taken by the student while solving a sequence of tasks, calculate the posterior probability of mastery—that is, the probability of mastery of each knowledge component after all the tasks have been completed.

This is clearly not the only kind of assessment that could be computed by an ITS, and it might not even be the most useful one, especially for human decision makers. However, it can be used to calculate several useful kinds of assessments. Some examples follow.

> Suppose an instructor wishes to see a single number that represents "how well the student is doing." Such an assessment can be calculated by simply counting the number of knowledge components that have mastery probabilities greater than .90 (or whatever threshold the instructor chooses).
>
> Suppose an instructor wishes to see what concepts in the current chapter are difficult for students to learn. A nice display would be a bar chart with the chapter's knowledge components on the x-axis, the height of the bars corresponding to the probability of mastery of the component averaged across students in the class, and error whiskers showing the standard error of the mean. Low bars or bars with large whiskers indicate concepts causing students difficulty.
>
> Suppose an instructor (or a student) wants to predict how well the student would do on a certain standardized test (AP Physics B, in the case of Andes). Moreover, suppose the test items have been analyzed in order to determine, for each test item, which knowledge components are relevant. Given the student's posterior probability of mastery of each knowledge component, one can calculate for each test item the probability of a correct answer. This predicts not only the score on the test, but the pattern of correct and incorrect answers. This method has also been used to test the validity of ITS assessments described below (e.g., Corbett, McLaughlin, & Scarpinatto, 2000; VanLehn & Martin, 1998).

## A Synthesis of Bayesian Assessments

This section frames the assessment problem as construction and evaluation of a Bayesian network (Russell & Norvig, 2003). Several existing techniques are special cases of this framework, including one

of the earliest and arguably the most widely used methods, Knowledge Tracing (Corbett & Anderson, 1995), which is described first. Next, several extensions that overcome the limitations of Knowledge Tracing are described. These extensions synthesize several recent approaches to Bayesian assessment. The resulting system is one view of the "state of the art" in Bayesian assessment.

### *A Generic Dynamic Bayesian Network Assessor*

The assessment problem can be formulated as follows: Given (1) a set of knowledge components each with a prior probability of mastery, and (2) a chronological sequence of M steps along with the student's step attempt history for each, calculate (3) the posterior probabilities of mastery for each of the knowledge components. Put more simply, the assessment problem consists of updating a set of mastery probabilities based on the student's solution process, as revealed in their step attempt histories.

Suppose, for instance, that the problem is so simple that it can be solved in just two steps. If the student did one step successfully on the first attempt but had to be given a bottom-out hint on the other, then some probabilities of mastery would rise (for knowledge components relevant to only the successful step) and some probabilities would fall (for knowledge components relevant to only the unsuccessful step).

A Bayesian network is built from nodes and directed links, where the nodes represent random variables and the links represent conditional dependencies among them. Let us use nodes to represent the mastery of knowledge components, and let each node have two values: mastered or not. A knowledge component's mastery can change over time, so we need to use two subscripts to denote a particular mastery: Let P(Kij) denote the probability of mastery of knowledge component i just after step j. Let Ki0 denote the prior probability of mastery of the knowledge component i before any steps have been done. If we have N knowledge components and the student's solution comprised M steps, then we need N(M + 1) nodes.

We also need a random variable for each step in order to represent how the student accomplished the step. Thus, if there are M steps in the student's solution of the problem, then we need M step nodes. Now we have a representational choice to make. We cannot feasibly

represent all possible step attempt histories; there are just too many of them. Thus, the common practice is to define just a few *categories* of step attempt histories, such as (a) entered a correct step on the first try; (b) entered a correct step after seeing the bottom-out hint; (c) all other step attempt histories. With this classification, each step node would have three values. Another popular representation is the binary classification: (a) entered a correct step on the first try; (b) all other step attempt histories.

Next we need assumptions about conditional independence. This establishes the links among the nodes in the Bayesian network. Let us start with the links shown in Figure 5.2. This network topology makes two rather plausible assumptions.

1. The step attempt history depends only on the state of the student's knowledge just prior to the step. Thus, the links *into* a step node (e.g., S2) come only from a subset of the knowledge nodes, namely, those representing the knowledge components' mastery just before the step (e.g., K11, K21, K31, …).
2. Student knowledge just after a step is determined by their knowledge just before the step and what happened during the step, such as whether they got hints on that step. Thus, the links *into* a knowledge component node come from the immediately preceding step and from the node representing its mastery just before the step.

The usual way to update a Bayesian network is to clamp the values of the nodes that are observable, update the network, and read out the posterior probabilities on the nodes of interest. In this case, the step attempt histories are observable, so we clamp each of the M step

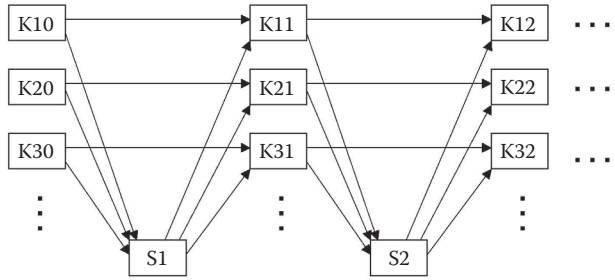


**Figure 5.2** A generic approach to assessment, represented as a Bayesian network.

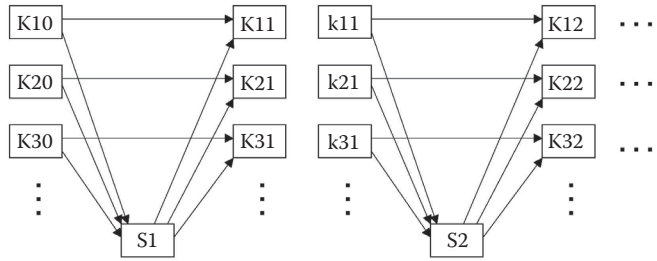


**Figure 5.3** Two slices from a generic dynamic Bayesian network for assessment.

node values (e.g., on S1, clamp the value to "Correct on first attempt"; on S2, clamp the value to "Saw bottom-out hint"; etc.). The network is updated, and we read out the posterior probability of mastery on the nodes on the right end, namely, K1M, K2M, K3M, etc.

However, this network would be so large that the computation required to update this network may be intractable. Thus, we use *dynamic* Bayesian networks (Russell & Norvig, 2003). The key idea is to divide the network into time slices, as shown in Figure 5.3. A time slice consists of a step and the knowledge component nodes just before and after it. Slices are updated in chronological order. After a slice has been updated, the *posterior* probabilities on the poststep knowledge component nodes (which represent only the marginal probabilities of those random variables) are copied to the next time slice, where they become the *prior* probabilities on the prestep knowledge component nodes. In Figure 5.3, the posterior probabilities of K11, K21, K31,… become the prior probabilities of k11, k21, k31,…. This method means that only relatively small networks need to be updated. It is somewhat inaccurate, however, because it loses probabilistic dependences among nodes. Dynamic Bayesian networks are used in many applications, and seem to perform well despite their inaccuracies.

### *Knowledge Tracing*

A widely used method of assessment, called Knowledge Tracing (Corbett & Anderson, 1995), is a special case of the generic network discussed in the preceding section (Reye, 2004). It makes several simplifying assumptions that result in networks such as the one shown in Figure 5.4.

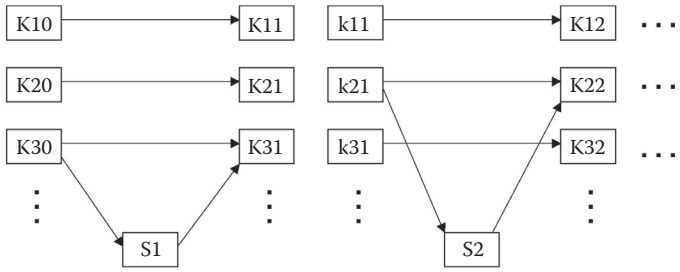


**Figure 5.4** Knowledge tracing: Two slices from a dynamic Bayesian network.

First, Knowledge Tracing assumes that each step depends on applying only one knowledge component. That is, the set of knowledge components relevant to a step (as defined in a preceding section) is always a singleton set. To represent this in the network, every step node has just one link coming into it, namely, a link from the one and only knowledge component that should be applied when making that step. In Figure 5.4, step 1 depends only on knowledge component 3, and step 2 depends only on knowledge component 2.

Second, it is assumed that what happens during a step impacts the mastery of only the relevant knowledge component. Thus, there is only one link leaving each node, and it goes to the relevant knowledge component in the time slice just after the step.

Every node in a Bayesian network has a conditional probability table that represents how the value of that node depends on the values of the parent nodes. (When a link comes *into* a node, the node on the other end of the link is called a "parent node.") In knowledge tracing, these conditional probability tables are particularly simple.

For the step nodes, the conditional probability table is shown in Figure 5.5. Knowledge tracing uses a binary categorization of step attempt histories: either the student entered the step correctly on the first try or not. Thus, the P(correct) row represents the probability of a correct response on the first attempt, and the P(other) row represents the probability of any other kind of step attempt history. The columns refer to the values of the parent, which is a knowledge component node. In the "Yes" column, the value .92 represents that if the knowledge

| Mastered? | Yes | No |
|---|---|---|
| P(correct) | 0.92 | 0.10 |
| P(other) | 0.08 | 0.90 |

**Figure 5.5** Conditional probability table for a step node in Knowledge Tracing.

component is mastered, there is a high probability that the step will be answered correctly, but there is a small probability (.08; called the "slip parameter") that the student will answer incorrectly even though the knowledge is mastered. In the "No" column, the value .10 (called the "guess parameter") represents that there is a small chance of getting the step right on the first try even if the relevant knowledge component is not mastered, and a large chance (.90) that the step attempt history will include errors, hints, etc. The columns must each sum to 1.0, so this table only has two degrees of freedom: the slip parameter and the guess parameter. Different step nodes can have different values for these parameters.

| Mastered? | Yes | No |
|---|---|---|
| P(mastered) | 1.00 | 0 |
| P(not mastered) | 0 | 1.00 |

**Figure 5.6** Conditional probabilities for a knowledge component node with one parent.

Some knowledge components are not relevant to the preceding step, so they have only one link coming into them, as shown earlier in Figure 5.4. Knowledge Tracing assumes that such a knowledge component's probability of mastery is unaffected by the student's step attempt history. Thus, all knowledge component nodes with just one parent have the conditional probability table shown in Figure 5.6.

The other knowledge components have two links coming into them, from the preceding step and from an earlier knowledge component node. This represents the fact that their new probability of mastery depends on both the step attempt history and their old probability of mastery. For such nodes, the conditional probability table is shown in Figure 5.7. The top row represents the impact of the knowl-

| Was mastered? | Yes | | No | |
|---|---|---|---|---|
| Step response | Correct | Other | Correct | Other |
| P(mastered) | 1.00 | 1.00 | 0.56 | 0.56 |
| P(not mastered) | 0 | 0 | 0.64 | 0.64 |

**Figure 5.7** Conditional probabilities for a knowledge component node with two parents.

| P(mastered) | 0.21 |
|---|---|
| P(not mastered) | 0.79 |

**Figure 5.8** Conditional probabilities for a knowledge component node with no parents.

edge component's prior mastery. The next row represents what happened during the preceding step. The bottom two rows represent the probability of this nodes' mastery given the various combinations of the parent node values. The numbers say that if the knowledge component was mastered, then it certainly (probability 1.0) will still be mastered. On the other hand, if the knowledge component was not mastered, then there is a moderate chance (.56) that it will become mastered regardless of what happened during the step. (As discussed later, this assumption of Knowledge Tracing was modified in subsequent work.) There is only one degree of freedom in this table, namely, the probability that an unmastered knowledge component will become mastered given that the step occurred. This is called the "acquisition parameter" and has the value .56 in Figure 5.7. The acquisition parameter can be different for different knowledge components.

The leftmost knowledge component nodes (Ki0) have no incoming links. Their conditional probability tables represent the prior probability of mastery of the knowledge component. For instance, the table in Figure 5.8 indicates that this knowledge component has a prior probability of .21 of being mastered.

For each knowledge component, we need four numbers: the guess parameter, the slip parameter, the acquisition parameter, and the prior probability of mastery.[3] Thus, calibration of the model requires estimating 4(N) values. This can be done given enough student data. Corbett and Anderson (1995) used a gradient descent algorithm to calibrate their networks from student data, but it appears that Expectation Maximization does an even better job (Chang et al., 2006).

### *Extensions to Knowledge Tracing*

Several projects have explored extensions to Knowledge Tracing that relax some of its less plausible assumptions. This section discusses their basic ideas, but recasts them in terms of the generic dynamic Bayesian network.

The Knowledge Tracing approach ignores considerable information when it categorizes step attempt histories as either correct or other. Intuitively, one would expect a student who makes an error and corrects it with just a mild hint to have a higher probability of mastery than one who makes errors after every hint and only succeeds after seeing the bottom-out hint. Shute (1995) developed an assessment technique that used a larger number of categories. Although it was not compared directly with Knowledge Tracing, its accuracy at predicting posttest scores was quite high. The basic idea of this approach can be easily represented in our dynamic Bayesian network by using multiple categories of step attempt histories instead of the two categories used by Knowledge Tracing. This adds extra rows to the conditional probability tables of the step nodes and extra columns to the conditional probability tables of the knowledge nodes that have multiple incoming links.

It also makes sense to assume that what happens during a step may partially determine whether an unmastered knowledge component becomes mastered. For instance, if a student zooms all the way down to the bottom-out hint without bothering to read the intervening hints, then it seems unlikely that the student will learn the relevant knowledge component. This particular step attempt history is often called help abuse (Aleven & Koedinger, 2000; Aleven, Stahl, Schworm, Fischer, & Wallace, 2003). We can represent this by including help abuse as a step attempt history category, and modifying the relevant knowledge component nodes' conditional probability tables as shown in Figure 5.9. This table assumes that step attempt histories are categorized three ways: correct on the first attempt, help abuse, and other. It assumes that if the knowledge component was not mastered before and help abuse occurs, then the knowledge component

| Was mastered? | Yes | | | No | | |
|---|---|---|---|---|---|---|
| Step response | Correct | Help abuse | Other | Correct | Help abuse | Other |
| P(mastered) | 1.00 | 1.00 | 1.00 | 0.56 | 0 | 0.56 |
| P(not mastered) | 0 | 0 | 0 | 0.64 | 1.00 | 0.64 |

**Figure 5.9** Conditional probabilities for a two-parent knowledge component node when step attempt histories have three categories.

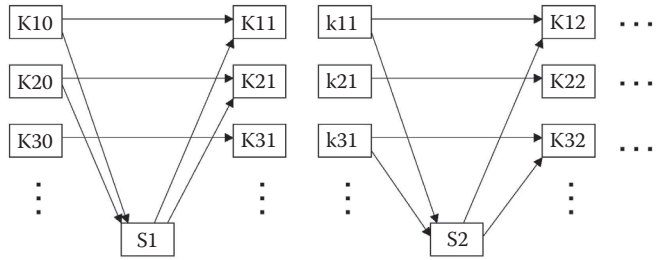


**Figure 5.10** A step can have more than one relevant knowledge component.

stays unmastered. Otherwise, it makes the same assumptions as Knowledge Tracing. The point here is only that the formalism allows one to easily represent step → mastery causation that has been uncovered from observation, theory, or intuition.

In the first version of the Andes tutoring system (Conati, Gertner, & VanLehn, 2002), a Bayesian network was used for assessment in a way that extended knowledge tracing. Whereas Knowledge Tracing assumes that there is one and only one knowledge component relevant to each step, this assumption was unwarranted in Andes. Some of the steps in Andes required a student to apply several knowledge components. This can be easily represented by extending the network as shown in Figure 5.10, which shows that knowledge components 1 and 2 are relevant to step 1, whereas knowledge components 1 and 3 are relevant to step 2.

As part of a decision theoretic tutoring system (Murray, VanLehn, & Mostow, 2004), Murray added nodes that represented meta-cognitive strategies and motivation. In particular, observation of students using tutoring systems suggests that some students exhibit a help abuse pattern at every opportunity, whereas others rarely do so (Aleven & Koedinger, 2000). This may be related to a well-known motivation variable (Dweck, 1986), which is whether students are trying to learn the domain (a learning orientation) or merely to get their problems solved correctly and quickly (a performance orientation). Such predilections can be represented by adding a set of nodes to the dynamic network, which we might call M0, M1, M2,… as at the top of Figure 5.11. These nodes might have two values: learning-orientation and performance-orientation. They are linked to the steps because the students' motivation may affect how they respond at the step, e.g., whether they will display the help abuse pattern. On

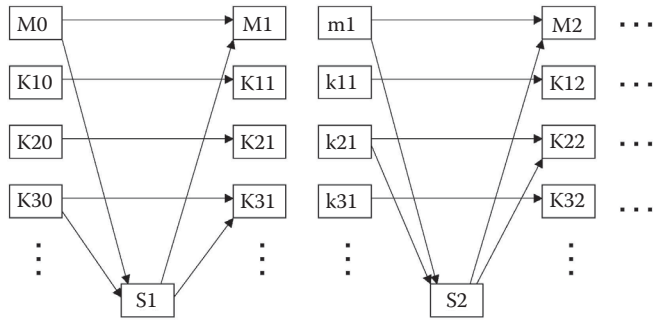


**Figure 5.11** Nodes (M0, M1, M2,…) can be added to represent motivation.

the other hand, the way the step plays out may also cause students to change their motivation, so there are links from the steps to the motivation nodes. Zhou and Conati (2003) have explored a similar technique for diagnosing motivation and meta-cognition.

All these extensions to Knowledge Tracing add parameters whose values need to be determined. When a new kind of node is added (e.g., motivation nodes), they have conditional probability tables whose cells need values. When new step attempt history categories are added, they too introduce new cells into the conditional probability tables that need to have values determined for them. Little work has been done on finding calibration techniques that take advantage of the particular topology of these networks. This may be a good topic for future work. Once good calibration algorithms are in use, we can compare these various extensions to Knowledge Tracing and/or other assessment methods in order to see if they actually provide added diagnostic value.

## Conclusions

This chapter introduced a particular kind of intelligent tutoring system that helps students solve multistep problems. The key idea is to give feedback and hints at the level of individual steps, instead of on the final answer alone. Although these systems were developed in order to increase student learning rate (and they are demonstrably good at that), they also provide the opportunity to assess student performance. The key assessment idea, as explained in the last section, is to assess individual knowledge components' probability of

mastery, and to use the fact that each step requires knowing only a few knowledge components in order to do the step correctly.

An analogy might help. A step is like one item on a multiple-choice diagnostic test, where the test items were designed to tap only a few knowledge components each. When analyzing data from such a test, one assumes that performance on a test item is conditionally independent of the performance on the preceding items given the mastery of the relevant knowledge components, that is, those involved in answering the test item. If one knows the probability of mastery of the relevant knowledge components just before the test item is answered, one's predictions cannot be improved by also knowing their performance on earlier test items. The same Markov assumption is made when interpreting step attempt history data, except that one must allow the knowledge components to change their mastery over time. That is, students are assumed to *learn* as they use the tutoring system, whereas they are assumed *not to learn* during a test. As shown in this paper, learning complicates the data analysis, but not by much.

This sets the stage for increasing the use of assessment embedded in a tutoring system. This may have desirable impacts on schools—it may cause them to teach to a "test" that really deserves to be taught to. It may free up classroom time currently taken up with testing that, by design, does not cause learning. As shown late in the last section, the technique can be used to diagnose motivation and meta-cognitive variables as well as cognitive ones. This technology is so radically different from conventional non-learning testing that it may even have many unanticipated benefits as well, and perhaps some unanticipated drawbacks.

## Acknowledgments

This research was supported by the Cognitive Science division of the Office of Naval Research under grant N00014-96-1-0260; and by the Pittsburgh Science of Learning Center (NSF 0354420). I thank Chas Murray for commenting on the manuscript.

## References

Aleven, V., & Koedinger, K. R. (2000). Limitations of student control: Do students know when they need help? In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Intelligent Tutoring Systems: 5th International Conference, ITS 2000* (pp. 292-303). Berlin: Springer-Verlag.

Aleven, V., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. M. (2003). Help seeking and help design in interactive learning environments. *Review of Educational Research, 73*(2), 277-320.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*(2), 167-207.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*(6), 4-16.

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. In A. Iran-Nejad & P. D. Pearson (Eds.), *Review of research in education* (Vol. 24, pp. 61-100). Washington, DC: American Educational Research Association.

Chang, K.-m., Beck, J., Mostow, J., & Corbett, A. (2006). A Bayes net toolkit for student modeling in intelligent tutoring systems. In K. Ashley & M. Ikeda (Eds.), *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. Berlin: Springer-Verlag.

Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interactions, 12*(4), 371-417.

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

Corbett, A., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring systems. In M. Helander, T. K. Landauer, & P. Prahu (Eds.), *Handbook of human-computer interaction* (2nd ed., pp. 849-874). Amsterdam: Elsevier Science.

Corbett, A., McLaughlin, M., & Scarpinatto, K. C. (2000). Modeling student knowledge: Cognitive tutors in high school and college. *User Modeling and User-Adapted Interactions, 10*, 81-108.

Dick, W., & Carey, S. (1990). *The systematic design of instruction* (3rd ed.). New York: Scott-Foresman.

Douglas, S. A. (1991). Tutoring as interaction: Detecting and repairing tutoring failures. In P. Goodyear (Ed.), *Teaching knowledge and intelligent tutoring* (pp. 123-147). Hillsdale, NJ: Lawrence Erlbaum Associates.

Dweck, C. S. (1986). Motivational processes affecting learning. *American Psychologist, 41*, 1040-1048.

Fox, B. A. (1993). *The human tutorial dialogue project: Issues in the design of instructional systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Greer, J., & McCalla, G. (1989). A computational framework for granularity and its application to educational diagnosis. In *International Joint Conference on Artificial Intelligence* (pp. 477-482). Menlo Park, CA: AAAI Press.

Hume, G., Michael, J., Rovick, A., & Evens, M. (1996). Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences, 5*(1), 23-49.

Lepper, M. R., Woolverton, M., Mumme, D. L., & Gurtner, J.-L. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as cognitive tools* (pp. 75-105). Hillsdale, NJ: Lawrence Erlbaum Associates.

Litman, D., Rose, C., Forbes-Riley, K., VanLehn, K., Bhembe, D., & Silliman, S. (in press). Spoken versus typed human and computer dialogue. *International Journal of Artificial Intelligence and Education*.

McArthur, D., Stasz, C., & Zmuidzinas, M. (1990). Tutoring techniques in algebra. *Cognition and Instruction, 7*(3), 197-244.

Merrill, D. C., Reiser, B. J., Ranney, M., & Trafton, J. G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *Journal of the Learning Sciences, 2*(3), 277-306.

Messick, S. (1994). The interplay of evidence and consequences in the validation of performance assessments. *Educational Researcher, 23*(2), 13-23.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence and Education, 10*, 238-256.

Murray, C., VanLehn, K., & Mostow, J. (2004). Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence and Education, 14*(3-4), 235-278.

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

Rickel, J. (1989). Intelligent computer-aided instruction: A survey organized around system components. *IEEE Transactions on Systems, Man and Cybernetics, 19*(1), 40-57.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A modern approach* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Schofield, J. W. (1995). *Computers, classroom culture and change*. Cambridge, UK: Cambridge University Press.

Shute, V. J. (1995). SMART: Student modeling approach for responsive tutoring. *User Modeling and User-Adapted Instruction, 5*(1), 1-44.

Shute, V. J., & Psotka, J. (1996). Intelligent tutoring systems: Past, present and future. In D. Jonassen (Ed.), *Handbook of research on educational communications and technology* (pp. 570-600). New York: Macmillan.

VanLehn, K., Lynch, C., Schultz, K., Shapiro, J. A., Shelby, R. H., Taylor, L., et al. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence and Education, 15*(3), 147-204.

VanLehn, K., & Martin, J. (1998). Evaluation of an assessment system based on Bayesian student modeling. *International Journal of Artificial Intelligence in Education, 8*(2), 179-221.

VanLehn, K., Siler, S., Murray, C., Yamauchi, T., & Baggett, W. B. (2003). Human tutoring: Why do only some events cause learning? *Cognition and Instruction, 21*(3), 209-249.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16.

Wood, D. J., & Middleton, D. (1975). A study of assisted problem-solving. *British Journal of Psychology, 66*(2), 181-191.

Zhou, X., & Conati, C. (2003). Inferring user goals from personality and behavior in a causal model of user affect. In *Proceedings of IUI 2003, international conference on intelligent user interfaces* (pp. 211-218). Miami, FL.

## Endnotes

1. The assessment technique described later makes the assessment a deterministic function of both the log data and a set of prior probabilities, which describe what is known about the student prior to the problem solving.
2. This definition is deliberately vague, because more precise definitions can only be made in the context of specific tutoring systems. For instance, some tutoring systems require students to enter a step correctly before they can even start working on a new step, whereas others allow students to leave steps incorrect and to change steps that were entered much earlier.
3. In practice, steps that have the same relevant knowledge component are often given the same values for their guess and slip parameters. For the purpose of counting parameters, it is more accurate to count one guess and slip parameter per knowledge component rather than per step.