# Bits: a Bayesian Intelligent Tutoring System For Computer Programming

C.J.Butz, S.Hua, R.B.Maguire
*Department of Computer Science, University of Regina*
*Regina, SK, Canada, S4S 0A2*
{butz, huash111, rbm }@cs.uregina.ca

**Abstract:** *In this paper, we present a Bayesian intelligent tutoring system for computer programming, called Bits. Our system takes full advantage of Bayesian networks (BNs), which are a formal framework for uncertainty management. We discuss how to employ BNs as an inference engine to guide the students' learning process. In addition, we describe the architecture of Bits and the role of each module in the system. Bits has been implemented and will be employed in an upcoming introductory programming course at the University of Regina.*

**Keywords:** *Intelligent tutoring system, Bayesian networks, educational tools for learning*

## 1. Introduction

The motivation for developing our *Bayesian intelligent tutoring system* (Bits) can be easily understood with the following example. A student in an introductory computer programming course states that she does not understand the For-Loop construct. During the explanation of the For-Loop construct, the class instructor realizes that the student does not understand relational operators, which are the prerequisite to the For-Loop construct. In other words, although the student claimed to not understand the For-Loop construct, the real problem actually involved a different, prerequisite concept. Thus, there is a clear need to automate this exchange, namely, to develop a tutoring system that can help guide a student through the introductory programming concepts.

Obviously, there is a certain amount of uncertainty built into this process. For instance, a student may truly believe she understands a concept when in fact she doesn't. To manage the uncertainty, we adopt *Bayesian networks* (BNs) [4,5,6], which utilize probability theory as a formal framework for uncertainty management. BNs have been successfully applied in practice to a wide variety of uncertain problems, including the jet propulsion systems on NASA's space shuttles [1], and Office Assistant in the Microsoft Windows operating system [2].

The rest of this paper is organized as follows. In Section 2, we described the modules comprising the Bits architecture. A sample session is outlined in Section 3. The conclusion is presented in Section 4.

## 2. General Architecture of Bits

In this section, we outline the major components of our system and describe how they interact with each other.

To simplify the task of developing an intelligent tutoring system, we restrict the scope of the problem as follows. First, the system is built to tutor students using the C++ programming language. Second, only elementary topics are covered, namely, those typically found in a first course on programming. That is, concepts such as variables, assignments, and control structures are included, but more sophisticated topics like pointers and inheritance are not. As illustrated in Figure 1, the four main modules of Bits are Bayesian networks, the knowledge base, the user interface, and the study module. These components are examined in the following subsections.

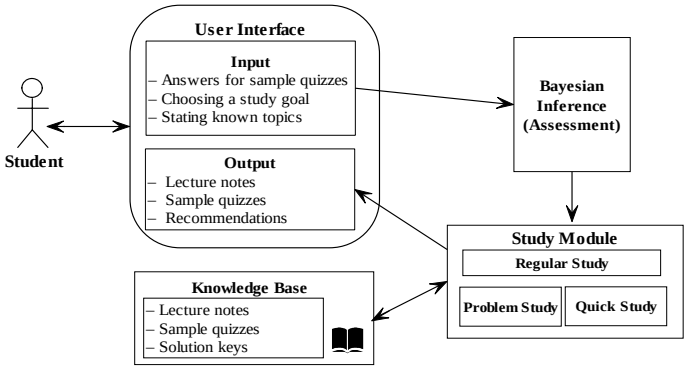


Figure 1 Architecture of Bits.

### 2.1 Bayesian networks

The key to aid the student to navigate through the concepts is two-fold. On one hand, the prerequisite information has to be modeled. On the other hand, we need to keep track of student knowledge regarding each concept. Bayesian networks can help us meet both of these objectives.

A Bayesian network (BN) [4,5,6] consists of directed acyclic graph (DAG) and a corresponding set of conditional probability distributions (CPDs). Based on the *probabilistic conditional independencies* [6] encoded in the DAG, the product of the CPDs is a *joint probability distribution* (jpd). In other words, Bayesian networks serve as both a semantic modeling tool and an economical representation of a jpd. There are many inference algorithms in BNs for computing probabilities of variables given other variables to take on certain values. For example, given that variable B has value *b* and variable D has value *d*, what is the probability that variable A is *a*? There are also numerous implementations of BN software [3].

For our purposes, we identified a set of concepts that are taught in CS110 (the first computer programming course at the University of Regina). Each concept is represented by a node in the graph. We add a directed edge from one concept (node) to another, if knowledge of the former is a prerequisite for understanding the latter. Thus, the DAG can be constructed manually with the aid of the course textbook. For example, consider one instance of the For-Loop construct in C++ such as

```
for (i=1; i<=10; i++);
```

To understand the For-Loop construct, one must first understand the concepts of Variable assignment, Relational operators, and Increment (decrement) operators. These relationships can be modeled as depicted in Figure 2. Naturally, Figure 2 depicts a small portion of the entire DAG implemented in Bits.

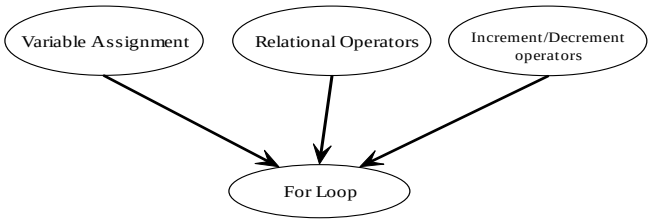


Figure 2 Sub-DAG for the For-Loop construct.

The next task in the construction of the BN is to specify a CPD for each node given its parents. For variable $a_i$ with parent set $P_i$, a CPD $p(a_i|P_i)$ has the property that for each configuration (instantiation) of the variables in $P_i$, the sum of the probabilities of $a_i$ is 1.0. In Figure 2, the parent set of the For-Loop node is {Variable assignment, Relational operators, Increment /decrement operators}. The corresponding CPD $p(For-Loop \mid Assignment, Relational\ Operators, Incre/Decrement\ operators)$ is shown in Figure 3.

| Parent Nodes | | | For Loop | |
|---|---|---|---|---|
| Variable Assignment | Relational Operators | Incre/Decrement operators | *known* | *not known* |
| *known* | *known* | *known* | 0.75 | 0.25 |
| | | *not known* | 0.39 | 0.61 |
| | *not known* | *known* | 0.50 | 0.50 |
| | | *not known* | 0.22 | 0.78 |
| *not known* | *known* | *known* | 0.50 | 0.50 |
| | | *not known* | 0.29 | 0.71 |
| | *not known* | *known* | 0.40 | 0.60 |
| | | *not known* | 0.15 | 0.85 |

Figure 3 The CPD corresponding to the For-Loop node in Figure 2.

All CPDs for the DAG were obtained from the results of previous CS110 final exams. We first identified the concept being tested for each question. If the student answered the question correctly, then we considered the concept *known*. Similarly, if the student answered the question incorrectly, then we considered the concept *unknown (not known)*. The probability of each concept being known, namely, $p(a_i = known)$, can then be determined. Moreover, we can also compute $p(a_i=known, P_i=known)$, i.e., the probability that the student correctly answers both the concept $a_i$ and the prerequisite concepts $P_i$. From $p(a_i=known, P_i=known)$, the desired CPD $p(a_i=known \mid P_i=known)$ can be obtained. Thereby, we can calculate every CPD for the entire Bayesian network.

### 2.2 Knowledge Base

The knowledge base contains the class lecture notes in the form of web pages, a repository of sample tests, which are in the form of interactive flash multimedia files, and solution keys. Both the lecture notes and quizzes are organized by concept. This allows the concepts to be indexed and retrieved efficiently.

The class lecture notes are displayed while the user is learning a new concept. On the contrary, a sample quiz is displayed when Bits is trying to determine whether or not a student has understood a particular concept.

### 2.3 User Interface Module

A student interacts with Bits through the user interface module. This interaction is partitioned into two sub-modules; an input module for input from a student to Bits, and an output module for output from Bits to a student.

The output module displays the class lecture notes through a web browser. It uses dialog boxes to display quizzes and offer pedagogical suggestions.

The primary goal of the input module is to update the BN based on evidence collected from the student. There are two ways in which the student can enter information into Bits:

(a) Direct input,

(b) Select multiple-choice answer in sample quiz.

For (a), after the student has finished reading the displayed lecture notes, Bits will ask the student to select one of the three options: (i) I understand this concept, (ii) I don't understand this concept, or (iii) I am not sure (quiz me).

If either of the first two choices is selected, then the BN can be immediately updated. On the other hand, the last choice leads to the second type of input.

In case (b) when the student is not sure about a concept, Bits will retrieve the appropriate quiz from the knowledge base and present it to the student. The question(s) are multiple choice. After the student inputs the answer, Bits compares the answer with the solution key. Bits then gives the student immediate feedback and updates the BN accordingly.

### 2.4 Study Module

The study module guides the student through the class concepts using three sub-modules called Regular Study, Problem Study, and Quick Study.

#### 2.4.1 Regular Study

We keep track of the student's knowledge of a concept using three categories; (i) known, (ii) ready to learn, and (iii) not ready to learn. A concept is considered known if the BN indicates a probability greater or equal to 0.70. A concept is marked ready to learn if the probability is less than 0.70 and all of the parent concepts are known. Finally, a concept is labeled not ready to learn, if at least one parent concept is not known.

When Bits is first started, the concepts are labeled based on the initial probabilities obtained from the BN. Traffic signs are employed as follows: yellow (known), green (ready to learn), and red (not ready to learn), as illustrated in Figure 4. A student can select a green topic in Figure 4. The lecture notes on the chosen topic are then displayed. For example, if the student chooses the green concept "Floating-point numbers" in Figure 4, Bits displays the lecture notes in Figure 5.

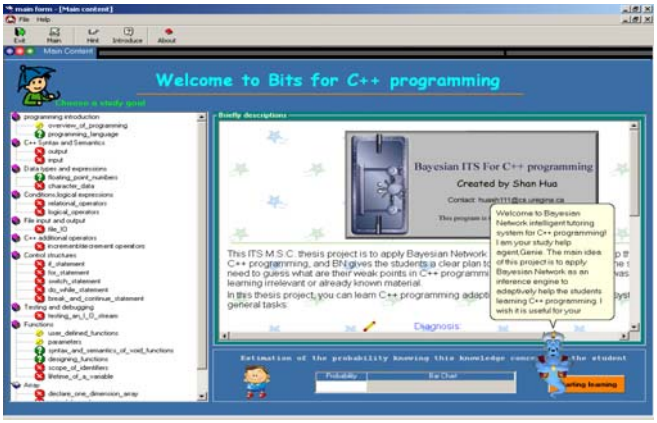


Figure 4 Navigation Menu: green means "ready to learn," yellow means "already known," and red means "not ready to learn."

After reading the lecture notes, the student selects one of the following three choices:

(a) I understand this concept;
(b) I don't understand this concept;
(c) I am not sure if I understand this concept,

as shown in the bottom right corner of Figure 5.

If the student selects (a), then the BN is updated and the Navigation Menu is again shown. If (b) is selected, then additional help is required. We will discuss this in the next subsection. If (c) is selected, the sample quiz on this topic is retrieved from the knowledge base and shown to the student. The student answers the question and Bits can give immediate feedback to the student, such as the correct solution to the question, and decides whether (a) or (b) is appropriate. Again, if (a), then the Navigation Menu is refreshed. We discuss (b) next.

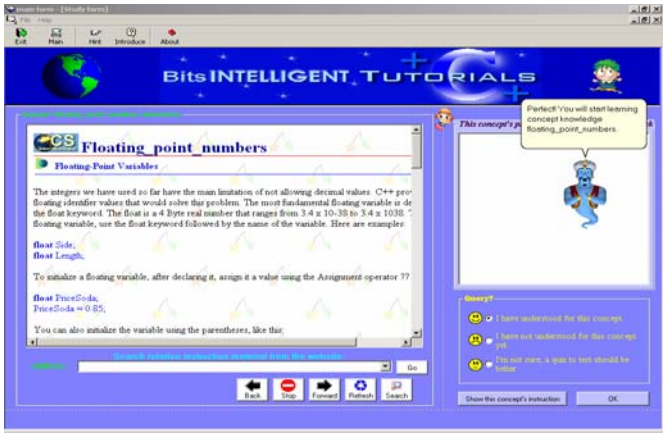


Figure 5 Lecture notes for concept "Floating-point-numbers."

#### 2.4.2 Problem Study

This module is useful when a student indicates that a concept is not understood. One possibility is to simply ask the student to read the lecture notes again. However, Bits is designed to show the student both the concept to be learned and the prerequisite concepts of this topic, namely, the parents of the concept in the BN.

The student is given the flexibility to revisit the prerequisite concepts to confirm that they are indeed understood. The rationale is that a student may believe that a prerequisite concept is understood when in fact it is not.

#### 2.4.3 Quick Study

A student may want to learn a particular topic without learning every single topic. For example, a student may want to learn "File I/O" for an impending exam or assignment deadline. The student would then like to learn the minimum set of concepts in order to understand the chosen concept.

Bits meets this need with the Quick Study sub-module. The student is allowed to select a "not ready to learn" concept in the Navigation Menu. In this situation, Bits will display a learning sequence for the chosen topic. In other words, all necessary ancestral concepts in the BN will be shown to the student in a proper sequence for learning.

For example, suppose the student selected the not ready to learn concept "File I/O" in the Navigation Menu of Figure 4. Then Bits displays the ancestral concepts in order, grouping by "known" and "unknown", namely, "overview of programming" marked by "known", and "programming language," "output," "input" marked by "unknown", as depicted in Figure 6. The student needs to learn "programming language," "output" and "input" first.

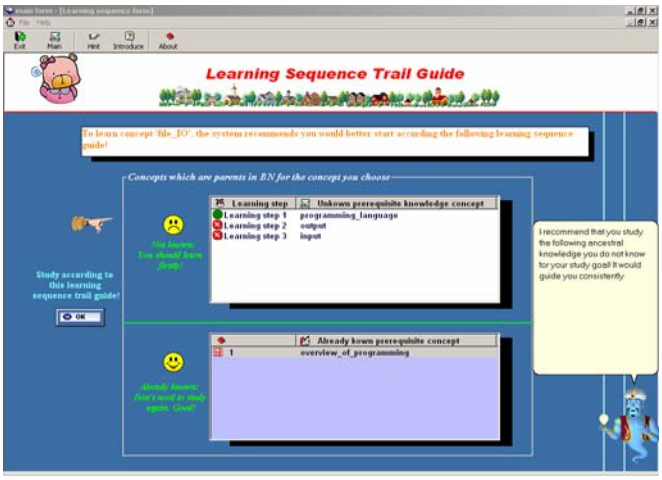


Figure 6 The Quick Study menu generates a learning sequence for the concept "File I/O" in Figure 4, which is "not ready to learn."

## 3. A Sample Session

When Bits starts, the Navigation Menu is displayed. The student can either study any topic that is marked "ready to learn," or ask for the minimum set of topics to be learned in order to quickly learn a topic marked "not ready to learn."

Suppose the Navigation Menu in Figure 4 is displayed to the user. The left side indicates, for instance, that the concept "overview of programming" is already known, the concept "Floating point numbers" is ready to learn, and the concept "File I/O" is not ready to learn.

### 3.1 Regular Study

In this situation, the student selects any topic that is marked ready to learn. For example, if the student selects "Floating-point numbers" in Figure 4, then the lecture notes for this concept are displayed, as shown in Figure 5. After reading the notes, the student selects one of the choices at the bottom right of Figure 5. If the student indicates that she understands floating point numbers, then the BN is updated and the Navigation Menu is again displayed (but this time the concept floating point numbers will be labeled as known).

### 3.2 Problem Study

If the student indicates that the concept is not understood after reading the lecture notes, then Bits displays the prerequisite concepts. Again using Figure 5 as an example, the prerequisite concepts are displayed, similar to that illustrated in Figure 6. Here Bits confirms that each of the prerequisite concepts is understood. Only afterwards does Bits go back to the problem concept (Figure 5).

### 3.3 Quick Study

In some situations, a student may want to quickly learn a particular concept. For instance, a student may want to learn "File I/O" without learning all of the concepts discussed in the class textbook leading up to the discussion on this topic. In other words, the students wishes to learn the minimal set of concepts needed to understand "File I/O" concept.

Bits facilitates this procedure by allowing the student to select a "not ready to learn" concept in the Navigation Menu. For example, using Figure 4 as the instance of the Navigation Menu, the user can select "File I/O" from the left side. Bits then displays the ancestor concepts of "File I/O" in the BN, as depicted in Figure 6. Then student then selects the first unknown concept shown in the top box of Figure 6. After this concept is understood, the student selects the second concept, and so on.

## 4. Conclusions And Future Work

This paper discusses a new architecture of designing an ITS (Bits) for computer programming using Bayesian technology. Centering on the explicit structure and the contents of each component of the architecture, we described the concept and realized the prototype of Bits. Bits provides remote access to hypermedia-structured learning material which includes instruction notes, tests, and examples. Unlike traditional web based education tools, Bits provides the learner with intelligent navigation support, recommendation, and integrates the features of an electronic hypermedia textbook with intelligent tutoring tactics. Bits can propose learning goals and guide users by generating reading sequences for them.

Future work will involve incorporating the more sophisticated concepts of C++ into Bits. We also hope to extend Bits by incorporating other programming languages such as Java.

## References

[1]. E. Horvitz, E., M. Barry, "Display of Information for Time Critical Decision Making," *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, pp. 296-305,1995.

[2]. E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users," *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, pp. 256-265, 1998.

[3]. http://www.ai.mit.edu/~murphyk/Bayes/bnsoft.html

[4]. J. Pearl. *Probabilistic Reasoning in Intelligent Systems:*

*Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.

[5]. S.K.M. Wong and C.J. Butz, "Constructing the Dependency Structure of a Multi-Agent Probabilistic Network," *IEEE Transactions on Knowledge and Data Engineering*, 13(3): pp. 395-415, May 2001.

[6]. S.K.M. Wong, C.J. Butz, and D. Wu, "On the Implication Problem for Probabilistic Conditional Independency," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 30(6): pp.785-805, November 2000.