Incorporating Bayesian Inference into Adaptation Rules in AHA architecture

Loc Nguyen[1]

[1]University of Science, Ho Chi Minh city, Vietnam

Key words: *adaptive learning, AHS, Bayesian inference, AHA!*

Abstract:

Adaptive Hypermedia System (AHS) aims to provide users the adaptation effect based on their characteristics. In other words, AHS ensures that the links that are offered and the content of the information pages are adapted to each individual user. AHA! [1] developed by Debra is an open Adaptive Hypermedia Architecture that is suitable for many different applications; it aims to generic purpose. The architecture of AHA based on Dexter Reference Model [2] have some prominences but the inside user model is built up by overlay method in which the domain is decomposed into a set of elements and the overlay is simply a set of masteries over those elements. Although overlay model is easy to represent user information, there is no inference mechanism for reasoning out new assumptions about user. So we propose a new way to incorporating Bayesian inference into AHA so that it is able to improve modeling functionality in AHA.

1 AHA Architecture

AHA architecture has three layers: runtime layer, storage layer, within-component layer

- The *runtime layer* represents the user interface (UI). It is not necessary to outline exactly what the UI should do but the abstraction of its functions is described in the *presentation specifications*. For example, the presentation specifications can determine whether the adaptive paragraph has bold font, whether the link should be hidden.
- Storage layer is the core of AHA architecture. This layer has three main model with respective functions: *domain model, user model* and *adaptation model*. The domain model is conceptual representation of the application domain. The user model contains user characteristics. Overlay model is often set up by overlay method in which domain model is the mastered subset of domain model. The adaptation model describes how the domain model and user mode are combined to generate adaptations such as the presentation adaptation, link adaptation.
- The *within-component layer* describes the internal, implementation-specific data objects which can be accessed by means of *anchoring*.

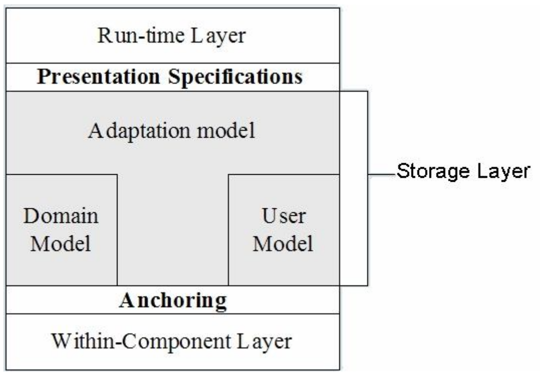

Figure 1: AHA architecture

Domain model

The domain model consists of concepts which are topics in the application domain. Each concept has:

- An unique identifier (c-id)
- A description structure called "*concept information*" (c-info) is constituted of three fixed part, namely a *sequence of anchors*, a *presentation specification* and a set of *attribute-value pairs*.

The sequence of anchors describes substructures within a concept. These can be used as anchor point for links such as the source or destination of links. The presentation specification navigates the runtime layer how to display concept's content. The attribute-value pairs are arbitrary. Each of them tells us optional information of concept and depends on idea of experts. Some usual attributes are showed in below:

- *access*: when a concept is accessed, this attributed is triggered. Attribute "access" is the starting point of process of executing adaptation rules.
- *knowledge*: amount of knowledge that user are mastered over concept.
- *visited*: this attribute indicate whether user visited the page associated with concept or not

User model

User model contains important information about user such as: knowledge, learning style, goals, background. User model is used as basis of adaptive process. In overlay method, user model is the subset of domain model. Namely, domain is decomposed into a set of concepts and overlay model is simply the set of masteries over these concepts. In this paper, user model is implicated as overlay model.

Adaptation model

The adaptation model (AM) is responsible for associating user model and domain model to generate adaptation. AM contains a set of adaptive rules used to tailor learning concept & material to user characteristics in user model. Rules are categorized into two classes associated two purposes:

- Updating user model
- Defining adaptation effect by setting presentation specification. For example: if user is mastered over concept "Control Structure", he/she should be recommended concept "Loop" in programming language course.

In the overlay model, with each concept in domain model, there is corresponding concept in user model with the same name. So the expression "*concept.attribute*" refers both the domain model attribute and user model attribute. The adaptation rules whose purpose is to update user model are called *event-condition-action* (ECA) rules because:

- They are triggered by an *event*, e.g. users access a concept by following a link

- The *condition* which is Boolean expression is evaluated. The expression has terms which are domain/user model attribute in form "*concept.attribute*".
- When the condition is satisfied, the *action* is executed. The action performs an update to attribute value and can be trigger another rule. This is propagation mechanism.

For example, the ECA rule "when the concept *C* is accessed and it was visited before, its attribute *knowledge* is set to be 100%" is interpreted as below:

Event:	access(C)
Condition:	C.visited = true
Action:	C.knowledge = 100%

The adaptation rules whose purpose is top define adaptation effect are called condion-action (CA) rules. The have main responsibility for showing adaptive presentation:

- CA rules have no event. They are checked when the engine need to deliver learning objects to user.
- The condition in the form of Boolean expression in which terms are attributes is evaluated similarly to ECA rules.
- *Action* results in adaptive effect according to presentation specification.

For example, the CA rule "If user knowledge about concept C reaches or exceeds 50% then user is provided advanced subjects about C. Otherwise, user should pay attention to basic explanations".

Condition:	C.knowledge > 50
Action:	True status: providing advanced subjects about C
	False status: providing basic explanations about C

This rule can be interpreted in XML form

```
<if expr="C.knowledge > 50">
<block>
        Here advanced subjects about C for advanced user
</block>
<block>
        Here basic explanations about C for novice
</block>
</if>
```

2 Incorporating Bayesian inference into adaptation rules

2.1 Bayesian inference and Bayesian network

Note that Bayesian inference is a mathematical tool in which current belief of given hypothesis is changed by collecting evidences and reasoning.

$P(H|E) = \frac{P(E|H)*P(H)}{P(E)}$ where

- H is probability variable denoting a hypothesis existing before evidence.
- *E* is also probability variable notating an observed evidence
- *P(H)* is *prior probability* of hypothesis. It is also hypothesis' initial value
- *P(H|E)*, conditional probability of H with given E, is called *posterior probability*
- *P(E|H)* is conditional probability of occurring evidence E when hypothesis was true
- *P(E)* is probability of occurring evidence E together all mutually exclusive cases of hypothesis

Bayesian network is the directed acyclic graph in which nodes represent the variables (being hypothesis or evidence) and arcs represent dependency relationship between nodes. The strengths of these dependencies are quantified by conditional probabilities. So, each node has the conditional probability table (CPT). Every node has two possible values: 1 (mastered) and 0 (not mastered) representing the user's mastery of knowledge. Bayesian network use Bayesian inference as the main reasoning mechanism. However, the user model in AHA is overlay model (see section 1). We need to combine overlay model and Bayesian network in order to incorporate Bayesian inference into adaptation rules. The combination between overlay model and Bayesian network is done through following steps

- The structure of overlay model is translated into BN, each user knowledge element becomes an variable in BN
- Each prerequisite relationship between domain elements in overlay model becomes a conditional dependence assertion signified by CPT of each variable in Bayesian network

2.2 Forward reasoning and backward reasoning

There are two techniques of deductive logic: forward reasoning and backward reasoning:

- Suppose a student is recommended to learn concept "Class & Object" before concept "Interface & Package" in Java course. There is the prerequisite relationship in which "Class & Object" is the precondition of "Interface & Package". Namely, the *"recommended"* attribute of "Interface & Package" become *true* if and only if the *"knowledge"* attribute of "Class & Object" reaches 50%. Whenever the *"knowledge"* attribute changes its values, the *"recommended"* attribute is checked and can be re-assigned new value (*"true"*). This technique is called forward reasoning because the *"recommended"* attribute is determined as soon as it is possible and thus before it is actually needed, regardless of user requirement
- Otherwise, the *"recommended"* attribute is merely determined when students require to learn "Interface & Package", meaning we check whether the *"knowledge"* attribute reaches 50%. If user does not ask, *"recommended"* attribute is not considered even the *"knowledge"* is changed. This technique is called backward reasoning because the attribute is not pre-computed but it is tracked back when actually needed.

Forward reasoning and backward reasoning have both advantages and drawbacks:

- The strong point of forward reasoning is that condition is evaluated immediately by checking the attribute value because it was already stored. That respond time is very fast is very useful for real-time applications. Otherwise, the drawback is that the attributes are changed many times even they are not considered yet (before actually needed).
- The advantage of backward reasoning is to avoid the drawback of backward reasoning. The condition is only checked when actually needed. But drawback is that checking attribute takes more time than forwarding reasoning because it is necessary to perform more operations in reasoning process.

2.3 Dummy attribute and backward reasoning

Suppose Java course is constituted of three concepts considered as knowledge variables whose links are dependency relationships. Additionally, there are two evidence variables: "*Questions > 10*" and "*Exercise: set up the class Human* ". That learner asks more than 10 questions is to tell how much her/his amount of knowledge. Like that, evidence "Exercise: set up..." proves whether or not he/she understands concept "Class & Object". The number (in range 0...1) that measures the relative importance of each prerequisite or evidence is defined as the weight of arc from parent node to child node. All weights concerning the child variable are normalized and used to build up its CPT.

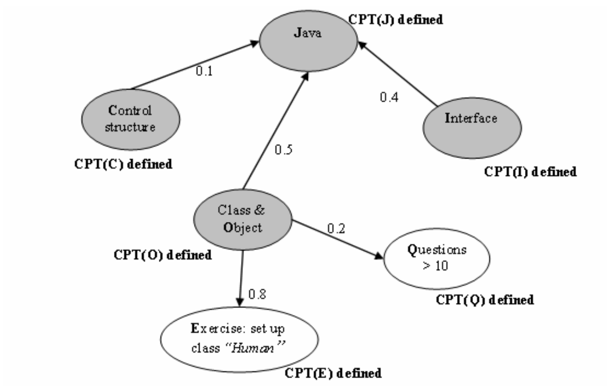

Figure 2: AHA architecture

Note that five concepts O, I, J, E, Q denote knowledge variables (and evidences) *Class & **O**bject, **J**ava, **I**nterface, "Exercise: set up…"* and *"Questions > 10"* respectively. Concepts E, Q have the new attribute *is_evidence* indicating whether E, G are fit to become evidences or not. Concepts O, I, J have the new dummy attribute *do$bayes$infer*. It called "dummy" because attribute *do$bayes$infer* does not exists actually in user model. It is only used as the denotation for backward reasoning with Bayesian network inference. Namely, *J. do$bayes$infer* is the posterior probability tell us how mastered leaner is over the concept *J*.

We defined two ECA rules for updating evidences "*Questions > 10*", "*Exercise: set up…* " and one CA rule setting presentation specification.

- The 1st ECA rule: If user asks more than 10 questions then the attribute *Q. is_evidence* is set to true.
- The 2nd ECA rule: If user does exercise "*Exercise: set up…* " the the attribute *E. is_evidence* is set to true.
- The 3rd CA rule: If the probability of user knowledge about concept Java reaches or exceeds 0.5 then user is provided advanced subjects about Java. Otherwise, user should pay attention to basic explanations. The probability of user knowledge about concept Java is denoted as dummy attribute *J. do$bayes$infer*

	Event	Condition	Action
Rule 1st	access(Q)	Q.visited > 10	Q.is_evidence = true
Rule 2nd	access(E)	E.visited > 10	E.is_evidence = true
Rule 3rd		*J. do$bayes$infer*>=0.5	True status: providing advanced subjects about Java False status: providing basic explanations about Java

The 3rd rule is translated in XML form:
<if expr="J. do$bayes$infer > 0.5">
<block>
        Here advanced subjects about Java for advanced user
</block>
<block>
        Here basic explanations about Java for novice
</block>
</if>

In 3rd rule, attribute *J.do$bayes$infer* is not stored and checked instantly but whenever adaptive engine requires to determine its value, it will be tracked back and computed by Bayesian inference. This is backward reasoning. For example, after 1st and 2nd rules are executed, concepts Q and E become actual evidence.

J.do$bayes$infer = $\Pr(J=1 \mid Q=1, E=1) = \dfrac{\sum_{O,I} \Pr(O, I, J=1, E=1, Q=1)}{\sum_{O,I,J} \Pr(O, I, J, E=1, Q=1)}$

Where $\Pr(O, I, J, E, Q) = \Pr(O) * \Pr(I) * \Pr(J \mid O, I) * \Pr(E \mid O) * \Pr(Q \mid O)$ is the global joint probability distribution and *J.do$bayes$infer* is posterior probability that represents user's knowledge about J.

3 Conclusion

Our method is to use dummy attribute to execute backward reasoning. Whenever it is required to compute adaptation, the dummy attribute of a domain element (variable) denotes the posterior probability representing user's knowledge about this domain element. Then adaptation rules will refer to such dummy attribute in order to decide adaptation strategies. It is possible to extend our method into other inferences such as neural network, hidden Markov model…and there is no need to change main technique accepts that it is required to add new dummy attributes to domain element.

References:

[1] Paul De Bra, Licia Calvi. AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia, vol. 4, pp. 115-139, Taylor Graham Publishers, 1998

[2] Wu, H. A Reference Architecture for Adaptive Hypermedia Applications. PhD thesis, Eindhoven University of Technology, ISBN 90-386-0572-2, 2002

Author(s):

Loc Nguyen
University of Natural Science, Faculty of Information Technology
Address: 227 Nguyen Van Cu, Ho Chi Minh city, Vietnam
Email: ng_phloc@yahoo.com