A Web-based Bayesian Intelligent Tutoring System for Computer Programming

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

Abstract

In this paper, we present a Web-based intelligent tutoring system, called BITS. The decision making process conducted in our intelligent system is guided by a Bayesian network approach to support students in learning computer programming. Our system takes full advantage of Bayesian networks, which are a formal framework for uncertainty management in Artificial Intelligence based on probability theory. We discuss how to employ Bayesian networks as an inference engine to guide the students' learning processes. In addition, we describe the architecture of BITS and the role of each module in the system. Whereas many tutoring systems are static HTML Web pages of a class textbook or lecture notes, our intelligent system can help a student navigate through the online course materials, recommend learning goals, and generate appropriate reading sequences.

1 Introduction

Web-based learning systems are increasingly popular due to their appeal over traditional paper-based textbooks. Web courseware is easily accessible and offers greater flexibility through the internet [6], that is, students can control their own pace of study and do not depend on a teacher's presence and rigid classroom schedules. Unlike printed textbooks, Web-based tutoring systems can incorporate rich multi-media and interactive elements, such as audio, video and animation to make a point. Web-based learning systems can add hyperlinks to allow students to click on a link on one Web page and immediately be transferred to another page or to other relevant sites. However, since many current Web-based tutoring systems are static HTML Web pages, they suffer from two major shortcomings, namely, they are neither interactive nor adaptive [6]. Most Web courses present the same static learning materials to students with widely differing knowledge levels of a given subject. Therefore, this kind of system is unable to satisfy the heterogeneous needs of many users [7].

Web Intelligence is a direction for scientific research that explores practical applications of Artificial Intelligence to the next generation of Web-empowered systems [54]. Yao and Yao [58] argue that a system should be robust enough to deal with various types of users. Moreover, Brusilovsky and Maybury [7] explicitly state that the solution needed to fix the problem of the traditional "one-size-fits-all" approach is to develop systems with an ability to adapt their behavior to the goals, tasks, interests, and other features of individual users and groups of users. In the context of Web-based tutoring systems, Liu et al. [36] developed an intelligent system for assisting a user in solving a problem. Obviously, this involves creating systems that can make decisions based on uncertain or incomplete information. One formal framework for uncertainty management is *Bayesian networks* [41, 55, 56], which utilize probability theory as a formal framework for uncertainty management in Artificial Intelligence. Web intelligence researchers have applied Bayesian networks to many tasks, including student monitoring [28, 36], e-commerce [27, 42], and multi-agents [34, 53].

The purpose of this paper is to put forth a Web-based intelligent tutoring system, called BITS, to support students in learning computer programming. The decision making process conducted in our intelligent system is guided by a Bayesian network. Similar to [28, 36], BITS can assist a student in navigation through the online materials. Unlike [28, 36], however, BITS can recommend learning goals, and generate appropriate reading sequences. For example, a student may want to learn "File I/O" without having to learn every concept discussed in the previous materials. BITS can determine the minimum prerequisite knowledge needed in order to understand "File I/O" and display the links for these concepts in the correct learning sequence. In this way, one can address the problem of Web-based learners' unproductive navigation, and refocus them on their study objectives by making the tutoring systems adaptable to different types of learners. BITS has been implemented and was used in the summer 2004 session of CS110, the initial computer programming course in Computer Science at the University of Regina. Although the feedback collected from the students was limited, the result was very positive. We believe that BITS will be useful for supporting instructors teaching computer programming in their institutions. In particular, this research work is important for institutions where it is difficult to provide personalized instruction that they need [31]. Moreover, we believe that BITS is very useful to any work applying Bayesian networks as a model for developing adaptive Web-based education tools for different courses. Unlike our previous work [11], in this paper, we give an in-depth discussion of Intelligent Tutoring Systems (see Section 2) and a Bayesian network approach applied in our system (see Section 3). In addition, each component in the architecture of BITS is examined in greater detail (see Section 4). In Section 5.2, the implementation of BITS is presented. A more comprehensive survey of related work is discussed in Section 6. Furthermore, our future work to enhance BITS is discussed (see Section 7.2). Empirical studies have shown that individual one-on-one tutoring is the most effective mode of teaching and learning [5]. With large numbers of students in Web-based learning environment, however, this kind of individualized tutoring is difficult to deliver. BITS serves as intelligent software for implementing computer-assisted one-on-one tutoring.

This paper is organized into seven sections. Section 2 reviews Intelligent Tutoring Systems. In Section 3, we discuss Bayesian networks and probabilistic inference. In Section 4, we describe the architecture of BITS, the role of each module in the system. In particular, we discuss BITS's capability for adaptive guidance by applying the Bayesian network approach. In Section 5, we describe the features that allow BITS to be accessed via the Web and the implementation of BITS. Related works are discussed in Section 6. The conclusion is presented in Section 7.

2 Background Knowledge for Intelligent Tutoring Systems

In this section, we review several problems in conventional tutoring systems and motivations for designing Intelligent Tutoring Systems. We then discuss the general framework of an Intelligent Tutoring System and the function for each component. In addition, we focus on the student model, which is the key aspect of Intelligent Tutoring Systems needed to realize one-on-one tutoring. Finally, we review two classical approaches applied in current Web-based Intelligent Tutoring Systems.

2.1 Review of Intelligent Tutoring Systems

Since the 1960s, researchers have created numerous Computer Assisted Instructional systems [48, 51]. The purpose for applying computers in assisting instruction is to help students learn more efficiently. Traditional education systems instructing via computers are called *Computer-Assisted Instruction* (CAI) systems. CAI systems present instructional materials in a rigid tree structure to guide the students from one content page to another depending on their answers [37], as illustrated in Figure 1 [22]. While traditional CAI systems may be somewhat effective in helping learners, they are restrictive in that they do not consider the diversity of students' knowledge states and their particular needs (c.f. [7] and [58]). Such systems do not generate flexible instructional plans. Instead, they follow a pre-specified and fixed plan. Moreover, CAI systems are not adaptive and unable to dynamically provide the same kind of individualized attention that students would receive from human teachers [4].

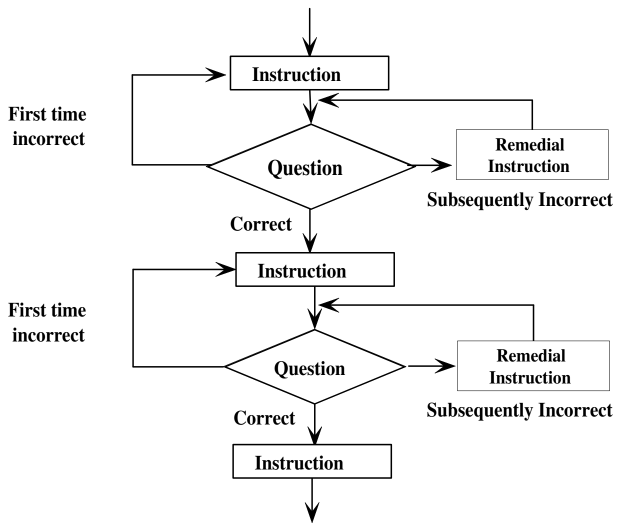


Figure 1. Tree structure for one traditional CAI system to guide the student in navigating through the instructional materials.

This drawback has prompted a promising direction in the application of Artificial Intelligence techniques in education known as *Intelligent Tutoring Systems* (ITSs) [10]. Intelligent Tutoring Systems are computer-based programs that present educational materials in a flexible and personalized way that is similar to one-to-one tutoring [6]. In particular, ITSs have the ability to provide learners with tailored instructions and feedback. The basic underlying idea of ITSs is to realize that each student is unique. These systems can be used in the traditional educational setting or in distant learning courses, either operating on stand-alone computers or as applications that deliver knowledge through the internet.

ITSs have been shown to be highly effective in increasing students' performance and motivation levels compared with traditional instructional methods (e.g. [32], [46]). One of the key elements that distinguishes ITSs from more traditional CAI systems is ITSs' capability to dynamically maintain a model of a student's reasoning and learning that keeps track of a student's knowledge during the study [47]. As noted by Shute and Psotka [47], ITSs must be able to achieve three main tasks:

(i) accurately diagnose a student's knowledge level using principles rather than preprogrammed responses;

(ii) decide what to do next and adapt instruction accordingly;

(iii) provide feedback.

This kind of diagnosis and adaptation, which is usually accomplished using Artificial Intelligence techniques, is what distinguishes ITSs from CAIs. Bloom [5] demonstrates that individual one-on-one tutoring is the most effective mode of teaching and learning. Carefully designed and individualized tutoring produces the best learning for the majority of people. ITSs uniquely offer a technology that implements computer-assisted one-on-one tutoring.

2.2 The Key Components of ITS

Early CAI systems were not modular [57]. This unfavourable structure caused problems when a system required modification, and it was sometimes necessary to restructure the whole system. There was, then, a need to divide the system into separate components: the knowledge to be taught, the instructional method, the user interface and the student modelling.

Researchers typically separate an ITS into several different parts, and each part plays an individual function. Usually, most ITSs has four common major components [48], as illustrated in Figure 2:

(i) Knowledge domain;

(ii) Student model;

(iii) Teaching strategies;

(iv) User interface.

1. **Knowledge Domain:**

   The knowledge domain stores learning materials that the students are required to study for the topic or curriculum being taught.

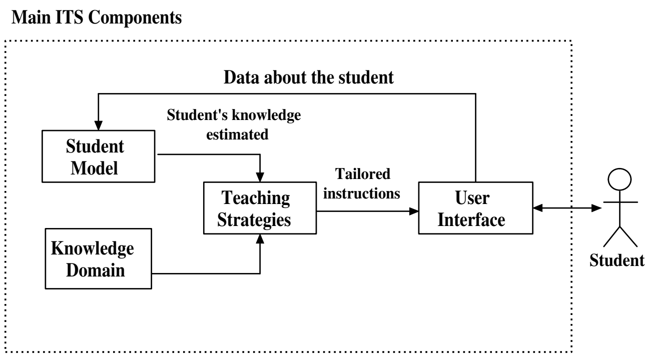


Figure 2. The major components of most Intelligent Tutoring Systems.

2. **Student Model:**

   The student model stores information that is specific to each individual learner and enables the system to identify different users. Usually, this information reflects the system's understanding of one learner's current knowledge state. Thus, the student model can track a student's understanding and particular need. Without an explicit student model, the teaching strategies component is unable to make decisions to adapt instructional content and guidance (see Figure 2) and is forced to treat all students similarly.

   Student modelling is sometimes thought of as a subproblem of the user modelling problem [25], whereby the target application is an ITS. Student modelling presents well-known difficulties stemming from the fact that modelling the student within an intelligent tutoring system involves a good deal of inherent uncertainty [14]. It is hard to establish unequivocally what a student knows and what she is learning [14, 25]. Thus, one of the biggest challenges in designing ITSs is the effective assessment and representation of the student's knowledge state and specific needs in the problem domain based on uncertainty information.

   The task of dealing with the uncertainty management for the student model is thus challenging [17]. Until the late 1980s, researchers interested in student modelling had only limited techniques for uncertainty management available, and they mostly had to rely either on poorly understood ad hoc techniques or on general techniques [25]. Fortunately, over the past decade the question of how to manage uncertainty has been a rapidly expanding and increasingly mainstream research topic in Artificial Intelligence. Various approaches in Artificial Intelligence have been proposed for uncertainty reasoning [40], including rule-based systems [9], fuzzy logic [30], DempsterShafer theory of evidence, and neural networks. Bayesian networks [41] are a powerful approach for uncertainty management in Artificial Intelligence [55, 56]. In Section 3, we will discuss details of the Bayesian network approach.

3. **Teaching Strategies:**

   The teaching-strategies component refers to instructional techniques for teaching. For example, the component decides when to present a new topic, how to provide recommendations and guidance, and which topic to present. As mentioned earlier, the assessment result of the student model is input to this component, so the system's pedagogical decisions reflect differing needs of students. Thus, this component needs to take appropriate actions to manage one-on-one tutoring, such as switching teaching strategies and using a variety of teaching approaches at the appropriate times according to the student's particular needs and problems.

4. **User Interface Component**

   The user interface component decides how the system interacts with a user. The dialogue and the screen layouts are controlled by this component. A well-designed interface can enhance the capabilities of an ITS by allowing the system to present instructions and feedback to the student in a clear and direct way.

2.3 Review of Intelligent Tutoring Systems Technology Research

In the review of existing ITSs by Murray [39], two major types of Intelligent Tutoring Systems are identified according to the system's objective:

1. **Problem Solving Support:**

   For many years, problem solving support was considered as the primary duty of an ITS [6]. The purpose for problem solving support is usually to provide the student with intelligent help for each step when resolving a task, such as a project or a problem. When the student is stuck on one step, the system provides a hint showing the next correct solution step for the student, or offering appropriate error feedback. In this setting, the critical problem for the system is to interpret the student's actions and infer the solution plan that the student is currently following based on a partial sequence of observable actions. That is, the system needs to understand the student's plan, and apply this understanding to provide help. Examples of this type of ITS are [1, 17, 21, 28, 36, 43, 50].

2. **Curriculum Sequencing:**

   Curriculum sequencing is now the most popular and important technology in Web-based ITSs [6]. The objective of curriculum sequencing technology (also referred to as instructional planning technology) [6] is to provide the student with a personalized optimal path through the learning material. The examples are [3, 8].

   Recommending appropriate learning sequencing is necessary in Web-based education. Web-based learning students usually work alone without a teacher's instructional assistance and they study the subject at their own pace. As a result, appropriate learning sequencing recommendations are essential in order to enable each student to learn the subject in the most beneficial and individualized way [6]. In this paper, we describe the implementation of this type of instructional planning technology in BITS for teaching computer programming.

3 Bayesian Networks

In this section, we discuss modelling with Bayesian networks and probabilistic inference.

Let $U = \{v_1, v_2, \ldots, v_n\}$ denote a finite set of discrete random variables. Each variable v_i is associated with a finite domain, denoted $dom(v_i)$, representing the values v_i can take on. Let $dom(U)$ be the Cartesian product the domains of the individual variables in U; namely,

$$dom(U) = dom(v_1) \times dom(v_2) \times \ldots \times dom(v_n).$$

Each element $u \in dom(U)$ is called a *configuration* of U

Definition 1 *[44]* A *joint probability distribution* (JPD) is a function p on $dom(U)$, such that the following two conditions hold: (i) $0 \leq p(u) \leq 1.0$, for each configuration $u \in dom(U)$, and (ii) $\sum_{u \in dom(U)} p(u) = 1.0$.

For brevity, we refer to p as a probability distribution on U rather than $dom(U)$, and we call U not $dom(U)$ as its domain.

Next, we discuss an important notion of the conditional probability distribution.

Definition 2 Let v_i be a variable and X be a finite, possibly empty set of variables. A *conditional probability distribution* (CPD) for v_i given X, denoted $p(v_i|X)$, is the probability distribution that has a property whereby, for each configuration $x \in dom(X)$,

$$\sum_{c \in dom(v_i)} p(v_i = c|X = x) = 1.0,$$

where X is the set of conditional variables for variable v_i, $v_i \notin X$, and $x \in dom(X)$.

Example 1 *Let $U = \{a, b, c, d, e, f\}$ be a set of binary variables. The corresponding CPDs $p(a)$, $p(b|a)$, $p(c|a)$, $p(d|b)$, $p(e|c)$, $p(f|d, e)$ are given in Figure 3, where the missing conditional probabilities can be calculated according to the definition of CPD. For instance, $p(a = 0) = 0.45$, $p(b = 1|a = 0) = 0.3$.*

a	$p(a)$
1	0.55

a	b	$p(b\| a)$
1	1	0.7
0	1	0.4

a	c	$p(c \| a)$
1	1	0.1
0	1	0.2

b	d	$p(d\|b)$
1	1	0.5
0	1	0.7

c	e	$p(e \| c)$
1	1	0.4
0	1	0.7

d	e	f	$p(f \| d, e)$
1	1	1	0.8
1	0	1	0.4
0	1	1	0.3
0	0	1	0.9

Figure 3. One set of CPDs for the variables in $U = \{a, b, c, d, e, f\}$, namely.

Clearly, it may be impractical to obtain the joint distribution on U directly; for example, one would have to specify $2^n - 1$ entries for a distribution over n binary variables. Bayesian networks [41] utilize the notion of conditional independence to facilitate the acquisition of JPD.

Definition 3 *[56]* Let $p(U)$ be a JPD, and X, Y, Z be pairwise disjoint subsets of U. Let x, y and z denote arbitrary values of X, Y and Z respectively. We say X and Z are *conditionally independent* given Y under the joint probability distribution p, denoted by $I(X, Y, Z)$, if

$$p(x|y, z) = p(x|y),$$

whenever $p(y, z) > 0$. This conditional independence $I(X, Y, Z)$ can be equivalently written as

$$p(x, y, z) = \frac{p(x, y) \cdot p(y, z)}{p(y)}.$$

Example 2 *Suppose $U = \{a, b, c, d, e, f\}$, the following conditional independencies, namely, $I(c, a, b)$, $I(d, b, ac)$, $I(e, c, abd)$, $I(f, de, abc)$ hold on U, namely,*

$$\begin{aligned} p(c|a, b) &= p(c|a), \\ p(d|a, b, c) &= p(d|b), \\ p(e|a, b, c, d) &= p(e|c), \\ p(f|a, b, c, d, e) &= p(f|d, e). \end{aligned}$$

By the chain rule of probability,

$$\begin{aligned} p(a, b, c, d, e, f) \;\; = \;\; & p(a) \cdot p(b|a) \cdot (c|a, b) \cdot (d|a, b, c) \\ & \cdot p(e|a, b, c, d) \cdot (f|a, b, c, d, e). \end{aligned}$$

By the above conditional independencies, the JPD $p(a, b, c, d, e, f)$ can be factorized as:

$$p(a, b, c, d, e, f) = p(a) \cdot (b|a) \cdot (c|a) \cdot (d|b) \cdot (e|c) \cdot (f|d, e).$$

Bayesian networks are a graphical method for representing the independence information.

Definition 4 A *Bayesian network [41]* is a pair $\mathcal{B} = (D, C)$. In this pair, D is a *directed acyclic graph* (DAG) on a set U of variables. The directed edges in the graph reflect the dependencies that hold among the variables. $C = \{p(v_i|P_i) \mid v_i \in D\}$ is the corresponding set of conditional probability distributions, where P_i denotes the *parent set* of each variable v_i in the DAG D.

We will use the terms Bayesian network and DAG interchangeably if no confusion arises. The *d-separation* algorithm [41] can be used to infer *probabilistic conditional independencies* from a DAG. That is, if Y d-separates X and Z in the DAG D, then the conditional independence $I(X, Y, Z)$ holds in the JPD. The strength of the dependency is quantified by a conditional probability.

Example 3 *Let $U = \{a, b, c, d, e, f\}$ be a set of binary variables. One Bayesian network on U is the DAG in Figure 4 together the CPDs in Figure 3. It can be verified using d-separation, that the four independencies in Example 2 hold in the DAG.*

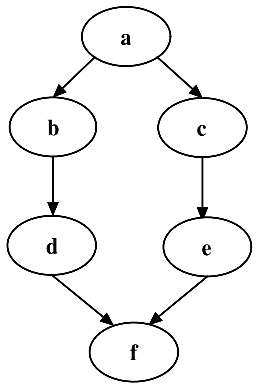


Figure 4. A directed acyclic graph (DAG) on variables $U = \{a, b, c, d, e, f\}$.

Based on the probabilistic conditional independencies encoded in the DAG, the product of the CPDs in C is a unique joint probability distribution on U:

$$p(v_1, v_2, \ldots, v_n) = \prod_{i=1}^{n} p(v_i|P_i),$$

as shown in Example 2 for the DAG in Figure 4.

Example 4 *Recall Example 2. While specifying $p(U)$ directly involves stating 63 prior probabilities ($2^6 - 1$ for six binary variables), by the DAG in Figure 4, only 13 conditional probabilities need be given.*

Bayesian networks, thus, provide a semantic and concise modelling tool for modelling uncertainty in complex domains, which greatly facilitates the acquisition of probabilistic knowledge.

Another important notion we need to discuss is ancestral number. A numbering of the variables in a DAG is called *ancestral* [12], if the number corresponding to any vertex is lower than the numbers corresponding to all of its children. For example, recall the DAG in Figure 4. One ancestral numbering of these variables is $a = 1, b = 2, c = 3, d = 4, e = 5, f = 6$. We now turn our attention to probabilistic inference in the next discussion.

The main task of Bayesian networks is for probabilistic reasoning. Probabilistic reasoning, namely, processing queries, simply means computing $p(V)$, the marginal distribution for a set V of variables, or $p(V|E = e)$, the posterior probability distributions of a set V of unobserved variables given $E = e$, where $V \cup E = \emptyset$ and $V, E \subseteq U$. The *evidence* in the latter query is that the set E of variables is observed to be configuration e, $e \in dom(E)$. When we enter the evidence and use it to update the probabilities, we call it *propagation of evidence*, or simply *propagation*. There are many probabilistic inference algorithms [16, 26, 33, 35, 45, 60] for processing queries which seem to work well in practice. There are also numerous implementations of Bayesian network software [20].

4 General Architecture of BITS

In this section, we introduce BITS for computer programming. We outline the major components of our system and describe how they interact with each other. We discuss how to use a Bayesian network within BITS for modelling and inference, and how BITS can offer tailored pedagogical options to support an individual student.

As BITS is designed, we adapt the common framework of the ITS discussed in Section 2 and divide the user interface module into two interface submodules, the input submodule and output submodule. We also separate the adaptive guidance module (this module corresponds with the teaching-strategies component in the general framework of an ITS in Section 2) into three submodules: Navigation Support, Prerequisite Recommendations, and Generating Learning Sequences. The student model is implemented by the Bayesian network approach.

Figure 5 shows the overall architecture of BITS. Four modules contained in BITS are Bayesian networks, the knowledge base, the user interface, and the adaptive guidance module. Details of each component will be explained and examined in the following sections.

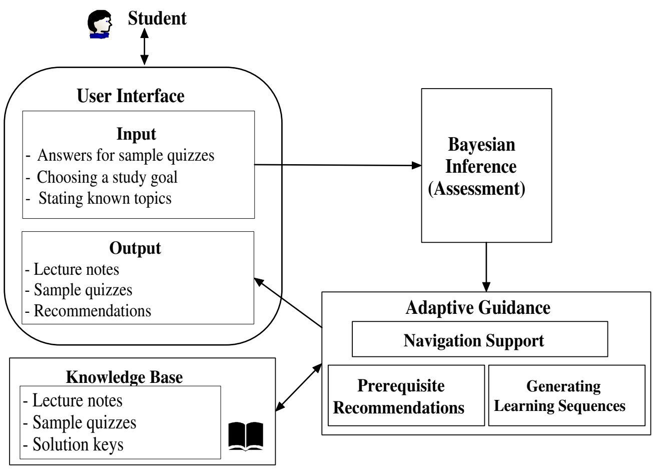


Figure 5. Architecture of BITS.

4.1 Bayesian Networks in BITS

In this sub-section, we discuss how BITS employs Bayesian networks as an inference engine to guide the students' learning process.

4.1.1 Modelling the Problem Domain

There are two tasks involved in helping a student navigate in a personalized Web-based learning environment: firstly, the structure of the problem domain must be modelled; secondly, student knowledge regarding each concept in the problem domain should be tracked. Bayesian networks can help us meet both these objectives.

To simplify the task of developing an intelligent tutoring system, we restrict the scope of the problem as follows: firstly, the system is built to tutor students in the C++ programming language; secondly, only elementary topics are covered, such as those typically found in introductory courses on programming. These concepts include concepts such as variables, assignments, and control structures, while more sophisticated topics like pointers and inheritance are excluded. For our purposes, we have identified a set of concepts that are taught in CS110, the introductory computer programming course at the University of Regina. Each concept is represented by a node in the graph. There exist learning dependencies among knowledge concepts, namely, prerequisite relationships. Using a Bayesian Network, the prerequisite relationships among the concepts can be represented directly and clearly. 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 textbook for our CS110 course [15], and it encodes the proper sequence for learning all the concepts in the problem domain.

Example 5 *Consider the following instance of the "For Loop" construct in C++:*

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

To understand the "For Loop" construct, one must first understand the concepts of "Variable Assignment" `(i=1)`*, "Relational Operators"* `(i<=10)`*, and "Increment/Decrement Operators"* `(i++)`*. These prerequisite relationships can be modelled as the DAG depicted in Figure 6.*

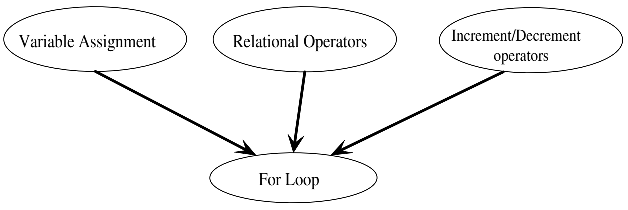


Figure 6. Modelling the prerequisite concepts of the "For Loop" construct.

Figure 6 depicts a small portion of the entire DAG implemented in BITS. The entire DAG implemented in BITS consists of 29 nodes and 43 edges, which is shown in Figure 17 in the Appendix of this paper. While the student runs BITS, she also can see the whole DAG constructed in BITS.

The next task in the construction of the Bayesian network is to specify a CPD for each node given its parents.

Example 6 *Recall the node v_i= For Loop with the parent set P_i=$\{$Variable Assignment, Relational Operators, Increment/Decrement Operators$\}$ depicted in Figure 6. A CPD p(For Loop$|$Variable Assignment, Relational Operators, Increment/Decrement Operators) is shown in Figure 7, where the missing conditional probabilities can be computed by the definition of the CPD.*

All CPDs for the DAG are obtained from the results of previous CS110 final exams. Firstly, we identify the concepts being tested by each question. Normally, these exam questions consist of multiple choice or filling-in-the-blank. If the student answers the question correctly, then we considered the concept *known*. Similarly, if the student answers the question incorrectly, then the concept is *unknown* (*not known*). The probability of each concept being known, namely $p(v_i = known)$, can then be determined. Moreover, we can compute $p(v_i = known, P_i = known)$

Variable Assignment	Relational Operators	Incre/Decrement Operators	For Loop	*p* (F|V,R,I)
known	*known*	*known*	*known*	0.75
known	*known*	*not known*	*known*	0.39
known	*not known*	*known*	*known*	0.50
not known	*known*	*known*	*known*	0.50
known	*not known*	*not known*	*known*	0.22
not known	*known*	*not known*	*known*	0.29
not known	*not known*	*known*	*known*	0.40
not known	*not known*	*not known*	*known*	0.15

Figure 7. The CPD corresponding to the "For Loop" node in Figure 6.

(i.e., the probability that the student correctly answers both the concept v_i and the prerequisite concepts P_i). From $p(v_i = known, P_i = known)$, the desired CPD $p(v_i = known|P_i = known)$ can be obtained through the following equation:

$$\begin{aligned} & p(v_i = known|P_i = known) \\ = \quad & \frac{p(v_i = known, P_i = known)}{p(P_i = known)}. \end{aligned}$$

While it is acknowledged that the accuracy of calculating the CPDs in this fashion is arguable [14], this approach to CPD acquisition is the only available way that we currently have. Thereby, by this approach, we calculate every CPD for the entire Bayesian network.

4.1.2 Personalized Learning

It has been argued [58] that systems should provide personalized environments. In this sub-section, we show how BITS adapts to the individual user. We begin by motivating the discussion.

Brusilovsky [6] states that several systems detect the fact that the student reads some information to update the assessment of her knowledge level. Some of them also include reading time or the sequence of read pages to enhance this assessment. However, the disadvantage of the above approaches is that it is difficult to measure whether a student really understands the knowledge by visiting Web pages of lecture notes.

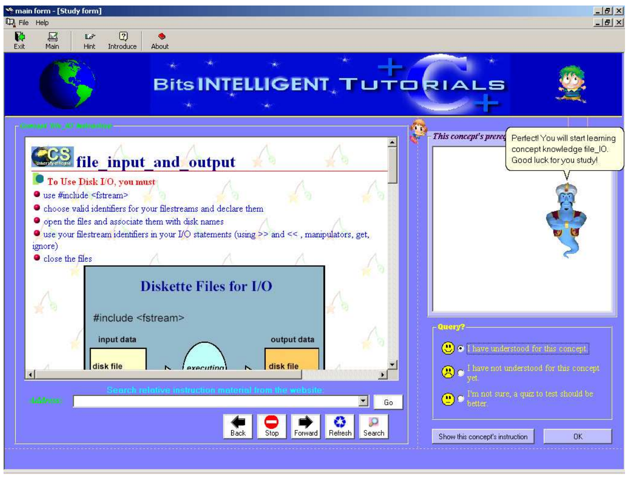
The primary purpose of observing a student's interaction with BITS is to obtain the evidence collected from the student to update the Bayesian network. In BITS, there are two methods of obtaining the evidence required to update the Bayesian network.

(i) A student's direct reply to a BITS query if this student knows a particular concept.

(ii) A sample quiz result for the corresponding concept to determine whether or not a student has understood a particular concept. Although a quiz question may pertain to several C++ topics, we subjectively associated each question with the most relevant topic.

We believe this approach is a more reliable method of estimation.

After the student finishes reading the displayed lecture notes, she provides BITS with feedback. More specifically, she selects from one of the following three choices: I understand this concept; I don't understand this concept; I'm not sure (quiz me). These choices are illustrated in the bottom right corner of Figure 8. The first two answers fall under the above-mentioned method (i) to obtain the evidence, while the last answer falls under method (ii).



file_input_and_output

To Use Disk I/O, you must
- use #include <fstream>
- choose valid identifiers for your filestreams and declare them
- open the files and associate them with disk names
- use your filestream identifiers in your I/O statements (using >> and << , manipulators, get, ignore)
- close the files

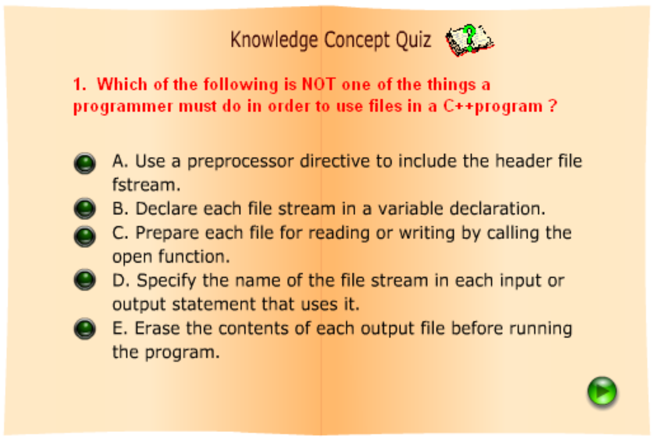
#include <fstream>

Perfect! You will start learning concept knowledge file_IO. Good luck for you study!

[✓ I have understood for this concept]
I have not understood for this concept yet
I'm not sure, a quiz to test should be better

Figure 8. A screen shot of BITS displaying the lecture notes and querying whether this concept is understood for the concept "File I/O."

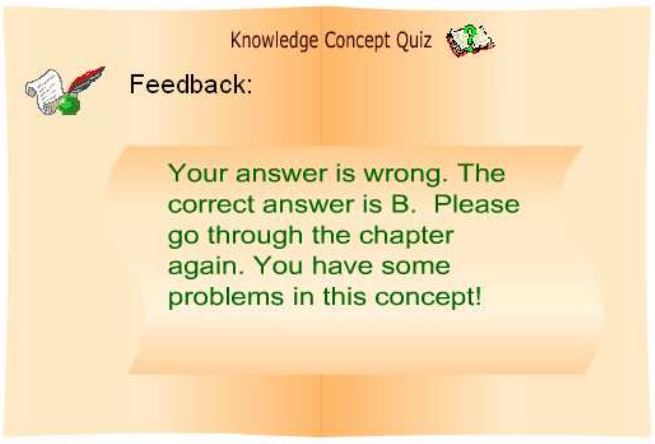
For method (i), the Bayesian network can immediately be updated to reflect the student's knowledge, or her lack thereof. In the case of method (ii), BITS retrieves the appropriate quiz from the database and presents it to the user. For instance, if the student indicates that she is not sure whether she understands the concept "File I/O," then BITS displays the quiz on "File I/O" in Figure 9. BITS then compares the student's answer with the appropriate solution key stored in the database. After providing an answer, the student is informed by immediate feedback whether the answer is correct or not. If the student answers all questions correctly in the sample quiz, the Bayesian network is updated and the Navigation Menu is again displayed (but this time the current concept will be marked as already known). If some of the answers are incorrect, the correct answer is displayed, and BITS also recommends that the student review the learning material on the current concept again. The Bayesian network is also updated accordingly.


Knowledge Concept Quiz

1. Which of the following is NOT one of the things a programmer must do in order to use files in a C++program ?

- A. Use a preprocessor directive to include the header file fstream.
- B. Declare each file stream in a variable declaration.
- C. Prepare each file for reading or writing by calling the open function.
- D. Specify the name of the file stream in each input or output statement that uses it.
- E. Erase the contents of each output file before running the program.

Figure 9. One question on a sample quiz for the concept "File I/O" in Figure 8.


Knowledge Concept Quiz

Feedback:

Your answer is wrong. The correct answer is B. Please go through the chapter again. You have some problems in this concept!

Figure 10. The feedback of BITS when the student chooses the incorrect answer for the question in Figure 9.

Example 7 *Consider the question shown in Figure 9, the correct answer is "B". If a student chooses an incorrect answer, say "A", the student receives the feedback shown in Figure 10.*

As will be discussed in Section 5.4, the Bayesian network is implemented using MSBNx [24]. This toolkit includes a probability inference algorithm for processing queries. We refer the reader to [16, 26, 33, 35, 45, 60] for detailed discussion on the inner workings of probabilistic inference algorithms. In Section 4.4, we turn our attention to using the updated Bayesian network for adaptive guidance.

4.2 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. The class lecture notes are displayed while the user learns 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.

All course materials, including both lecture notes and quizzes, are organized by knowledge concepts for C++ programming that correspond to the nodes in the Bayesian network. Using this approach, we separate knowledge concepts from actual instructional contents, and this separation is advantageous in many ways.

Firstly, this separation allows multiple teachers to write parts of the instructional materials and work independently. Instructors can compose their course contents without having to be concerned about other contents; moreover, it allows them to include information resources located anywhere on the World Wide Web.

Secondly, this separation enables us to apply Bayesian networks as the inference mechanisms that estimate the student's knowledge state. Because each concept corresponds to one node in the Bayesian network, it facilitates the sample quiz's result as the evidence for Bayesian networks to update the system's belief.

Thirdly, this separation facilitates changes to the content of the tutoring system. If we add additional information pages or change content, we need only re-index these pages accordingly; we need not change any components in the Bayesian network. This development allows for easy modification of the instructional contents without having to reformulate the whole system.

The last advantage is that this separation allows the concepts to be indexed and retrieved efficiently, allowing the system to adapt to one particular student's needs and knowledge state. When a student determines a study goal, the indexed lecture notes are displayed appropriately. We can also either propose programming examples for this concept on the Websites or generate reading sequences according to the student's actual knowledge state.

4.3 User Interface Module

A student interacts with BITS through the user interface module. This interaction is partitioned into two submodules; 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 dialogue boxes to display quizzes and offer pedagogical suggestions. These pedagogical suggestions include either the adaptive guidance provided by BITS, which we will discuss in the next subsection, or learning tactics offered by Genie, an animated study agent in BITS, which we will discuss in Section 5.

The primary goal of the input module is to update the Bayesian network based on evidence collected from the student, as we discussed in Section 4.1.2.

4.4 Adaptive Guidance

Using the state of the Bayesian network regarding the knowledge of the student, BITS can offer tailored pedagogical options that support the individual student. In this section, we describe three kinds of adaptive guidance that BITS can provide: *navigation support*, *prerequisite recommendations* for problem solving, and *the generation of a learning sequence* when studying a particular concept. These three adaptation methods are current issues in Web-based Intelligent Tutoring Systems [7, 8].

4.4.1 Navigation Support

The navigation menu is used to navigate through the concepts under consideration. To help the student browse the materials, BITS marks each concept with an appropriate traffic light. These traffic lights are computed dynamically from the Bayesian network and indicate the student's knowledge regarding these topics.

This method is a very efficient approach to supporting the user who is navigating through the online Websites and in helping her to find appropriate or relevant information. Brusilovsky [7] calls this technique *Adaptive Annotation*, which means that the system uses visual cues (icons, fonts, colors) as the metaphor to help the user select appropriate information.

In BITS, each concept is marked as belonging to one of the following three categories:

(i) *already known*,
(ii) *ready to learn*, and
(iii) *not ready to learn*.

A concept is considered *already known* if the Bayesian network indicates the probability $p(concept = known|evidence)$ is greater than or equal to 0.70, where the evidence is the student's knowledge on previous concepts obtained indirectly from quiz results or directly by the student replying to a query from BITS. It should be noted that the choice of 0.70 to indicate a concept is known is arbitrarily chosen.

If a concept is marked *ready to learn*, it means that all prerequisites are known by the student. In terms of the Bayesian network, the probability $p(concept = known|evidence)$ is less than 0.70 for the current knowledge concept and all of the parent concepts are already known. Please note that knowing the parent concepts does not mean the child concept is already known. On the contrary, it only means that the child concept is ready to learn, provided the current probability $p(concept = known|evidence) < 0.7$. The second point we need to note is that we only need be concerned with whether the parent concepts are already known. According to the conditional independencies encoded in the Bayesian network, given the parents of one variable v_i, v_i is conditionally independent of all non-descendants.

Finally, a concept is labelled *not ready to learn* if at least one of the prerequisites for this concept is not already known by the student; in terms of the Bayesian network, there exists at least one parent concept that is not already known.

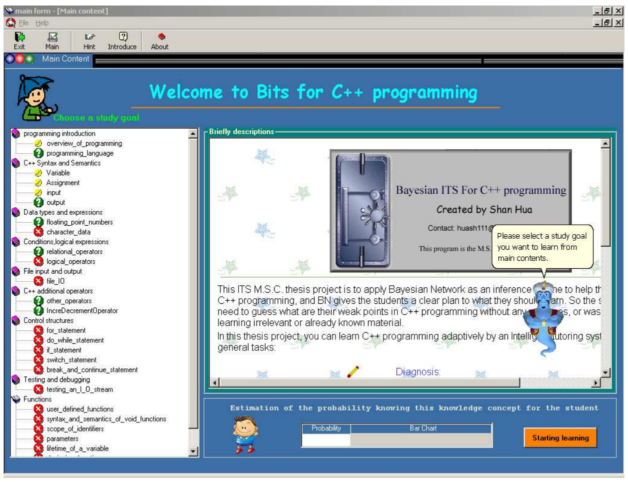
Traffic lights are employed as follows: yellow (already known), green (ready to learn), and red (not ready to learn).

When BITS is first entered, the concepts are marked with traffic lights based on the initial probabilities obtained from the Bayesian network. The opening screen shot of BITS is depicted in Figure 11. The navigation menu appears on the left, while a brief introduction to BITS is shown on the right. A student can *preview* a concept by highlighting it; BITS then displays a brief description of the concept, which explains why it is important. In the bottom right corner of Figure 11, BITS also displays the initial probability $p(concept = known)$ for the highlighted concept. This probability indicates BITS' belief that the student knows the concept that is currently highlighted.

Example 8 Recall the entry page for BITS in Figure 11. By highlighting the concept "Floating-point numbers," a student can *preview* this topic. A brief overview of "Floating-point numbers" is shown in Figure 12. *In the bottom right corner of Figure 12, BITS also displays the initial probability* $p(Floating - point\ numbers = known)$. *This probability informs BITS' belief that the student knows the concept "Floating-point numbers."*

If the student decides to study this topic, she can press the *start learning* button. If this topic belongs to a *ready to learn* concept, the lecture notes for this topic are retrieved from the database and displayed for the user.

Example 9 Recall the initial state of the navigation menu shown in Figure 11. If the student selects the *ready to learn* concept "Floating-point numbers," then BITS displays the lecture notes shown in Figure 13. After reading the notes, the student selects one of the choices at the bottom right of Figure 13. If the student indicates that she understands "Floating-point numbers," then the Bayesian network is updated and the navigation menu is again displayed (this time,



Please select a study goal you want to learn from main contents.

This ITS M.S.C. thesis project is to apply Bayesian Network as an inference ne to help th C++ programming, and BN gives the students a clear plan to what they shoul arn. So the s need to guess what are their weak points in C++ programming without an s, or was learning irrelevant or already known material.

In this thesis project, you can learn C++ programming adaptively by an Intellig utoring syst general tasks:

Figure 11. Entry page of BITS with navigation menu (left frame), where green means *ready to learn*, **yellow means** *already known*, **red means** *not ready to learn*, **and a brief introduction to BITS is provided (right frame).**

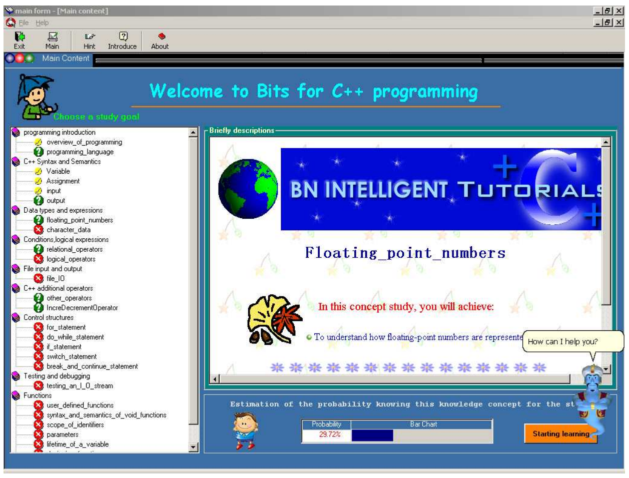
however, the concept "Floating-point numbers" is labelled as *already known*).

4.4.2 Prerequisite Recommendations

After reading the lecture notes of a *ready to learn* concept (see Section 4.4.1), a student may indicate that she does not understand the concept either by directly answering a query or indirectly through incorrect answers on the corresponding quiz (see Section 4.1.2).

In such situations, BITS is designed to present links to prerequisite concepts of this topic, such as the links to each concept in the parent set of the variables in the Bayesian network. Instead of methods that repeat the problem concept over and over, this approach is useful because it provides the flexibility to revisit the prerequisite concepts and confirm they are indeed understood. The rationale behind our method is that a student may believe that a prerequisite concept is understood when, in fact, it is not, and lacking prerequisites usually affects the student's learning performance.

Example 10 Suppose that, after reading the lecture notes for the concept "For Loop," the student indicates that she has not understood. BITS then determines the parent set of the "For Loop" node (i.e., Variable Assignment, Relational Operators, and Increment/Decrement Operators as shown in Figure 6). Finally, BITS displays the links to the lecture notes for these three concepts, as illustrated in the top right corner of Figure 14.



Welcome to Bits for C++ programming

programming introduction
- overview_of_programming
- programming_language

C++ Syntax and Semantics
- Variable
- Assignment
- input
- output

Data types and expressions
- floating_point_numbers
- character_data

Conditions,logical expressions
- relational_operators
- logical_operators

File input and output
- file_IO

C++ additional operators
- other_operators
- IncreDecrementOperator

Control structures
- for_statement
- do_while_statement
- if_statement
- switch_statement
- break_and_continue_statement

Testing and debugging
- testing_an_I_O_stream

Functions
- user_defined_functions
- syntax_and_semantics_of_void_functions
- scope_of_identifiers
- parameters
- lifetime_of_a_variable

Floating_point_numbers

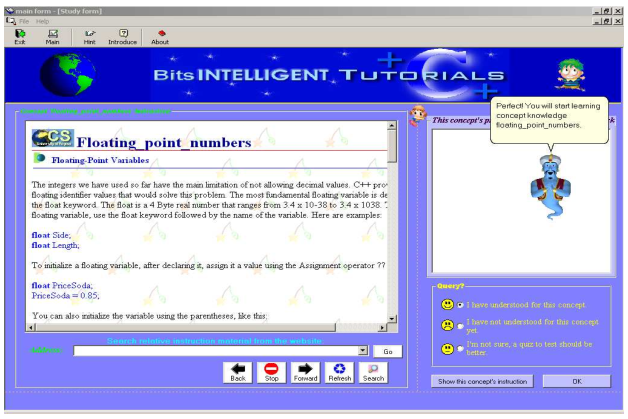
In this concept study, you will achieve:

- To understand how floating-point numbers are represente

Estimation of the probability knowing this knowledge concept for the st

Probability	Bar Chart
29.72%	

Figure 12. A student can preview the concept “File I/O" (right frame) by highlight it in the navigation menu (left frame).


main form - [Study form]
File Help
Exit Main Hint Introduce About
Bits INTELLIGENT TUTORIALS
CS University of Regina
This concept's p
Query?
Search relative instruction material from the website:
Address:
Go
Back Stop Forward Refresh Search
Show this concept's instruction
OK

Floating_point_numbers

Floating-Point Variables

The integers we have used so far have the main limitation of not allowing decimal values. C++ pro floating identifier values that would solve this problem. The most fundamental floating variable is de the float keyword. The float is a 4 Byte real number that ranges from 3.4 x 10-38 to 3.4 x 1038. floating variable, use the float keyword followed by the name of the variable. Here are examples:

```
float Side;
float Length;
```

To initialize a floating variable, after declaring it, assign it a value using the Assignment operator ??

```
float PriceSoda;
PriceSoda = 0.85;
```

You can also initialize the variable using the parentheses, like this:

Perfect! You will start learning concept knowledge floating_point_numbers.

- [✓ I have understood for this concept.]
- I have not understood for this concept yet.
- I'm not sure, a quiz to test should be better.

Figure 13. A screen shot of BITS displaying the lecture notes for the concept “Floating-point numbers".

4.4.3 Generating Learning Sequences

Students may sometimes want to learn one particular concept without having to go through every single topic previously mentioned. For example, a student may wish to learn “File I/O” for an impending exam or assignment deadline. In such a case, the student would want to learn a minimum number of concepts.

BITS meets this need by generating learning sequences. The student is allowed to select a *not ready to learn* concept in the navigation menu. In this situation, BITS displays a learning sequence for the chosen topic. In other words, all unknown ancestral concepts in the Bayesian network are revealed to the student in one proper sequence for learning. More formally, let $X = \{x_1, x_2, \ldots, x_n\}$ be the ancestral set of a selected *not ready to learn* concept in the Bayesian network. BITS will display the variables in X in accordance with the fixed ancestral numbering of the variables in the Bayesian network. That is, if concept x_i is a parent of concept x_j, then x_i will be presented before x_j, indicating x_i must be learned before x_j can be learned.

Example 11 Suppose the student selects the *not ready to learn* concept “File I/O” in the navigation menu of Figure 11; BITS displays the ancestral concepts in order, grouping by *known* and *unknown* (overview of programming, variable, Assignment, input, marked by *known*, where programming language and output are marked by *unknown*, as depicted in Figure 15). The student needs to learn *programming language*, and *output* first.

5 BITS Via the Web

In this section, we describe the features that allow BITS to be accessed via the Web, and discuss the implementation of BITS. In BITS, all course material, including lecture notes, examples and quizzes, is stored in hypermedia format. BITS is regarded as an online intelligent and interactive integrated textbook.

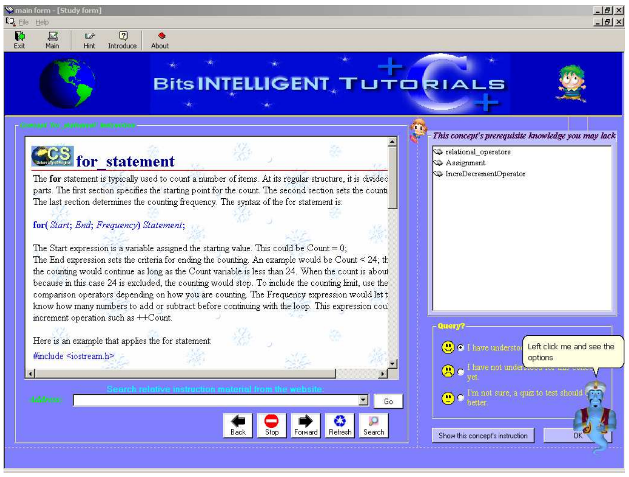
5.1 General Features of BITS

BITS is interactive because the quizzes used to test the student are not static texts as in other regular Web-based learning systems, rather they are interactions. The quizzes include: *True/False*, *Multiple Choice* and *Fill-in-the-blank* questions.

Each quiz consists of interactive Flash multimedia files combined with XML documents. More specifically, Flash multimedia files are used to format the questions displayed, while XML documents are used to describe the quiz contents, store solution keys, and validate the student’s answer.

Example 12 Recall the quiz for the concept “File I/O” in Figure 9. The XML document for this quiz is shown in Figure 16.

The question and the answer choices are described in the contents of XML elements. The correct answer for the question is stated in the value of the XML attribute *answer*. The correct answer for the question in Figure 9 is E, namely, $answer$ = “E” in the top left corner of Figure 16. This facility allows BITS to provide dynamic validation of the input answer and to proceed with appropriate action.


main form - [Study form]
File Help
Exit
Main
Hint
Introduce
About
Bits INTELLIGENT TUTORIALS
Search relative instruction material from the website:
Address:
Go
Back
Stop
Forward
Refresh
Search
Query?
Show this concept's instruction
OK

for_statement

The **for** statement is typically used to count a number of items. At its regular structure, it is dividec
parts. The first section specifies the starting point for the count. The second section sets the counti
The last section determines the counting frequency. The syntax of the for statement is:

for(*Start*; *End*; *Frequency*) *Statement*;

The Start expression is a variable assigned the starting value. This could be Count = 0;
The End expression sets the criteria for ending the counting. An example would be Count < 24; th
the counting would continue as long as the Count variable is less than 24. When the count is about
because in this case 24 is excluded, the counting would stop. To include the counting limit, use the
comparison operators depending on how you are counting. The Frequency expression would let t
know how many numbers to add or subtract before continuing with the loop. This expression cou
increment operation such as ++Count.

Here is an example that applies the for statement:

```
#include <iostream.h>
```

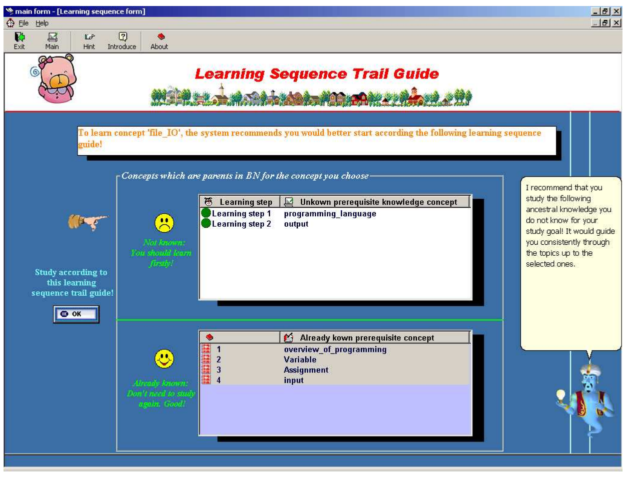
This concept's prerequisite knowledge you may lack

- relational_operators
- Assignment
- IncreDecrementOperator

I have understo
I have not under
yet.
I'm not sure, a quiz to test should
better.

Left click me and see the options

Figure 14. A screen shot of BITS displaying the links to the lecture notes for the prerequisite concepts of “For Loop."


main form - [Learning sequence form]
File Help
Exit
Main
Hint
Introduce
About
Learning Sequence Trail Guide
OK

To learn concept 'file_IO', the system recommends you would better start according the following learning sequence guide!

Concepts which are parents in BN for the concept you choose

Learning step	Unkown prerequisite knowledge concept
Learning step 1	programming_language
Learning step 2	output

Not known: You should learn firstly!

Study according to this learning sequence trail guide!

	Already kown prerequisite concept
1	overview_of_programming
2	Variable
3	Assignment
4	input

Already known: Don't need to study again. Good!

I recommend that you study the following ancestral knowledge you do not know for your study goal! It would guide you consistently through the topics up to the selected ones.

Figure 15. BITS generates a learning sequence for the concept “File I/O" in Figure 11, which is *not ready to learn*.

BITS uses HTML Web pages to represent the online instructional materials. In some cases, multimedia examples using animated Flash concepts are utilized to illustrate various abstract concepts of C++. Surveys of computer science educators suggest a widespread belief that visualization technology and multimedia play a positive and important role in student learning [49]. Multimedia provides a student with vivid images or procedures instead of abstract and rigid concepts, and it improves the student’s learning efficiency [49].

The lecture notes and quiz questions are displayed using a Web browser embedded in BITS. BITS also provides the student with the ability to access other C++ programming sites on the Web. When studying a new concept, the student can utilize the web browser provided by BITS to search external resources; BITS also recommends a set of URL links to external Websites.

Example 13 *As illustrated in the bottom left corner of Figure 13, BITS allows the user to access C++ sites found on the Web.*

Finally, it is worth mentioning that BITS includes an animated study agent *Genie* [28]. The goal of the animated study agent is to convey appropriate emotion and encouragement, thus demonstrating learning strategies to the student. The feedback given by the student agent is in the form of voice animation and dialogue boxes. Researchers in education theory have provided evidence that animated educational assistance can promote effective learning in computer-based learning environments [29]. Learning materials that incorporate interactive agents engender a higher degree of interest than similar materials that lack animated agents. Animated agents encourage various emotional and intellectual responses from the learners, and thus further promote learner motivation.

Example 14 In the bottom right corner of Figure 13, the study agent *informs* the student that she has chosen to learn the concept “Floating-point numbers,” while the study agent *recommends* that the student first learn three prerequisite concepts in Figure 15.

By providing useful and informative feedback, BITS provides a positive environment for learning.

5.2 Development Tools of BITS

The general framework of BITS was developed using Visual Studio.net [19]. As the aim for BITS is based on the Web environment, we developed the instructional materials with Macromedia Studio MX [23], a set of Web design tools, including Dreamweaver MX, Flash MX and Fireworks MX. Employing Dreamweaver MX, we developed active server pages (ASP), XML files, and HTML Web pages. We designed multimedia courseware and quiz files by employing Flash MX. We applied Fireworks MX to design pictures for the instructional material. These tools facilitate making allow BITS a rich, interactive educational environment.

The Bayesian network for BITS was implemented by MSBNx [24], the Microsoft Bayesian Network Toolkit. MSBNx is a component-based Windows application created at Microsoft Research for creating, assessing, and

```
<MainElement>
    <Question answer="E"> Which of the following is  NOT one  of the things
         a programmer must do in order to use files in a C++ program?
       <choices>
         <Items> Use a preprocessor directive to include the header file fstream.
         </Items>
         <Items> Declare each file stream in a variable declaration.</Items>
         <Items> Prepare each file for reading or writing  by calling the
                   open function.  </Items>
         <Items> Specify the name of the file stream in  each input or output
                  statement that uses  it. </Items>
         <Items> Erase the contents of each output file before running the
                  program.</Items>
      </choices>
    </Question>
</MainElement>
```

Figure 16. The XML file for the question on the concept “File I/O" in Figure 9.

evaluating Bayesian Networks. After implementation, the Bayesian network is saved in an XML-based format. Therefore, the Bayesian network is independent of the whole system's programming. This feature facilitates any changes or refining the Bayesian network in the future, i.e., extending BITS to a second course on computer programming by incorporating more sophisticated concepts, without modification of existing components of BITS.

6 Related Works

Intelligent Tutoring Systems for programming have been developed and evaluated for many years in the field of Artificial Intelligence in Education. Programming has been a very productive domain in the evolution of most aspects of the field, including student modelling, knowledge representation, and the application of sound pedagogical principles [50].

Recently, there have been several efforts to develop ITSs for programming, such as SQL-Tutor [21], an ITS for SQL programming based on Constraint-Based Modelling approach. Lisp-Tutor [2] is an ITS for teaching lisp programming based on rule-based system. JITS [50] is a prototype for an ITS for students learning to program in Java which is based on a decision tree method. While these ITSs have been effective, they pay little attention to helping the students navigating online learning materials by recommending individualized reading sequences. The main objective of these ITSs centers around problem solving support technologies by providing appropriate error feedback and updating the student model. The purpose of BITS, on the other hand, is quite different; it is to help the student *navigate* the course material. Although problem solving is an integral part of computer programming, it is outside the focus of BITS.

Villano [52] first suggested applying Bayesian networks in Intelligent Tutoring Systems. However, Martin and Vanlehn [38] explicitly state that Villano's assessments cannot communicate precisely what a student does not know and cannot identify the components of knowledge that must be taught. BITS, on the other hand, uses yellow traffic lights to indicate known concepts, green traffic lights to indicate ready to learn concepts, and red traffic lights to indicate concepts that the student is not ready to learn.

The *assessment* system proposed by Martin and Van-Lehn [38] is focused solely on assessing what a student knows. Our *intelligent tutoring system* not only assesses what a student knows, but, in addition, assists the student in navigating the unknown concepts.

Zapata and Greer [59] designed a system to visualize Bayesian student models. One advantage of graphical representations of student models is to help instructors determine the learning deficiencies for a student. However, learning the deficiencies for a student was not one of our objectives when we designed BITS.

There have been several other recent efforts to apply Bayesian networks to student modelling in Web-based tutoring systems [17, 18, 28]. Most of them involved applying Bayesian networks to help a student with problem solving support (refer to Section 2) during resolution of one task [13, 14, 28, 36, 37]. For example, Conati et al. [14] developed an intelligent tutoring system, Andes, for physics. The primary objective of that system is to help the student learn how to *problem solve*, while the focus of BITS is quite different.

It is worth mentioning that Jameson [25] reviewed several frameworks for managing uncertainty in Intelligent Tutoring Systems, including Bayesian networks, the Dempster-Shafer theory of evidence, and fuzzy logic. As Pearl [41] has shown, Bayesian neworks have certain advantages over the other two frameworks. This influenced our decision to use Bayesian networks for uncertainty management in BITS.

7 Concluding Remarks

This section concludes the paper by providing a summary of the work done and stating our tentative proposals for future work.

7.1 Conclusions

Web Intelligence explores the practical applications of Artificial Intelligence to the next generation of Web-empowered systems [54]. In this paper, we proposed a Web-based intelligent tutoring system (BITS) for computer programming that utlizes Bayesian networks, a proven framework for uncertainty management in Artificial Intelligence [41, 55, 56]. We discussed a new architecture for designing an ITS 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. We discuss in detail how to use a Bayesian network in BITS for modelling and inference purposes.

Applying Bayesian networks in Intelligent Tutoring Systems to assess the student's knowledge level was first suggested by Villano in 1992 [52]. However, Villano's assessments cannot communicate precisely what a student does not know and cannot identify the components of knowledge that must be taught [38]. BITS, on the other hand, marks each concept with an appropriate traffic light to indicate the student's knowledge regarding these topics and helps the student select appropriate instructional content to study (see Section 4.4.1). The *assessment* system proposed by Martin and Vanlehn [38] is focused solely on assessing what a student knows (see Section 6). Our *intelligent tutoring system* not only assesses what a student knows, but, in addition, helps the student navigate the unknown concepts (see Section 4.4.3).

Currently, most Web-based learning systems are static HTML Web pages and are simply a copy of regular textbooks. These forms of instructional materials suffer from two major shortcomings [6]: firstly, they are not interactive, since students can only passively read the educational materials; secondly, they are not adaptive (see Section 1.2). BITS provides remote access to hypermedia-structured learning materials which include instruction notes, tests, and examples. BITS can help a student navigate the online course materials using traffic lights (see Section 4.4.1); it can also recommend learning goals when a particular concept is not understood (see Section 4.4.2). Finally, when a student wants to learn a particular concept without learning all of the previous concepts, BITS can present the minimum prerequisite knowledge needed in order to understand the desired concept in the proper learning sequence (see Section 4.4.3). BITS has been implemented and was recently used in the summer 2004 session of CS110, the initial computer programming course at the University of Regina. While the feedback collected from the students was limited, the result was very positive. Empirical studies have shown that individual one-on-one tutoring is the most effective mode of teaching and learning, and that Intelligent Tutoring Systems offer a unique technology to implement computer-assisted one-on-one tutoring [5].

BITS is significant since it has the potential to be extended to various computer programming courses. Furthermore, it is important to both Web-based learning and traditional settings. We believe, thus, that BITS will be useful for supporting instructors teaching computer programming in their institutions. BITS is especially important for institutions where there are more students wishing to learn to program and where it is difficult to provide personalized instruction that they need [31]. This research work here, together with [27, 28, 34, 36, 42, 53], explicitly demonstrates the practical usefulness of Bayesian networks for Web Intelligence.

7.2 Future Work

BITS will be used in future offerings of CS110. Data will be collected to formally evaluate whether BITS improves the learning results for students who used BITS, compared to those who did not.

We plan to extend BITS in the future to a second course on computer programming by incorporating more sophisticated concepts of C++ into BITS, such as inheritance and pointers. We also hope to extend BITS by incorporating other programming languages like Java and Perl. Since the Bayesian network implemented in BITS is independent of the materials of CS110 and is separated from the system's programming work, it will be possible to refine or reconstruct the Bayesian network efficiently.

A second enhancement of BITS is that currently, for one knowledge concept, BITS distinguishes between two states, *known* and *not known*. This might be not very useful in practical applications, since sometimes the student may only partially know a concept. In such cases, we can add a state somewhere between *known* and *not known*. Thus, we can describe the student's knowledge on a specific concept in terms of finer gradations.

In future work, moreover, we hope to add problem solving support to BITS. Problem solving is another important part of computer programming. Currently, however, BITS does not provide this. We plan to enhance BITS and integrate this kind of guidance in BITS. We will also concentrate on refining the current modules and integrating all these modules with the special needs of different teachers to improve the adaptability of BITS as a Web-based learning tool.

References

[1] J.R. Anderson, F.G. Conrad and A.T. Corbett, Skill acquisition and the LISP tutor, Cognitive Science, 13 (1989) 467-505.

[2] J.R. Anderson and B.J. Reiser, The LISP tutor, Byte, 10(4) (1985) 159-175.

[3] A. Barr, M. Beard and R.C. Atkinson, The computer as tutorial laboratory: the Stanford BIP project, International Journal on the Man-Machine Studies, 8(5) (1976) 567-596.

[4] F. Bennett. *Computers as Tutors: Solving the Crisis in Education*, Sarasota, FL: Faben Inc. Publishers, 1999.

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

[6] P. Brusilovsky, Adaptive and intelligent technologies for Web-based education, Special Issue on Intelligent Systems and Teleteaching, 4 (1999) 19-25.

[7] P. Brusilovsky and M.T. Maybury, From adaptive hypermedia to adaptive Web, Communications of the ACM, Special Issue on the Adaptive Web, 45(5) (2002) 31-33.

[8] P. Brusilovsky and E. Schwarz, User as student: towards an adaptive interface for advanced Web-based applications, in: Proceedings of 6th International Conference on User Modeling, Sardinia, Italy, 1997, pp. 177-188.

[9] B.G. Buchanan and E.H. Shortliffe, Rule-based expert systems: the MYCIN experiments of the stanford heuristic programming project, Addison-Wesley, 1985.

[10] H.L. Burns and C.G. Capps, Foundations of intelligent tutoring systems, Foundations of Intelligent Tutoring Systems, Lawrence Erlbaum Associates, Chapter 3, (1988) 55-78.

[11] C.J. Butz, S. Hua and R.B. Maguire, A Web-based Intelligent Tutoring System for Computer Programming, in: Preceedings of the IEEE/WIC/ACM Conference on Web Intelligence, Beijing, China, 2004, pp. 159-165.

[12] E. Castillo, J. Gutierrez and A. Hadi, Expert Systems and Probabilistic Network Models, Springer, 1997.

[13] J.A. Collins, J.E. Greer and S.X. Huang, Adaptive assessment using granularity hierarchies and Bayesian nets, in: Proceedings of the 3rd International Conference on Intelligent Tutoring Systems, Montreal, Canada, 1996, pp. 569-577.

[14] C. Conati, A. Gertner and K. Vanlehn, Using Bayesian networks to manage uncertainty in student modeling, User Modeling and User-Adapted Interaction, 12(4) (2002) 371-417.

[15] N. Dale, C. Weems and M. Headington, Programming and Problem Solving with C++, Jones and Bartlett Publishers, 2000.

[16] R. Dechter, Bucket elimination: a unifying framework for probabilistic inference, in: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, Portland, Oregon, 1996, pp. 211-219.

[17] A. Gertner, C. Conati and K. Vanlehn, Procedural help in Andes: generating hints using a Bayesian network student model, in: Proceeding of 15th National Conference on Artificial Intelligence, Madison, Wisconsin, 1998, pp. 106-111.

[18] N. Henze and W. Nejdl, Adaptation in open corpus hypermedia, International Journal of Artificial Intelligence in Education, 12 (2001) 325-350.

[19] http://msdn.microsoft.com/vstudio/, January 9, 2005.

[20] http://www.ai.mit.edu/∼murphyk/Bayes/bnsoft.html, April 28, 2004.

[21] http://www.cosc.canterbury.ac.nz/∼tanja/sql-tut.html, April 28, 2004.

[22] http://www.cs.mdx.ac.uk/staffpages/serengul/Traditional.Computer.Aided.Learning.Systems.htm, April 28, 2004.

[23] http://www.macromedia.com/, July 30, 2004.

[24] http://www.research.microsoft.com/adapt/MSBNx/, April 28, 2004.

[25] A. Jameson, Numerical uncertainty management in user and student modelling: an overview of systems and issues, User Modelling and User-Adapted Interaction, 5(3- 4) (1995) 193-251.

[26] F.V. Jensen, S.L. Lauritzen and K.G. Olesen, Bayesian updating in causal probabilistic networks by local computations, Computational Statistics Quaterly, 4 (1990) 269-282.

[27] J. Ji, L. Zheng and C. Liu, The intelligent electronic shopping system based on Bayesian customer modelling, in: Proceedings of First Asia-Pacific Conference on Web Intelligence, Maebashi City, Japan, 2001, pp. 574-578.

[28] W.L. Johnson. Pedagogical agents for Web-based learning, in: Proceedings of First Asia-Pacific Conference on Web Intelligence, Maebashi City, Japan, 2001, pp. 43-44.

[29] W.L. Johnson and E. Shaw, Using agents to overcome difficulties in Web-based courseware, in: Proceedings of the workshop on Intelligent Educational

Systems on the World Wide Web, 8th World Conference of the AIED Society, Kobe, Japan, 1997, pp. 283-290.

[30] G.J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice-Hall, 1995.

[31] K.R. Koedinger, Cognitive tutors, Smart machines in education, (2001) 145-167.

[32] K.R. Koedinger, J.R. Anderson, W. Hadley and M. Mark, Intelligent tutoring goes to school in the big city, International Journal of Artificial Intelligence in Education, 8 (1997) 30-43.

[33] S.L. Lauritzen and D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, Journal of Royal Statistical Society, Series B, 50(2)(1988) 157-224.

[34] S. Lee, C. Sung and S. Cho, An effective conversational agent with user modeling based on Bayesian network, in: Proceedings of First Asia-Pacific Conference on Web Intelligence, Maebashi City, Japan, 2001, pp. 428-432.

[35] Z. Li, B. D'Ambrosio, Efficient inference in Bayes networks as a combinatorial optimization problem, International Journal of Approximate Reasoning, 11(1) (1994) 55-81.

[36] C. Liu, L. Zheng, J. Ji, C. Yang and W. Yang, Electronic homework on the WWW, in: Proceedings of First Asia-Pacific Conference on Web Intelligence, Maebashi City, Japan, 2001, pp. 540-547.

[37] J. Martin and K. VanLehn, A Bayesian approach to cognitive assessment, Cognitively Diagnostic Assessment, (1995) 141-165.

[38] J. Martin and K. Vanlehn, Student assessment using Bayesian nets, International Journal of Human-Computer Studies, 42 (1995) 575-591.

[39] T. Murray, Authoring intelligent tutoring systems: an analysis of the state of the art, International Journal of Artificial Intelligence in Education, 10 (1999) 98-129.

[40] N. Nilson, Artificial Intelligence: A New Synthesis, Morgan Kaufmann, 1998.

[41] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.

[42] V. Robles, P. Lfarrañaga, E. Menasalvas, M.S. Pérez and V. Herves, Improvement of Naive Bayes collaborative filtering using interval estimation, in: Proceedings of 2nd Annual Asia-Pacific Conference on Web Intelligence, Halifax, Canada, 2003, pp. 168-174.

[43] K.G. Schulze, R.N. Shelby, D.J. Treacy, M.C. Wintersgill, K. Vanlehn and A. Gertner, Andes: an intelligent tutor for classical physics, The Journal of Electronic Publishing, University of Michigan Press, Ann Arbor, MI, 6(1) (2000).

[44] G. Shafer, Probabilistic Expert Systems, Society for Industrial and Applied Mathematics, 1996.

[45] G. Shafer and P.P. Shenoy, Probability Propagating, Annals of Mathematics and Artificial Intelligence, 2(1990) 327-352.

[46] V.J. Shute and R. Glaser, A large-scale evaluation of an intelligent discovery world: Smithtown, Interactive Learning Environments, 1(1) (1990) 51-77.

[47] V.J. Shute and J. Psotka, Intelligent tutoring systems: past, present, and future, Handbook of Research on Educational Communications and Technology, Macmillan, New York, (1996) 570-600.

[48] D. Sleeman and J.S. Brown, Introduction: Intelligent tutoring systems, Intelligent Tutoring Systems, (1982) 1-10.

[49] J. Stasko, A. Badre and C. Lewis, Do algorithm animations assist learning? An empirical study and analysis, in: Proceedings of the SIGCHI conference on Human factors in computing systems, Amsterdam, The Netherlands, 1993, pp. 61-66.

[50] E.R. Sykes and F. Franek, A Prototype for an Intelligent Tutoring System for Students Learning to Program in Java, in: Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies, Athens, Greece, 2003, pp. 485-486.

[51] M. Urban-Lurain, Intelligent tutoring systems: An historic review in the context of the development of artificial in-telligence and educational psychology, http://www.cse.msu.edu/rgroups/cse101/ITS/its.htm, August 3, 2004.

[52] M. Villano, Probabilistic student models: Bayesian belief networks and knowledge space theory, in: Proceedings of 2nd International Conference on Intelligence Tutoring System, Montreal, Canada, 1992, pp. 491-498.

[53] Y. Wang and J. Vassileva, Bayesian network-based trust model, in: Proceedings of 2nd Annual Asia-Pacific Conference on Web Intelligence, Halifax, Canada, 2003, pp. 372-378.

[54] Web Intelligence Consortium, April 1, 2004, http://wi-consortium.org/

[55] 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) (2001) 395-415.

[56] 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) (2000) 785-805.

[57] C.W. Woo, Instructional Planning in an Intelligent Tutoring System: Combining Global Lesson Plans with Local Discourse Control, Ph.D. Thesis, Illinois Institute of Technology, Chicago, IL, 1991.

[58] J.T. Yao and Y.Y. Yao, Web-based support systems, in: Proceedings of the WI/IAT Workshop on Applications, Products and Services of Web-based Support Systems, 2003, pp. 1-5.

[59] J.D. Zapata and J.E. Greer, Visualizing and inspecting Bayesian belief models, in: Proceedings of International Joint Conference on Artificial Intelligence, Seattle, August, 2001, pp. 47-49.

[60] N.L. Zhang, D. Poole, A simple approach to Bayesian network computations, in: Proceedings of The 10th Canadian Conference on Artificial Intelligence, Banff, Alberta, Canada, 1994, pp. 171-178.

Appendix

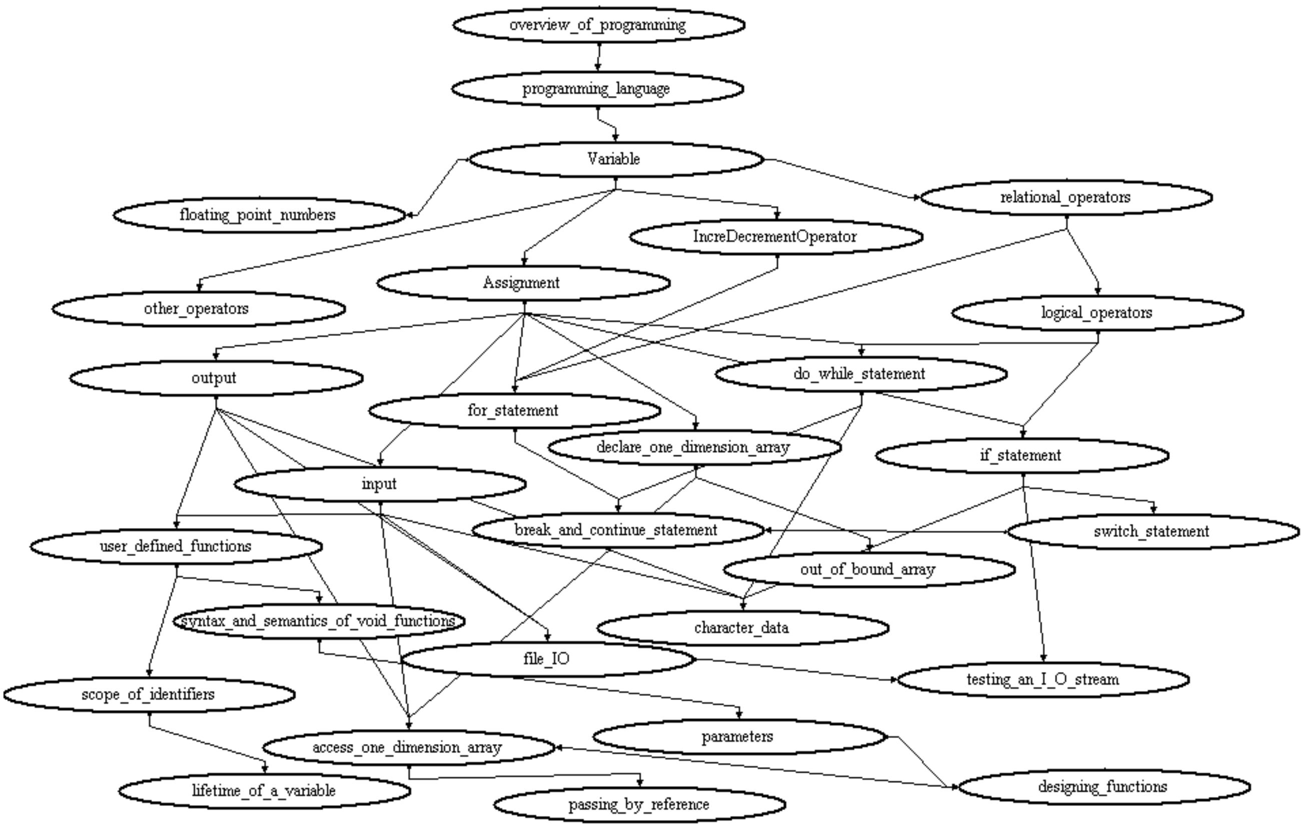


Figure 17. The entire DAG implemented in BITS.