

Logically Characterizing Adaptive Educational Hypermedia Systems

Nicola Henze and Wolfgang Nejdl

University of Hannover
ISI – Knowledge-Based Systems
Appelstr. 4
D-30167 Hannover
`{henze,nejdl}@kbs.uni-hannover.de`
<http://www.kbs.uni-hannover.de/~{henze,nejdl}>

Abstract

Currently, adaptive educational hypermedia systems (AEHS) are described with nonuniform methods, depending on the specific view on the system, the application, or other parameters. There is no common language for expressing functionality of AEHS, hence these systems are difficult to compare and analyze. In this paper we investigate how a logical description can be employed to characterize adaptive educational hypermedia. We propose a definition of AEHS based on first-order logic, characterize some AEHS due to this formalism, and discuss the applicability of this approach.

Contents

1	Motivation	4
2	Towards a Logic-Based Definition of AEHS	5
2.1	DOCS: The Document Space	5
2.2	UM: The User Model	6
2.3	OBS: The Observations	6
2.4	AC: The Adaptation Component	7
2.5	Definition of Adaptive Educational Hypermedia Systems	7
3	Examples	8
3.1	A very simple AEHS	8
3.1.1	Simple: Document Space	8
3.1.2	Simple: User Model	8
3.1.3	Simple: Observations	8
3.1.4	Simple: Adaptation Component	8
3.2	A very simple AEHS - Extension 1	9
3.2.1	Simple 1: Document Space	9
3.2.2	Simple 1: User Model	9
3.2.3	Simple 1: Observations	9
3.2.4	Simple 1: Adaptation Component	9
3.3	A very simple AEHS - Extension 2	10
3.3.1	Simple 2: Document Space	10
3.3.2	Simple 2: User Model	10
3.3.3	Simple 2: Observations	10
3.3.4	Simple 2: Adaptation Component	10
3.4	Summary of first three (artificial) examples	11
3.5	NetCoach	13
3.5.1	NetCoach: DocumentSpace	13
3.5.2	NetCoach: Observations	14
3.5.3	NetCoach: User Model	15
3.5.4	NetCoach: Adaptation Component	16
3.6	ELM-ART II	18
3.6.1	ELM-ART II: Document Space	18
3.6.2	ELM-ART II: Observations	19
3.6.3	ELM-ART II: User Model	19
3.6.4	ELM-ART II: Adaptation Component	20
3.7	Interbook	21
3.7.1	Interbook: Document Space	21
3.7.2	Interbook: Observations	22
3.7.3	Interbook: User Model	22
3.7.4	Interbook: Adaptation Component	23
3.8	KBS Hyperbook	25
3.8.1	KBS Hyperbook: Document Space	25
3.8.2	KBS Hyperbook: Observations	26
3.8.3	KBS Hyperbook: User Model	26
3.8.4	KBS Hyperbook: Adaptation Component	27
3.9	Summary of four exemplary described AEHS	29

4	Overview: Metadata required in the studied AEH systems	32
4.1	NetCoach	32
4.1.1	Metadata:	32
4.1.2	User's Observations	32
4.2	ELM-ART II	32
4.2.1	Metadata	32
4.2.2	Observations	33
4.3	Interbook	33
4.3.1	Observations	33
4.4	KBS Hyperbook	33
4.4.1	Metadata	33
4.4.2	Observations	33
5	Discussion	34
6	Conclusion	34

Logically Characterizing Adaptive Educational Hypermedia Systems

Nicola Henze and Wolfgang Nejdl

April 9, 2003

1 Motivation

This paper aims at developing a logical characterization of adaptive educational hypermedia and web-based systems (AEHS). AEHS have been developed and tested in various disciplines and have proven their usefulness for improved and goal-oriented learning and teaching. However, these systems normally come along as stand-alone systems - proprietary solutions have been investigated, tested and improved to fulfill specific, often domain-dependent requirements. So far, there has been no attempt to define a common language for describing AEHS. We claim that such a shared language will support the analysis and comparison of AEHS, and, in addition, a comprehensible description of AEHS will encourage an extended use of adaptive functionalities in e-Learning. This is especially important with respect to the Semantic Web [16], and, associated, the Adaptive Web [4] which knows like a personal agent the specific requirements of a user, takes goals, preferences or the actual context into account in order to optimize the access to electronic information.

Bringing personalization to the Web requires an analysis of existing adaptive systems, and of course this holds for the special case of e-learning and education. In this paper, we propose a component-based definition of adaptive educational hypermedia systems. A functionality-oriented definition of adaptive hypermedia has been given by Brusilovsky, 1996 [1].:

Definition 1 (Adaptive hypermedia system) *"By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user."*

The component-based definition proposed in this paper is motivated by Reiter's theory of diagnosis [13] which settles on characterizing systems, observations, and diagnosis in first-order logic (FOL). We decompose adaptive educational hypermedia systems into basic components, according to their different roles in the system: Each adaptive (educational) hypermedia system is obviously a hypermedia system, therefore it makes assumptions about documents and their relations in a *document space*. It uses a *user model* to model various characteristics of individual users or user groups. During runtime, it collects *observations* about the user's interactions. Based on the organization of the underlying document space, the information from user model and from the system's observation, *adaptive functionality* is provided. This paper is organized as follows: In the next section we give a first description of the components of an AEHS and explain their roles and functionality with examples. We then give a definition of AEHS based on FOL. Due to this formalization, an artificial AEHS with few adaptive functionalities is described in section 3, and four examples of existing AEHS. A synopsis of the results is given in section 3.9. We

conclude with a discussion about the results of our logic-based characterization of AEHS.

2 Towards a Logic-Based Definition of AEHS

In this section we will first give a description of the components in AEHS and their roles. Afterwards we will give a formal definition of adaptive educational hypermedia systems based on first-order logic. We claim that an Adaptive Educational Hypermedia System (AEHS) is a Quadruple

$$(\text{DOCS, UM, OBS, AC})$$

with

DOCS: Document Space belonging to the hypermedia system in question as well as information associated to this document space. This associated information might be *annotations* (e.g. metadata attributes, usage attributes, etc.), *domain graphs* that model the document structure (e.g. a part-of structure between documents, comparable to a chapter - section - subsection - hierarchy), or *knowledge graphs* that describe the knowledge contained in the document collections (e.g. domain ontologies).

UM: User Model: stores, describes and infers information, knowledge, preferences etc. about an individual user (might share some models with DOCS). The observations OBS are used for updating the user model UM. Examples of user models are overlay models where the user's state of knowledge is described as a subset of an expert's knowledge of the domain. Student's lack of knowledge is derived by comparing it to the expert's knowledge. A stereotype user modeling approach classifies users into stereotypes: Users belonging to a certain class are assumed to have the same characteristics.

OBS: Observations about user interactions with the AEHS. Here, everything about the runtime behavior of the system concerning user interactions is contained. Examples are observations whether a user has visited a document, or visited document for some amount of time, etc. Other examples are rules for compiling e.g. quizzes for testing a user's knowledge on some subject, etc.

AC: Adaptation Component: rules for *adaptive functionality* (e.g. whether to suggest a document for learning, or for generating reasonable learning paths, etc.), rules for *adaptive treatment* (e.g. sorting the links leading to further documents according to their usefulness for a particular user, etc.), etc.

To formalize this above definition let's gain a deeper insight into these components:

2.1 DOCS: The Document Space

The objects of discourse in the document space are the *documents*, and, if applicable, the knowledge *topics*. Their equivalent in the logical description are the atoms: the *document identifier* (*doc_id*) or *topic identifier* (*topic_id*) respectively.

Domain graphs (or knowledge graphs) are expressed as predicates that state the relations between the documents (or topics). For formalizing the part-of domain graph mentioned as an example in the previous section, we define predicates like

$$\text{part_of}(\text{doc_id}_1, \text{doc_id}_2) .$$

Another example is the *prerequisite* relation between documents stating which documents need to be learned before a certain document can be studied:

preq(doc_id₁, doc_id₂) .

Some AEHS use a separate knowledge graph to express relations about knowledge topics. These topics normally do not correspond one-to-one to the documents. If a separate knowledge graph exists, this graph will be expressed by several predicates as well. E.g., a taxonomy on topics will be expressed by predicates like

is_a(topic_id₁, topic_id₂) .

A further example are learning dependencies modeled on topics:

is_dependent(topic_id₁, topic_id₂) .

2.2 UM: The User Model

The user model expresses, derives and draws conclusions about the characteristics of users. This might be done by modeling each individual user or by modeling typical groups that represent users with similar behavior, requirements, etc. (so called *stereotypes*). Objects of discourse in the user model are the *user* which are logically expressed by atoms, the *user identifier* (*user_id*), and the various *characteristics* which can be assigned to this user in this AEHS. The characteristics of a user are expressed by predicates:

has_property(user_id, characteristic_x) or
has_property(user_id, characteristic_x, value), etc.

A prominent characteristic in AEHS is the knowledge a user has on documents (or knowledge topics). The first of the following examples uses a binary value for the knowledge, the second example allows different grades of knowledge:

has_property(doc_id, user_id, know) or
has_property(doc_id, user_id, know, value), etc.

The characteristic "knowledge" is very prominent for educational adaptive hypermedia systems, so we can abbreviate the above predicates by:

knows(doc_id, user_id) or
knows(doc_id, user_id, value), etc.

2.3 OBS: The Observations

Observations are the result of monitoring a user's interactions with the AEHS at runtime. Therefore, the objects for modeling observations are the users (as in the case of the UM) and the observations.

Typical observations in AEHS are whether a user has studied some document. The corresponding predicate is

obs(doc_id, user_id, visited) or
obs(doc_id, user_id, visited, value), etc.

If the document is a test and the user has worked on this test by answering the corresponding questions, predicates like

obs(doc_id, user_id, worked_on) or
obs(doc_id, user_id, worked_on, value), etc.,

are used.

2.4 AC: The Adaptation Component

Finally, the adaptation component contains rules for describing the *adaptive functionality* of the system. An example for adaptive functionality is to decide whether a user has sufficient knowledge to study a document (recommended for learning). This functionality belongs to the group of functionalities which determine the "learning state" of a document. A simple rule might be to recommend a document for learning if all documents that are "prerequisites", e.g. that need to be studied before this document can be learned, have been visited:

$$\begin{aligned} & \forall \text{user_id} \forall \text{doc_id}_1 \\ & (\forall \text{doc_id}_2 \text{ preq}(\text{doc_id}_1, \text{doc_id}_2) \implies \text{obs}(\text{doc_id}_2, \text{user_id}, \text{visited})) \\ & \implies \text{learning_state}(\text{doc_id}_1, \text{user_id}, \text{recommended_for_reading}). \end{aligned}$$

The *adaptive treatment* is a set of rules describing the runtime behavior of the system. An often used adaptive treatment is the traffic light metaphor [1] to annotate links: Icons with different colors are used to show whether a document corresponding to a link is recommended for reading (green color), might be too difficult to study (yellow color), or is not recommended for reading (red color). The rule defining this adaptive treatment "document annotation" is:

$$\begin{aligned} & \forall \text{doc_id} \forall \text{user_id} \\ & \text{learning_state}(\text{doc_id}, \text{user_id}, \text{recommended_for_learning}) \\ & \implies \text{document_annotation}(\text{doc_id}, \text{user_id}, \text{green_icon}). \end{aligned}$$

2.5 Definition of Adaptive Educational Hypermedia Systems

In this section, we will give a logic-based definition for AEHS. We have chosen first order logic (FOL) as it allows us to provide an abstract, generalized formalization. The notation chosen in this paper refers to [14]. The aim of this logic-based definition is to accentuate the main characteristics and aspects of adaptive educational hypermedia.

Definition 2 (Adaptive Educational Hypermedia System (AEHS)) *An Adaptive Educational Hypermedia System (AEHS) is a Quadruple*

$$(DOCS, UM, OBS, AC)$$

with

DOCS: Document Space: *A finite set of first order logic (FOL) sentences with atoms for describing documents (and knowledge topics), and predicates for defining relations between these atoms.*

UM: User Model: *A finite set of FOL sentences with atoms for describing individual users (user groups), and user characteristics, as well as predicates and rules for expressing whether a characteristic applies to a user.*

OBS: Observations: *A finite set of FOL sentences with atoms for describing observations and predicates for relating users, documents / topics, and observations.*

AC: Adaptation Component: *A finite set of FOL sentences with rules for describing adaptive functionality.*

The components "document space" and "observations" describe basic data (DOCS) and run-time data (OBS). User model and adaptation component process this data, e.g. for estimating a user's preferences (UM), or for deciding about beneficial adaptive treatments for a user (AC).

3 Examples

In this section we will provide some examples of a prototypical AEHS to illustrate the applicability of our above framework. The first three examples describe prototypical (artificial AEHS) whose purpose is to illustrate the applicability of the above proposed framework. The following four examples show the logical descriptions of existing AEHS: the NetCoach system [18], the Interbook system [3], the ELM-ART II system [19], and the KBS hyperbook system [10].

3.1 A very simple AEHS

We describe a simple AEHS, called *Simple* with the following functionality: Simple can annotate hypertext-links to documents by using the traffic light metaphor with two colors: red for non recommended, green for recommended pages.

3.1.1 Simple: Document Space

A set of n atoms (n corresponds to the number of documents in the document space) which name the documents:

$$D_1, D_2, \dots, D_n.$$

Plus a finite set of predicates stating the documents that need to be studied before a document can be learned, e.g. D_j is a prerequisite for D_i :

$$\text{preq}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

(N.B.: This AEHS does not employ an additional knowledge model).

3.1.2 Simple: User Model

A set of m axioms, one for each individual user:

$$U_1, U_2, \dots, U_m.$$

3.1.3 Simple: Observations

One atom for the observation whether a document has been visited:

Visited.

And a set of predicates

$$\text{obs}(D_i, U_j, \text{Visited}) \text{ for certain } D_i, U_j.$$

3.1.4 Simple: Adaptation Component

One atom for describing the values of the adaptive functionality "learning_state":

Recommended_for_reading,

and two atoms representing values of the adaptive treatment:

Green_Icon, Red_Icon.

Rules for describing the learning state of a document

$$\begin{aligned} & \forall U_i \forall D_j \\ & (\forall D_k \text{preq}(D_j, D_k) \implies \text{obs}(D_k, U_i, \text{Visited})) \\ & \implies \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}). \end{aligned}$$

And rules for describing the adaptive link annotation with traffic lights:

$$\begin{aligned}
& \forall U_i \forall D_j \\
& \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}) \\
& \implies \text{document_annotation}(D_j, U_i, \text{Green_Icon}), \\
& \forall U_i \forall D_j \\
& \neg \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}) \\
& \implies \text{document_annotation}(D_j, U_i, \text{Red_Icon}).
\end{aligned}$$

3.2 A very simple AEHS - Extension 1

We extend our AEHS Simple by an additional rule in the user model UM. The visible adaptive functionality of this system, which we call *Simple 1*, will remain the same as in Simple, however Simple 1 deduces more information from the user observations as Simple.

Simple 1 = (DOCS_{Simple 1}, UM_{Simple 1}, OBS_{Simple 1}, AC_{Simple 1}) with

3.2.1 Simple 1: Document Space

Same as DOCS_{Simple}.

3.2.2 Simple 1: User Model

As UM_{Simple}, plus a rule for inferring that whenever a document has been learned by a user, all the documents that are prerequisites for this document are learned, too. This inference rule processes the observation, it is abbreviated by p_obs for process observation. *Simple 1* uses an additional atom for describing user characteristics:

Learned.

The inference rule for processing the observation:

$$\begin{aligned}
& \forall U_i \forall D_j \\
& (\exists D_k \text{preq}(D_k, D_j) \wedge \text{obs}(D_k, U_i, \text{Visited})) \\
& \implies \text{p_obs}(D_j, U_i, \text{Learned}).
\end{aligned}$$

3.2.3 Simple 1: Observations

Same as OBS_{Simple}.

3.2.4 Simple 1: Adaptation Component

The rule describing the learning state of a document is updated as follows:

$$\begin{aligned}
& \forall U_i \forall D_j \\
& \forall D_k (\text{preq}(D_j, D_k) \implies (\text{obs}(D_k, U_i, \text{Visited}) \vee \text{p_obs}(D_k, U_i, \text{Learned}))) \\
& \implies \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}).
\end{aligned}$$

The rules for adaptive link annotation remain unchanged with respect to *Simple*.

3.3 A very simple AEHS - Extension 2

This example illustrates the use of a knowledge graph. Simple and Simple 1 only used a domain graph. The system, called *Simple 2* is able to give a more differentiated traffic light annotations to hypertext links as Simple or Simple 1. It is able to recommend pages absolutely (green icon), shows which links lead to documents that will become understandable (dark orange icon), which might be understandable (yellow icon), or which are not recommended yet (red icon).

Simple 2 = (DOCS_{Simple 2}, UM_{Simple 2}, OBS_{Simple 2}, AC_{Simple 2}) with

3.3.1 Simple 2: Document Space

The document space contains all axioms of DOCS_{Simple} but does not contain any of the predicates. In addition, it contains a set of s atoms (s corresponds to the number of topics in the knowledge space) which name the knowledge topics:

$$T_1, T_2, \dots, T_s.$$

Plus a finite set of predicates stating the learning dependencies between these topics: Topic T_k is required to understand T_j :

$$\text{depends}(T_j, T_k) \text{ for certain } T_j \neq T_k.$$

Furthermore, the documents are characterized by a set of n predicates which assign a non-empty set of topics to each document. This can be compared by assigning a set of keywords to each document (keep in mind that more than one keyword might be assigned to a document):

$$\forall D_i \exists T_j \\ \text{keyword}(D_i, T_j).$$

3.3.2 Simple 2: User Model

The user model is the same as UM_{Simple}, plus an additional rules which defines that a topic T_i is assumed to be learned whenever the corresponding document has been visited by the user. Therefore, *Simple 2* uses like *Simple 1* the atom

Learned.

The rule for processing the observation that a topic has been learned by a user:

$$\forall U_i \forall T_j \\ (\exists D_k \text{keyword}(D_k, T_j) \wedge \text{obs}(D_k, U_i, \text{Visited}) \\ \implies \text{p_obs}(T_j, U_i, \text{Learned}).$$

3.3.3 Simple 2: Observations

Are the same as OBS_{Simple}:

3.3.4 Simple 2: Adaptation Component

The adaptation component of Simple 2 contains two further atoms (in comparison to Simple) representing new values for the learning state of a document,

Might_be_understandable, Will_become_understandable.

and two further atoms representing new values for adaptive link annotation:

Orange_Icon, Yellow_Icon.

The following rules describe the educational state of a document. Rule_1 states that a document is recommended for learning if *all* prerequisites for the keywords of this document are learned

$$\begin{aligned} & \forall U_i \forall D_j \\ & \forall T_k \left(\text{keyword}(D_j, T_k) \implies \left(\forall T_\ell \text{depends}(T_k, T_\ell) \implies \text{p_obs}(T_\ell, U_i, \text{Learned}) \right) \right) \\ & \implies \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}). \end{aligned}$$

Rule_2 states that a document might be understandable if at least some of the prerequisites have already been learned by this user:

$$\begin{aligned} & \forall U_i \forall D_j \\ & \left(\forall T_k \text{keyword}(D_j, T_k) \implies \right. \\ & \left. \left(\exists T_\ell \text{depends}(T_k, T_\ell) \implies \text{p_obs}(T_\ell, U_i, \text{Learned}) \right) \right) \\ & \wedge \neg \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}) \\ & \implies \text{learning_state}(D_j, U_i, \text{Might_be_understandable}). \end{aligned}$$

Rule_3 derives that a document will become understandable if the user has some prerequisite knowledge for at least one of the document's keywords:

$$\begin{aligned} & \forall U_i \forall D_j \\ & \exists T_k \text{keyword}(D_j, T_k) \implies \\ & \left(\exists T_\ell \text{depends}(T_k, T_\ell) \implies \text{p_obs}(T_\ell, U_i, \text{Learned}) \right) \\ & \wedge \neg \text{learning_state}(D_j, U_i, \text{Might_be_understandable}) \\ & \implies \text{learning_state}(D_j, U_i, \text{Will_become_understandable}). \end{aligned}$$

Four rules describe the adaptive link annotation:

$$\begin{aligned} & \forall U_i \forall D_j \\ & \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Green_Icon}) \\ \\ & \forall U_i \forall D_j \\ & \text{learning_state}(D_j, U_i, \text{Will_become_understandable}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Orange_Icon}) \\ \\ & \forall U_i \forall D_j \\ & \text{learning_state}(D_j, U_i, \text{Might_be_understandable}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Yellow_Icon}) \\ \\ & \forall U_i \forall D_j \\ & \neg \text{learning_state}(D_j, U_i, \text{Recommended_for_reading}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Red_Icon}) \end{aligned}$$

3.4 Summary of first three (artificial) examples

We can now easily summarize and compare the above three example systems *Simple*, *Simple 1* and *Simple 2*. Table 1 shows which objects are used in the three example systems (e.g. documents, users, topics), and describes the taxonomy of user characteristics (e.g. learned), the taxonomy of observations (e.g. visited), the taxonomy of adaptive functionality (e.g. recommended_for_reading, might_become_understandable, etc.) and the taxonomy of adaptive treatment (e.g. green_icon, red_icon, etc.).

Table 2 shows the different relations between objects.

Table 3 gives an overview about rules used in *Simple*, *Simple 1* and *Simple 2*.

System	DOCS	UM	OBS
<i>Simple</i>	$D_1, D_2, \dots, D_n.$	$U_1, U_2, \dots, U_m.$	Visited.
<i>Simple 1</i>	$D_1, D_2, \dots, D_n.$	$U_1, U_2, \dots, U_m, \text{Learned.}$	Visited.
<i>Simple 2</i>	$D_1, D_2, \dots, D_n, T_1, T_2, \dots, T_s.$	$U_1, U_2, \dots, U_m, \text{Learned.}$	Visited.

System	AC-Learning State	AC-Adaptive Link Annotation
<i>Simple</i>	Recommended_for_reading.	Green_Icon. Red_Icon.
<i>Simple 1</i>	Recommended_for_reading.	Green_Icon. Red_Icon.
<i>Simple 2</i>	Recommended_for_reading. Might_be_understandable. Will_become_understandable.	Green_Icon. Red_Icon. Orange_Icon. Yellow_Icon.

Table 1: Atoms used in *Simple*, *Simple 1* and *Simple 2*.

System	DOCS	UM	OBS	AC
<i>Simple</i>	$\text{preq}(D_i, D_j).$	–	$\text{obs}(D_k, U_j, \text{Visited}).$	–
<i>Simple 1</i>	$\text{preq}(D_i, D_j).$	–	$\text{obs}(D_k, U_j, \text{Visited}).$	–
<i>Simple 2</i>	$\text{keyword}(D_i, T_j)$ $\text{depends}(T_j, T_k).$	–	$\text{obs}(D_k, U_j, \text{Visited}).$	–

Table 2: Predicates used in *Simple*, *Simple 1* and *Simple 2*.

System	DOCS	UM	OBS
<i>Simple</i>	–	–	–
<i>Simple 1</i>	–	$\text{p_obs}(D_i, U_j, \text{Learned})$	–
<i>Simple 2</i>	–	$\text{p_obs}(D_i, U_j, \text{Learned})$	–

System	AC-Learning State	AC-Adaptive Link Annotation
<i>Simple</i>	$\text{learning_state}(D_i, U_j, X),$ X is an atom from AC-Adaptive Functionality of this system.	$\text{document_annotation}(D_i, D_j, Y),$ Y is an atom of AC-Adaptive Treatment of this system.
<i>Simple 1</i>	”	”
<i>Simple 2</i>	”	”

Table 3: Rules used in *Simple*, *Simple 1* and *Simple 2*.

3.5 NetCoach

NetCoach [18] is the successor of ELM-ART II and provides a framework for building adaptive hypermedia systems. NetCoach uses a knowledge base which consists of concepts. "These concepts are internal representations of pages that will be presented to the learner" [18]. This knowledge base is the "basis for adaptive navigations support" [18] in NetCoach, and authors "can create content-specific relations" between the concepts in the knowledge base [18]. We can formalize NetCoach in the following way:

$$\text{NetCoach} = (\text{DOCS}_{\text{NetCoach}}, \text{UM}_{\text{NetCoach}}, \text{OBS}_{\text{NetCoach}}, \text{AC}_{\text{NetCoach}}, \text{UM}_{\text{NetCoach}})$$

3.5.1 NetCoach: DocumentSpace

The document space consists of documents, test-groups and test-items.

$$D_1, \dots, D_n, \text{TG}_1, \dots, \text{TG}_k, \text{TI}_1, \dots, \text{TI}_l.$$

NetCoach uses a concept space \mathcal{C} for internally representating the documents presented to the learner. The concept space in NetCoach is isomorphic to the document space \mathcal{D} as there is a one-to-one mapping between \mathcal{C} and \mathcal{D} . To describe NetCoach, we will only refer to the objects in the document space \mathcal{D} to emphasize that relations between the concepts / documents are first class relations of the hyperspace, e.g. they are directly used for adapting the hyperspace to the user.

Documents in NetCoach are structured hierarchically in a section – subsection – subsection manner. This hierarchical structure delivers information for adaptation, too, by giving for each concept – or document – a predecessor and successor in the document space. There are four kinds of relations between documents: "prerequisite-relation", "infers-relations", "successor-relations" and "part-of-relation". In addition, there is a flag "terminal-page" attached to each document indicating whether this document is a terminal page, a "criterion" which defines the number of tests necessary to learn a document, and a "test_assignment" which relates some test_items or test_groups to a document.

The prerequisite relation assigns a set of documents to a document D_i which contains documents that need to be learned before a student can learn D_i , i.e. the prerequisite relation defines the set of prerequisite documents for a document.

$$\text{preq}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

An infers-relations assigns a set of documents to a document D_i that can be inferred to be learned whenever D_i has been learned.

$$\text{infer}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

The successor-relation and the part-of-relation are given by the hierarchical document structure underlying NetCoach (see Figure 1).

The part-of-relation assigns to each document D_i the set of documents which are sub-documents of D_i . For simplicity reasons, we assume that all documents that are sub-documents of a document, are explicitly stated in these part_of predicates. However, one can define the transitive closure of part_of as well to solve this transitivity task. D_j is part of D_i :

$$\text{part_of}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

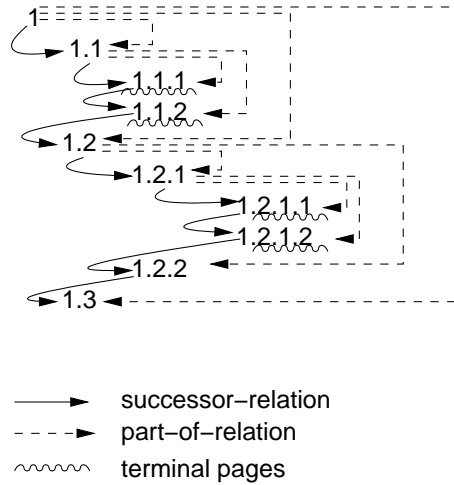


Figure 1: The hierarchy of documents/concepts in NetCoach

The successor-relation assigns for each document the next document in sequence. This is done by following the hierarchical structure step by step. A predecessor-relation is not necessary, as we can derive $\text{predecessor}(D_1, D_2)$ from $\text{successor}(D_2, D_1)$. D_j is the successor of D_i :

$\text{succ}(D_i, D_j)$ for certain D_i and one $D_j \rightarrow D_i$.

The terminal-page flag is set whenever a document has no subdocuments at all. Thus it can be directly derived from the part-of-relation.

$\text{terminal_flag}(D_i)$ for certain D_i .

NetCoach uses "test-groups" which are sets of test-items. Test-groups need not be disjoint. Test-items and test-groups are used to "assess the user's current learning state of a concept". NetCoach explicitly distinguishes between documents and test-items [17].

A test-assignment, which assigns certain test-items (TI) or test-groups (TG) to a document, is given by:

$\text{test_assignment}(D_i, TG_j)$ for certain D_i and TG_j .
 $\text{test_assignment}(D_i, TI_j)$ for certain D_i and TI_j .

In addition to the test-assignment, NetCoach assigns a criterion to each document D_i that determines how much training with the test-items and test-groups is sufficient to know D_i . This criterion is a numerical value indicating how many distinct test-items need to be successfully mastered for knowing D_i .

$\text{criterion}(D_i, \text{value})$ for certain D_i .

3.5.2 NetCoach: Observations

Observation in NetCoach are used to develop a multi-layered overlay model [18] with actually four different layers. The different layers are compiled by making observations about a user (layer 1, layer 2 and layer 4) and by processing this observations (layer 3). In the proposed formalism, everything that is a *direct* observation about the user's interactions with the system is modeled in OBS, the observations, and

all interpreted or processed observations are collected in UM, the user model description. In the following, we will therefor separate observations and processed observations into the components OBS and UM.

The **first layer** in NetCoach describes whether a user U has already visited the document page P corresponding to concept C (again, `observation` is abbreviated by `obs`):

$$\text{obs}(D_j, U_i, \text{Visited}) \text{ or certain } D_j, U_i.$$

The **second layer** contains information on which exercise or test items related to a document D_i the user has worked, and whether s/he has successfully worked on the test-items up to a certain criterion.

Thus NetCoach uses two kinds of observation (`obs`) for this layer: `worked_testitem` and `solved_testitem`:

$$\begin{aligned} &\text{obs}(TI_k, U_i, \text{Worked_testitem}) \text{ for certain } TI_k, U_i, \text{ and} \\ &\text{obs}(TI_k, U_i, \text{Solved_testitem}) \text{ for certain } TI_k, U_i. \end{aligned}$$

The **third layer** describes whether a concept could be inferred as known. This is not directly a observation but an processed observation. Due to our formalism, we collect all processed observations in UM.

The **fourth layer** finally describes whether a user has marked a concept as known. The multi-layered overlay model in NetCoach allows to reset every user model value, e.g. the user can mark or unmark concepts to be known as they like, if they pass testitems for a concept the expectation that this concept is learned rises, etc.

$$\text{obs}(D_j, U_i, \text{Marked}) \text{ for certain } D_j, U_i.$$

3.5.3 NetCoach: User Model

The User Model of NetCoach processes the observations about the user's interactions with the system.

The observation that a document D_j has been proven to be known by a user U_i by solving sufficient test_items is calculated in the following way: First, a list of all solved testitems belonging to D_j is calculated

$$\begin{aligned} &\text{solved_testitems}(U_i, D_j) = []. \\ &\forall D_j \forall U_i \\ &\forall TI_k \text{test_assignment}(D_j, TI_k) \wedge \text{obs}(TI_k, U_i, \text{solved_testitem}) \\ &\implies \text{solved_testitems}(U_i, D_j) = [\text{solved_testitems}(U_i, D_j), TI_k]. \end{aligned}$$

Then these observation are processed in the following way (`p_obs` is an abbreviation for process observation):

$$\begin{aligned} &\forall D_j \forall U_i \\ &\text{criterion}(D_j, \text{Value}) \wedge \text{length}(\text{solved_testitems}(U_i, D_j)) \geq \text{Value} \\ &\implies \text{p_obs}(D_j, U_i, \text{Tested}). \end{aligned}$$

The user model infers observations about visited documents to the according pre-requisites documents, too. This is described in NetCoach as the **third layer** of the User Modeling component. This inference is done on base of the `infer`-relation connecting to documents the user has already worked on successfully.

$$\begin{aligned} &\forall D_k \forall U_i \\ &\exists D_j (\text{infer}(D_j, D_k) \wedge \text{p_obs}(D_j, U_i, \text{Tested})) \\ &\implies \text{p_obs}(D_k, U_i, \text{Inferred_Known}) \end{aligned}$$

Furthermore, the User Model of NetCoach describes whether a document D_j has been learned by a user U_i . A document has been learned, if it is either tested, inferred from other learned documents, or marked by the user. If there are no test items assigned to the document D_j or the tests are treated as voluntary exercises (i.e. $\text{criterion}(D_j, \text{Value})$ for $\text{Value}=0$), then D_j is assumed to be learned if it has been visited, or it can be inferred from other learned concepts, or marked by the user.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \text{p_obs}(D_j, U_i, \text{Tested}) \\ & \vee (\text{criterion}(D_j, 0) \wedge (\text{obs}(D_j, U_i, \text{Visited}) \vee \text{p_obs}(D_j, U_i, \text{Inferred_Known}) \\ & \quad \vee \text{obs}(D_j, U_i, \text{Marked})) \\ & \implies \text{p_obs}(D_j, U_i, \text{Learned}). \end{aligned}$$

3.5.4 NetCoach: Adaptation Component

Adaptive link annotation A link to a document D_j is marked with a **green ball** (a sign that this document is recommended for reading) for a user U_i , if all prerequisites of this page have been learned by this user:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall D_k (\text{preq}(D_j, D_k) \implies \text{p_obs}(D_k, U_i, \text{Learned})) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Green_Ball}) \end{aligned}$$

A link to a document D_j is marked with a **red ball** (a sign that this document is *not* recommended for reading) for a user U_i , if at least one prerequisite of this page has not been learned by this user yet:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \exists D_k (\text{preq}(D_j, D_k) \wedge \neg \text{p_obs}(D_k, U_i, \text{Learned})) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Red_Ball}) \end{aligned}$$

Which is equivalent to

$$\begin{aligned} & \neg \text{document_annotation}(D_j, U_i, \text{Green_Ball}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Red_Ball}) \end{aligned}$$

A link to a document D_j is marked with a **yellow ball** (a sign that this document has been learned already) for a user U_i , if the tests corresponding to this page have been successfully passed or, if there are no tests corresponding to this page, if the page has been visited:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \text{terminal_flag}(D_j) \\ & \wedge (\text{p_obs}(D_j, U_i, \text{Tested}) \vee (\text{criterion}(D_j, 0) \wedge \text{obs}(D_j, U_i, \text{Visited})) \\ &) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Yellow_Ball}) \end{aligned}$$

In case of lessons, sections, or subsection, the yellow ball means that all subordinated pages have been learned.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \neg \text{terminal_flag}(D_j) \\ & \wedge (\forall D_k \text{part_of}(D_j, D_k) \implies \text{p_obs}(D_k, U_i, \text{Learned})) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Yellow_Ball}) \end{aligned}$$

A link to a document D_j is marked with an **orange ball** if D_j is a terminal page and inferred to be known. Otherwise (if D_j is a lesson, section, subsection, etc.) an orange ball indicates that this page has already been visited but not all subordinated pages have been learned or visited so far.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \text{terminal_flag}(D_j) \wedge \text{obs}(D_j, U_i, \text{Inferred_Known}) \wedge \neg \text{p_obs}(D_j, U_i, \\ & \text{Learned}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Orange_Ball}) \end{aligned}$$

$$\begin{aligned} & \forall D_j \forall U_i \\ & \neg \text{terminal_flag}(D_j) \wedge (\exists D_k \text{part_of}(D_j, D_k) \wedge \neg \text{p_obs}(D_j, U_i, \text{Learned}) \\ &) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Orange_Ball}) \end{aligned}$$

Adaptive Link Generation: Learning Goals NetCoach defines a learning goal as a set of documents need to be learned to fulfill the goal. The NetCoach systems recursively computes all prerequisite documents of the learning goal via the *prerequisite-relation* between documents. The resulting set of concepts (original goal concepts plus their prerequisite concepts) is ordered according to the sequential ordering of the documents (given by the *successor-relation*). Learning goals are defined by an author ("Name" is an identifier of the learning_goal):

$$\text{learning_goal}(\text{Name}) = [D_1, \dots, D_g].$$

The complete set of all learning goal-documents is recursively defined by

$$\begin{aligned} & \text{learning_goal_complete}(\text{Name}) = []. \\ & \forall D_k \\ & \neg \text{member}(D_k, \text{learning_goal_complete}(\text{Name})) \\ & \wedge (\text{learning_goal}(D_1, \dots, D_k, \dots, D_g) \\ & \quad \vee (\exists D_\ell \text{part_of}(D_\ell, D_k) \wedge \text{learning_goal}(D_1, \dots, D_\ell, \dots, D_g))) \\ & \implies \text{learning_goal_complete}(\text{Name}) = [\text{learning_goal_complete}(\text{Name}), D_k]. \end{aligned}$$

Finally, the complete set of documents belonging to a document is reordered according to the successor-relation.

$$\begin{aligned} & \text{sequence_learning_goal}(\text{Name}) = []. \\ & \forall D_k \\ & \neg \text{member}(D_k, \text{sequence}(\text{Name})) \\ & \wedge \text{learning_goal_complete}(D_1, \dots, D_k, \dots, D_n) \\ & \wedge \neg (\exists D_\ell \text{learning_goal_complete}(D_1, \dots, D_\ell, \dots, D_n) \wedge \text{succ}(D_\ell, D_k)) \\ & \implies \text{sequence_learning_goal}(\text{Name}) = [\text{sequence_learning_goal}(\text{Name}), D_k]. \end{aligned}$$

Adaptive Link Generation: Curriculum sequencing If a learning goal has been selected by a user U_i , the next page in the sequence of concepts computed for this learning goal, which is recommended for reading, is presented to U_i as the next best page.

$$\begin{aligned} & \forall D_j \\ & \text{learning_goal_complete}(D_1, \dots, D_j, \dots, D_n) \\ & \wedge \text{document_annotation}(D_j, U_i, \text{Green_Ball}) \\ & \wedge (\neg \exists D_k \text{learning_goal_complete}(D_1, \dots, D_k, \dots, D_n) \wedge \text{succ}(D_k, D_j)) \\ & \implies \text{next_best_page}(D_j, U_i) \end{aligned}$$

If the user U_i has not selected any learning goal, then the next page in the sequence of all concepts / pages in the document space which is recommended for reading according to the *green ball* annotation is presented to U_i as the next best page.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \text{document_annotation}(D_j, U_i, \text{Green_Ball}) \wedge \neg (\exists D_k \text{succ}(D_k, D_j)) \\ & \implies \text{next_best_page}(D_j, U_i) \end{aligned}$$

3.6 ELM-ART II

ELM-ART II [19] is a web-based adaptive course for learning the LISP programming language. The ELM-Art II System provides individualized feedback to students, based on analyzing the learning path of a student. It refers the student to adaptive selected examples and supports the problem solving process of students in various manner. The adaptation component of ELM-Art uses the information about prerequisite and outcome knowledge which is added to the hypermedia documents. We can formalize ELM-Art II by

$$\text{ELM-ART II} = (\text{DOC}_{\text{ELM-ART II}}, \text{UM}_{\text{ELM-ART II}}, \text{OBS}_{\text{ELM-ART II}}, \text{AC}_{\text{ELM-ART II}})$$

3.6.1 ELM-ART II: Document Space

ELM-ART II [19] uses a conceptual network for describing the knowledge of the application domain. This conceptual network is a set of knowledge concepts or concepts. Each concept corresponds to exactly one page or document in the document space thus the document space and the knowledge space are isomorphic. As in the case of NetCoach, we will only refer to documents where either a concept or document can be used. The description of the document space is given by ELM-ART II in the so called *static slot of a document* [19].

The document space consists of documents and test-items.

$$D_1, \dots, D_n, \text{TI}_1, \dots, \text{TI}_l.$$

The document space is organized hierarchically into lessons, sections, subsections and terminal pages. Terminal pages can introduce new concepts or offer problems to be solved. This hierarchical structure can be resolved into a sequence of pages (or concepts) thus each concept has a predecessor and a successor in the document space.

There are five different relations between documents: "prerequisite-relation", "outcome-relations", "related-relations", "successor-relations" and "part-of-relation". A prerequisite relation assigns a set of documents to a document D_i that are necessary for learning D_i , i.e. the prerequisite relation defines the set of prerequisite documents for a document.

$$\text{preq}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

An "outcome-relation" assigns a set of documents to a document D_i that the system assumes to be known when the user worked on that unit successfully.

$$\text{out}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

The "related-relation" can be freely used by authors to link concepts.

$$\text{related}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

The successor-relation and the part-of-relation are given by the hierarchical structure of the document space. The part-of-relation assigns to each document D_i the set of documents which are sub-documents of D_i . For simplicity reasons, we assume that all documents that are sub-documents of a document, are explicitly stated in these `part_of` predicates. However, one can define the transitive closure of `part_of` as well to solve this transitivity task. D_j is part of D_i :

`part_of(Di, Dj)` for certain $D_i \neq D_j$.

The successor-relation assigns for each document the next document in sequence. This is done by following the hierarchical structure step by step. A predecessor-relation is not necessary, as we can derive `predecessor(Di, Dj)` from `successor(Dj, Di)`. D_j is the successor of D_i :

`succ(Di, Dj)` for certain D_i and one $D_j \neq D_i$.

In addition to these above mentioned five relations between documents, there are two further document annotations: The "terminal flag" that indicates that a document has no parts, e.g. is a leaf in the hierarchy, and the "test_assignments" that relate some set of test_items to a document.

The terminal-page flag is set whenever a document has no subdocuments at all. Thus it can be directly derived from the part-of-relation.

`terminal_flag(Di)` for certain D_i .

In ELM-ART II, terminal pages have so called test slots "that may contain the description of a group of test items the learner has to perform" [19].

A test-assignment assigns certain test-items to a document D_i :

`test_assignment(Di, TIj)` for certain D_i and TI_j .

3.6.2 ELM-ART II: Observations

ELM-ART II uses a multi-layered overlay model with two layers for modeling a user [19].

The **first layer** describes whether a user U_i has already visited a document D_j :

`obs(Dj, Ui, Visited)`.

The **second layer** contains information whether a user U_i has successfully passed a test for a concept TI_k .

`obs(TIk, Ui, Solved_testitem)` for certain TI_k, U_i .

3.6.3 ELM-ART II: User Model

The user model of ELM-ART II processes the observations about the user's interactions with the system. It infers observations in various ways.

The conclusion that a user U_i has proven to know a document D_j is made whenever all test-items belonging to D_j have been solved.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall TI_k (\text{test_assignment}(D_j, TI_k) \implies \text{obs}(TI_k, U_i, \text{Solved_testitem})) \\ & \implies \text{p_obs}(D_j, U_i, \text{Tested}). \end{aligned}$$

ELM-ART II calculates that a document D_i is known to user U_i if it is either tested to be known, or there is a document D_j , that is known to a user, and D_i is one of the prerequisites of D_j . ELM-ART II explicitly distinguishes between concepts that are proven to be known by tests (*Tested*), and concepts that are only inferred to be known (*Inferred_Known*).

$$\begin{aligned} & \forall D_j \forall U_i \\ & \neg \text{p_obs}(D_j, U_i, \text{Tested}) \wedge (\exists D_k \text{preq}(D_k, D_j) \wedge \text{p_obs}(D_k, U_i, \text{Tested})) \\ & \implies \text{p_obs}(D_j, U_i, \text{Inferred_Known}) \end{aligned}$$

3.6.4 ELM-ART II: Adaptation Component

Adaptive link annotation A link to a document D_j is marked with a **green ball** (a sign that this document is recommended for reading) for a user U_i , if all prerequisites of this page have been *learned* by this user, e.g. either they are *Tested* or *Inferred_Known*:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall D_k (\text{preq}(D_i, D_k) \implies \\ & \quad (\text{p_obs}(D_k, U_i, \text{Tested}) \vee \text{p_obs}(D_k, U_i, \text{Inferred_Known}))) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Green_Ball}) \end{aligned}$$

A link to a document D_j is marked with a **red ball** (a sign that this document is *not* recommended for reading) for a user U_i , if some prerequisite of this page has not been *learned* by this user yet:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \exists D_k (\text{preq}(D_i, D_k) \wedge \\ & \quad \neg (\text{p_obs}(D_k, U_i, \text{Tested}) \vee \text{p_obs}(D_k, U_i, \text{Inferred_Known}))) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Red_Ball}) \end{aligned}$$

Which is equivalent to

$$\begin{aligned} & \neg \text{document_annotation}(D_j, U_i, \text{Green_Ball}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Red_Ball}) \end{aligned}$$

A link to a document D_j is marked with a **yellow ball** (a sign that this document has been learned already) if the tests corresponding to this document have been successfully passed or, if there are no tests corresponding to this document, if the document has been visited.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \text{terminal_flag}(D_j) \\ & \wedge (\text{p_obs}(D_j, U_i, \text{Tested}) \\ & \quad \vee (\neg \exists \text{TI}_k \text{test_assignment}(D_j, \text{TI}_k) \wedge \text{obs}(D_j, U_i, \text{Visited}))) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Yellow_Ball}) \end{aligned}$$

In case of lessons, sections, or subsection, the yellow ball means that all subordinated pages have been "learned":

$$\begin{aligned} & \forall D_j \forall U_i \\ & \neg \text{terminal_flag}(D_j) \\ & \wedge (\forall D_k \text{part_of}(D_j, D_k) \implies \\ & \quad (\text{p_obs}(D_k, U_i, \text{Inferred_Known}) \vee \text{p_obs}(D_k, U_i, \text{Tested}))) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Yellow_Ball}) \end{aligned}$$

A link to a document D_j is marked with an **orange ball** if D_j is a terminal page and inferred to be known. Otherwise (if D_j is a lesson, section, subsection, etc.) an orange ball indicates that this page has already been visited but not all subordinated pages have been learned so far.

$$\begin{aligned} & \forall D_j \forall U_i \\ & \text{terminal_flag}(D_j) \wedge \text{obs}(D_j, U_i, \text{Inferred_Known}) \wedge \neg \text{obs}(D_j, U_i, \text{Tested}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Orange_Ball}) \end{aligned}$$

$$\begin{aligned} & \forall D_j \forall U_i \\ & \neg \text{terminal_flag}(D_j) \wedge (\exists D_k \text{part_of}(D_j, D_k) \wedge \neg \text{obs}(D_j, U_i, \text{Tested})) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Orange_Ball}) \end{aligned}$$

3.7 Interbook

Interbook [5] allows the creation of adaptive electronic textbooks based on hierarchically structured MS-Word files. Courses compiled with Interbook provide individual guidance to students by annotating the navigational structure of the hypertext due to the user's learning progress, by generating individually learning paths and by personalized embedding of exercises.

We can formalize Interbook in the following way:

$$\text{Interbook} = (\text{DOCS}_{\text{Interbook}}, \text{UM}_{\text{Interbook}}, \text{OBS}_{\text{Interbook}}, \text{SD}_{\text{Interbook}}).$$

3.7.1 Interbook: Document Space

Interbook uses domain concepts which are "elementary pieces of knowledge for the given domain" [3]. The documents in Interbook are units from indexed electronic textbooks.

Interbook uses a knowledge model / concept space which consists of so called *domain concepts*. Each concept in the concept space can be used for indexing any number of documents in the document space, and for each document, there can be more than one concept that is related to this page.

Thus the document space of Interbook consists of documents, test_items, and concepts:

$$D_1, \dots, D_n, \text{TI}_1, \dots, \text{TI}_l, C_1, \dots, C_s.$$

Each electronic textbook is assumed to be hierarchically structured into chapters, sections, and subsections. At the terminal level are atomic presentations, examples, problems, or tests. A successor-relation and a part-of-relation are given by this hierarchical document structure.

The part-of-relation assigns to each document D_i the set of documents which are sub-documents of D_i . For simplicity reasons, we assume that all documents that are sub-documents of a document, are explicitly stated in these part_of predicates. However, one can define the transitive closure of part_of as well to solve this transitivity task. D_j is part of D_i :

$$\text{part_of}(D_i, D_j) \text{ for certain } D_i \neq D_j.$$

The successor-relation assigns for each document the next document in sequence. This is done by following the hierarchical structure step by step. A predecessor-relation is not necessary, as we can derive predecessor(D_i, D_j) from successor(D_j, D_i). D_j is the successor of D_i :

$\text{succ}(D_i, D_j)$ for certain D_i and one $D_j \neq D_i$.

A further document annotation is used in Interbook: The terminal-page flag is set whenever a document has no subdocuments at all. Thus it can be directly derived from the part-of-relation.

$\text{terminal_flag}(D_i)$ for certain D_i .

There are two kinds of relations between documents (or test_items) and concepts: "prerequisite-relation", and "outcome-relations". A prerequisite-relation assigns a set of concepts to a document D_i (test_item TI_k) that are necessary for learning $D_i(TI_k)$, i.e. the prerequisite relation defines the set of prerequisite concepts for $D_i(TI_k)$.

$\text{preq}(D_i, C_j)$ for certain D_i, C_j .
 $\text{preq}(TI_k, C_j)$ for certain TI_k, C_j .

An outcome-relations assigns a set of concepts to a document D_i (test_item TI_k) that describe the concepts that should be learned on this document (test_item).

$\text{out}(D_i, C_j)$ for certain D_i, C_j .
 $\text{out}(TI_k, C_j)$ for certain TI_k, C_j .

3.7.2 Interbook: Observations

Interbook distinguishes between different levels of knowledge a user can have about a domain concept C_i . These levels are *no_knowledge* if a user has not learned a concept at all, *beginner_knowledge* if a user has read a page, *intermediate_knowledge* if a user has read about this concept on two different pages, and *expert_knowledge* if a user has performed a test related to the concept successfully.

These knowledge grades are calculated in the user model of Interbook on basis of the following observations: A user can visited a document D_i

$\text{obs}(D_j, U_i, \text{Visited})$ for certain D_j, U_i .

Furthermore, a user U_i can solve a test-item TI_k :

$\text{obs}(TI_k, U_i, \text{Solved})$ for certain TI_k, U_i .

3.7.3 Interbook: User Model

The user model assigns for each user U_i the grade of knowledge s/he has for each concept from the concept space.

A user U_i has **Beginner_knowledge** if s/he has a read a page about this concept.

$$\begin{aligned} & \forall C_j \forall U_i \\ & \exists D_k \text{obs}(D_k, U_i, \text{Visited}) \wedge \text{out}(D_k, C_j) \\ & \implies \text{p_obs}(C_j, U_i, \text{Beginner_knowledge}) \end{aligned}$$

A user U_i is assumed to have **Intermediate_knowledge** if s/he has read about a concept C_j on two different documents D_k, D_ℓ .

$$\begin{aligned} & \forall C_j \forall U_i \\ & \exists D_k \exists D_\ell \neg(D_k = D_\ell) \wedge \text{obs}(D_k, U_i, \text{Visited}) \wedge \text{obs}(D_\ell, U_i, \text{Visited}) \\ & \implies \text{p_obs}(C_j, U_i, \text{Intermediate_knowledge}) \end{aligned}$$

The level of **Expert_knowledge** can be reached when a user U_i has solved a test belonging to a concept C_j .

$$\begin{aligned} & \forall C_j \forall U_i \\ & \exists \text{TI}_k \text{out}(\text{TI}_k, C_j) \wedge \text{obs}(\text{TI}_k, U_i, \text{Solved}) \\ & \implies \text{p_obs}(C_j, U_i, \text{Expert_knowledge}) \end{aligned}$$

If a user U_i has neither **Beginner_knowledge** nor **Intermediate_knowledge** nor **Expert_knowledge** about a concept C_i , this user is assumed to have **No_knowledge** about C_i .

$$\begin{aligned} & \forall C_j \forall U_i \\ & \neg \text{p_obs}(C_j, U_i, \text{Expert_knowledge}) \\ & \wedge \neg \text{p_obs}(C_j, U_i, \text{Intermediate_knowledge}) \\ & \wedge \neg \text{p_obs}(C_j, U_i, \text{Beginner_knowledge}) \\ & \implies \text{p_obs}(C_j, U_i, \text{No_knowledge}) \end{aligned}$$

3.7.4 Interbook: Adaptation Component

Adaptive link annotation Interbook uses different checkmarks to indicate a users knowledge about documents, and coloured balls to give advise to the user which documents to learn next, etc.

A **Big_checkmark** is used to indicate that a user has expert knowledge on all outcome concepts of this page:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall C_k (\text{out}(D_j, C_k) \implies \text{p_obs}(C_k, U_i, \text{Expert_knowledge})) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Big_checkmark}). \end{aligned}$$

A **Normal_checkmark** is used to indicate that a user has intermediate knowledge on all all outcome concepts of this page:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall C_k (\text{out}(D_j, C_k) \implies \text{p_obs}(C_k, U_i, \text{Intermediate_knowledge})) \\ & \wedge \neg \text{document_annotation}(D_j, U_i, \text{Big_checkmark}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Normal_checkmark}). \end{aligned}$$

A **Small_checkmark** is used to indicate that a user has **Beginner_knowledge** on all outcome concepts of this page:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall C_k (\text{out}(D_j, C_k) \implies \text{p_obs}(C_k, U_i, \text{Beginner_knowledge})) \\ & \wedge \neg \text{document_annotation}(D_j, U_i, \text{Normal_checkmark}) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Small_checkmark}). \end{aligned}$$

A link to a document D_j is marked with a **Green_ball** for a user U_i if it is recommended for reading, e.g. if all its prerequisites are known to U_i with grade **Beginner_knowledge**:

$$\begin{aligned} & \forall D_j \forall U_i \\ & \forall C_k (\text{preq}(D_j, C_k) \implies \text{p_obs}(C_k, U_i, \text{Beginner_knowledge})) \\ & \implies \text{document_annotation}(D_j, U_i, \text{Green_ball}) \end{aligned}$$

A **White_ball** indicates that a document D_j shows nothing new for this user, that means that all outcome concepts of this page have been read.

$$\begin{aligned}
& \forall D_j \forall U_i \\
& \forall C_k (\text{out}(D_j, C_k) \implies \text{obs}(C_k, U_i, \text{Visited})) \\
& \implies \text{document_annotation}(D_j, U_i, \text{White_ball})
\end{aligned}$$

A link to a document D_j is marked with a **Red_ball** if D_j is not recommended for reading yet, i.e. not all prerequisite concepts have been learned so far:

$$\begin{aligned}
& \forall D_j \forall U_i \\
& \exists C_k (\text{preq}(D_j, C_k) \wedge \text{p_obs}(C_k, U_i, \text{No_knowledge})) \\
& \implies \text{document_annotation}(D_j, U_i, \text{Red_ball})
\end{aligned}$$

Prerequisite-based help The prerequisite-based-help for a document D_j is a list of all pages that explain the prerequisites of all concepts that are presented on D_j .

$$\begin{aligned}
& \text{prerequisite_based_help_concepts}(D_i) = []. \\
& \forall C_j \\
& \neg \text{member}(C_j, \text{prerequisite_based_help_concepts}(D_i)) \wedge \text{preq}(D_i, C_j) \\
& \implies \text{prerequisite_based_help_concepts}(D_i) = \\
& \quad [\text{prerequisite_based_help_concepts}(D_i), C_j].
\end{aligned}$$

From this we derive the documents for a prerequisite-based-help by

$$\begin{aligned}
& \text{prerequisite_based_help_documents}(D_i) = []. \\
& \forall D_j \forall C_k \\
& \neg \text{member}(D_j, \text{prerequisite_based_help_documents}(D_i)) \\
& \wedge \text{member}(C_k, \text{prerequisite_based_help_concepts}(D_i)) \\
& \wedge \text{out}(D_j, C_k) \\
& \implies \text{prerequisite_based_help_documents}(D_i) = \\
& \quad [\text{prerequisite_based_help_documents}(D_i), D_j].
\end{aligned}$$

This set of help documents can be ordered due to the knowledge of a learner by sorting pages whose outcome concepts are not known to the user (that means documents D with $\text{p_obs}(D, U_i, \text{No_knowledge})$) to the beginning of the list.

Learning Goals Interbook associates *learning goals* to documents. The concepts for the learning goal are defined by the transitive closure on the prerequisite-relation of concepts, the starting concepts are the prerequisite concepts of the document associated to this learning goal.

We collect all prerequisite concepts of a document D_i recursively by

$$\begin{aligned}
& \text{prerequisite_concepts}(D_i) = []. \\
& \forall C_j \\
& \text{preq}(D_i, C_j) \implies \text{prerequisite_concepts}(D_i) = [\text{prerequisite_concepts}(D_i), \\
& \quad C_j] \\
& \forall C_j \forall C_k \forall D_\ell \\
& \text{member}(C_j, \text{prerequisite_concepts}(D_i)) \wedge \text{out}(D_\ell, C_k) \\
& \wedge \text{preq}(D_\ell, C_k) \wedge \neg \text{member}(C_k, \text{prerequisite_concepts}(D_i)) \\
& \implies \text{prerequisite_concepts}(D_i) = [\text{prerequisite_concepts}(D_i), C_k]
\end{aligned}$$

A *reading_sequence* is calculated for each learning goal in the following way: First, a list of all documents that contain the necessary prerequisite knowledge to the goal concepts itself is generated. As a learning goal is binded to a document D_i , this set of required documents is also binded to this D_i :

$$\begin{aligned}
& \forall D_j \\
& \exists C_k (\text{out}(D_i, C_k) \vee \text{member}(C_k, \text{prerequisite_concepts}(D_i))) \wedge \text{out}(D_j, \\
& C_k) \\
& \implies \text{required_documents}(D_i, D_j).
\end{aligned}$$

Afterwards this sequence is ordered according to the overall sequence of pages in Interbook.

TeachMe Interbook has a TeachMe-Button that allows a user to ask for a sequence of documents explaining the current document detailly. The TeachMe functionality is implemented as a goal whose goal concepts are the prerequisites and outcomes of the associated document.

3.8 KBS Hyperbook

The KBS hyperbook system [10] is an adaptive hypermedia system which guides the students through the information space individually by showing next reasonable learning steps, by selecting projects, generating and proposing reading sequences, annotating the educational state of information, and by selecting useful information, based on a user's actual goal and knowledge [9]. KBS Hyperbook implements the adaptation component on top of an existing, concept-based hypermedia system.

We can formalize KBS Hyperbook in the following way:

$$\text{KBS Hyperbook} = (\text{DOCS}_{\text{KBS Hyperbook}}, \text{UM}_{\text{KBS Hyperbook}}, \text{OBS}_{\text{KBS Hyperbook}}, \text{SD}_{\text{KBS Hyperbook}})$$

3.8.1 KBS Hyperbook: Document Space

KBS hyperbook distinguishes documents in the document space according to their role in the learning system, e.g. the underlying concept-based hypermedia system: Documents can be exercises, projects, examples, lecture notes, course notes, glossary entries, or topics.

Thus, the document space consists of documents

$$D_1, \dots, D_n.$$

Each document has a role which is defined by the concept-based hyperspace:

$$\begin{aligned}
& \text{role}(D_i, \text{Lecture}) \text{ for certain } D_i, \\
& \text{role}(D_i, \text{Lecture_Note}) \text{ for certain } D_i, \\
& \text{role}(D_i, \text{Course}) \text{ for certain } D_i, \\
& \text{role}(D_i, \text{Exercise}) \text{ for certain } D_i, \\
& \text{role}(D_i, \text{Project_Description}) \text{ for certain } D_i, \\
& \text{role}(D_i, \text{Example}) \text{ for certain } D_i, \\
& \text{etc.}
\end{aligned}$$

KBS Hyperbook uses a knowledge base or concept space. The knowledge base consists of so called *knowledge items*:

$$C_1, \dots, C_s.$$

A knowledge item might represent either an introduction to a concept or the concept itself:

$$\begin{aligned}
& \text{role}(C_i, \text{Introduction}) \text{ for certain } C_i, \\
& \text{role}(C_i, \text{Concept}) \text{ for certain } C_i.
\end{aligned}$$

Each document from the document space is indexed by some concepts from the knowledge base which describe the content of the resource. Thus this indexing is like adding a set of keywords to each resource, where the keywords come from a controlled vocabulary (the knowledge space).

keyword (D_i, C_j) for certain D_i, C_j

Documents are related in KBS Hyperbook according to the conceptual model of the hyperspace. These fixed relations are not used by the adaptation component therefor we will omit them. The KBS Hyperbook asks the adaptation component for annotation of links to document or for additional relations between the documents that are generated by the adaptation component during runtime.

The knowledge items in KBS Hyperbook are related to each other using a "learning dependency" relation which is mainly a prerequisite relation. If a knowledge concept C_j is required to learn or understand C_i then there is a learning dependency relation between C_i and C_j :

depends(C_i, C_j) for certain C_i, C_j .

3.8.2 KBS Hyperbook: Observations

KBS Hyperbook stresses the importance of *active learning*. For this purpose, KBS Hyperbook employs constructivist learning strategies [7]. Following this teaching approach, observations about the student's work with the hyperbook can only be made if a student has worked on a project. Observations about a student's performance are then made by mentors, or are based on self-judgment of the students. Each observation expresses the grade of knowledge the user has on a \mathcal{KI} . KBS Hyperbook uses four grades of knowledge: *expert knowledge*, *advanced knowledge*, *beginner's knowledge*, *novice's knowledge*.

The observations about a user's interaction required for KBS Hyperbooks are:

obs($C_j, U_i, \text{Expert_knowledge}$) for certain C_j, U_i ,
 obs($C_j, U_i, \text{Advanced_knowledge}$) for certain C_j, U_i ,
 obs($C_j, U_i, \text{Beginner_knowledge}$) for certain C_j, U_i ,
 obs($C_j, U_i, \text{Novice_knowledge}$) for certain C_j, U_i .

3.8.3 KBS Hyperbook: User Model

KBS Hyperbook constructs a knowledge model based on the learning-dependency-relation between the concepts in the knowledge base. On basis of this knowledge model a Bayesian Network is constructed which calculates estimations on the knowledge of each individual user [8]. The system's estimation about the knowledge of a user U_i are stored as ordered pairs

(knowledge concept, w(knowledge concept))

with w is a random variable with four discrete values E (expert), F (advanced), A (beginner), and N (novice). The probability distribution calculated by the Bayesian Network is interpreted to five different grades of knowledge in the following way:

$$\begin{aligned} \forall C_j \forall U_i P(C_i = F) + P(C_i = A) &> P(C_i = E) + P(C_i = N) \\ &\implies \text{p_obs}(C_j, U_i, \text{Known}) \\ \forall C_j \forall U_i P(C_i = E) + P(C_i = F) &> P(C_i = A) + P(C_i = N) \\ &\implies \text{p_obs}(C_j, U_i, \text{Well_known}) \\ \forall C_j \forall U_i P(C_i = E) &> P(C_i = F) + P(C_i = A) + P(C_i = N) \\ &\implies \text{p_obs}(C_j, U_i, \text{Excellently_known}) \end{aligned}$$

$$\begin{aligned}
& \forall C_j \forall U_i P(C_i = A) + P(C_i = N) > P(C_i = E) + P(C_i = F) \\
& \implies \text{p_obs}(C_j, U_i, \text{Partly_known}) \\
& \forall C_j \forall U_i P(C_i = N) > P(C_i = E) + P(C_i = F) + P(C_i = A) \\
& \implies \text{p_obs}(C_j, U_i, \text{Not_known})
\end{aligned}$$

KBS Hyperbook calculates further functions, e.g. “Child_known” which is the threshold value denoting that a prerequisite concept C_j is sufficiently known to a user U_i to understand the new concept:

$$\begin{aligned}
& \forall C_j \forall U_i \\
& \text{p_obs}(C_j, U_i, \text{Known}) \vee \text{p_obs}(C_j, U_i, \text{Well_known}) \\
& \vee \text{p_obs}(C_j, U_i, \text{Excellently_known}) \\
& \implies \text{p_obs}(C_j, U_i, \text{Child_Known})
\end{aligned}$$

The “Parent_known” function denotes a threshold value for a “good known concept”. It is useful e.g. to infer that the prerequisites of a concept must be known when the concept itself is parent_known:

$$\begin{aligned}
& \forall C_j \forall U_i \\
& \text{p_obs}(C_j, U_i, \text{Well_known}) \vee \text{p_obs}(C_j, U_i, \text{Excellently_known}) \\
& \implies \text{p_obs}(C_j, U_i, \text{Parent_Known})
\end{aligned}$$

3.8.4 KBS Hyperbook: Adaptation Component

Adaptive link annotation A document D_j is recommended for reading (**green ball**) to a user U_i if all dependent concepts of the keyword concepts of D_j are Child_known:

$$\begin{aligned}
& \forall D_j \forall U_i \forall C_k \forall C_\ell \\
& (\text{keyword}(D_j, C_k) \implies \\
& \quad (\text{learning_dependency}(C_k, C_\ell) \implies \text{p_obs}(C_\ell, U_i, \text{Child_known}))) \\
& \implies \text{document_annotation}(D_j, U_i, \text{Green_ball}) .
\end{aligned}$$

The content of a document D_j is already known (**white ball**) to a user U_i if all keyword concepts of D_j are Parent_known:

$$\begin{aligned}
& \forall D_j \forall U_i \forall C_k \\
& (\text{keyword}(D_j, C_k) \implies \text{p_obs}(C_k, U_i, \text{Parent_known})) \\
& \implies \text{document_annotation}(D_j, U_i, \text{White_ball}) .
\end{aligned}$$

A document D_j is not recommended for reading (**red ball**) if it is neither recommended for reading or already known:

$$\begin{aligned}
& \forall D_j \forall U_i \\
& \neg (\text{document_annotation}(D_j, U_i, \text{Green_ball})) \\
& \wedge \neg (\text{document_annotation}(D_j, U_i, \text{White_ball})) \\
& \implies \text{document_annotation}(D_j, U_i, \text{Red_ball}).
\end{aligned}$$

Learning Goals For KBS Hyperbook a learning goal is a set of knowledge concepts. Either a user can define a learning goal on his own by selecting some knowledge concepts he is interested in, or he can ask the KBS Hyperbook system for the next reasonable learning goal.

$$\text{learning_goal}(U_i) = \text{first}(\text{learning_goal}(C_1, \dots, C_s)).$$

The next reasonable learning goal for a user is calculated in the following way: A Learning Sequence through the entire hypertext is generated in the way described in the next paragraph. The first concept in the raw sequence which is marked as recommended for reading is taken as the next learning goal.

$$\text{next_reasonable_learning_goal} =$$

Learning Sequence In order to construct a learning sequence we first mark all concepts in the knowledge model which should be contained in the learning sequence. E.g. if a user defines a learning goal “I want to learn concepts A, B, C and D ”, the nodes in the knowledge model corresponding to A, B, C and D are marked, e.g. the nodes a, b, c, d, e, f, g and h (N.B. a learning goal or topic may comprise one or more knowledge concepts). The children (and the children of those children etc.) of the marked nodes are marked as well. A routine then checks for each marked node c whether one of the following expressions hold: $\text{p_obs}(C_j, U_i, \text{Known})$ or $\text{p_obs}(C_j, U_i, \text{Well_known})$ or $\text{p_obs}(C_j, U_i, \text{Excellently_known})$. If this function computes true for a node the marking of this node is deleted. We then make a depth-first traversal through the knowledge model and collect the marked nodes. Thus we obtain a sequence of knowledge concepts $[C_1, \dots, C_n]$.

$$\begin{aligned} \text{candidate_for_sequence}(H, (c_1, \dots, c_n)) \leftarrow & \\ (c_{H_1}, \dots, c_{H_n}) \subset (c_1, \dots, c_n) \wedge \text{index}((c_{H_1}, \dots, c_{H_n}), H) & \\ \wedge ((c_{H_1}, \dots, c_{H_m}) \subset (c_1, \dots, c_n) \wedge \text{index}((c_{H_1}, \dots, c_{H_m}), H) & \\ \Rightarrow (c_{H_1}, \dots, c_{H_m}) \subset (c_{H_1}, \dots, c_{H_n})) & \end{aligned}$$

This set of candidates for the sequence is ordered in the following way:

- $H \in \text{final_list}(c_1, (c_1, \dots, c_n)) \leftarrow \text{index}(c_1, H)$
- $\forall (c_1, \dots, c_i) \subset (c_1, \dots, c_n)$
 $(H \in \text{final_list}((c_1, \dots, c_i), (c_1, \dots, c_n)) \leftarrow$
 $\text{index}(c_i, H) \wedge \neg (H \in \text{final_list}((c_1, \dots, c_i), (c_1, \dots, c_n))))$

On base of this sequence of knowledge concepts we select a set of documents which match the contained knowledge concepts.

Glossary The glossary contains all concepts from the knowledge space that are either introductions to concepts or leaf-concepts concerning the learning-dependency-relation between concepts.

$$\begin{aligned} \forall C_i & \\ \text{role}(C_i, \text{Introduction}) \vee \neg (\exists C_j \text{learning_dependency}(C_i, C_j)) & \\ \Rightarrow \text{in_glossary}(C_i). & \end{aligned}$$

Information Index For each learning goal or abstract: for each set of concepts, an information index, e.g. a set of documents explaining these concepts, is generated :

$$\begin{aligned} \text{information_index}([C_1, \dots, C_g]) = [] . & \\ \forall C_i \forall D_j & \\ \text{member}(C_i, [C_1, \dots, C_g]) \wedge \text{keyword}(D_j, C_i) & \\ \wedge \neg \text{member}(C_i, \text{information_index}(\text{learning_goal}(C_1, \dots, C_g))) & \\ \Rightarrow \text{information_index}([C_1, \dots, C_g]) = [\text{information_index}([C_1, \dots, C_g]), & \\ D_j] & \end{aligned}$$

3.9 Summary of four exemplary described AEHS

This chapter provides synoptical tables of the logic-based characterization of the adaptive educational hypermedia systems NetCoach [18], Interbook [3], ELM-ART II [19], and KBS hyperbook [10]. The atoms used in the four systems in the components DOCS, UM, OBS, and AC are summarized in table 4. Table 5 shows the used predicates. An overview on the rules is given in table 6.

System	DOCS	UM	OBS
NetCoach	$D_1, \dots, D_n,$ $TG_1, \dots, TG_k,$ $TI_1, \dots, TI_\ell.$	$U_1, \dots, U_m,$ Learned, Inferred_Known, Tested.	Visited, Solved_Testitem, Marked.
ELM-ART II	$D_1, \dots, D_n,$ $TI_1, \dots, TI_\ell.$	$U_1, \dots, U_m,$ Tested, Inferred_known.	Visited, Solved_Testitem.
InterBook	$D_1, \dots, D_n,$ $TI_1, \dots, TI_\ell,$ $C_1, \dots, C_s.$	$U_1, \dots, U_m,$ Learned, Beginner, Intermediate, Expert, No_knowledge.	Visited, Solved.
KBS Hyperbook	$D_1, \dots, D_n,$ $C_1, \dots, C_s.$	$U_1, \dots, U_m,$ Learned, Known, Well_known, Excellently_known, Partly_known, Not_known, Child_known, Parent_known.	Marked, Expert, Advanced, Beginner, Novice.
System	AC-Adaptive Link Annotation		AC - Others
NetCoach	Green_Ball, Red_Ball, Yellow_Ball, Orange_Ball.		-
ELM-Art II	Green_Ball, Red_Ball, Yellow_Ball, Orange_Ball.		-
Interbook	Small_Checkmark, Normal_Checkmark, Big_Checkmark, Green_Ball, White_Ball, Red_Ball.		-
KBS Hyperbook	Green_Ball, White_Ball, Red_Ball.		-

Table 4: Atoms used in NetCoach, ELM-ART II, Interbook and KBS Hyperbook.

System	DOCS		
NetCoach	<preq(d<sub>i, D_j) (prerequisite knowledge) infer(D_i, D_j) (documents inferred to be learned by studying D_i) succ(D_i, D_j) (reading order) part_of(D_i, D_j) (chapter structure) terminal_flag(D_i) (whether a document has no further sub-documents) criterion(D_i, Value) (defines number of testitems necessary for mastering D_i) test_assignment(D_i, X), X ∈ {Testgroup, Testitem}, (relates documents with testgroups and testitems) </preq(d<sub>		
ELM-ART II	<preq(d<sub>i, D_j) (prerequisite knowledge) out(D_i, D_j) (documents inferred to be learned by studying D_i) related(D_i, D_j) (author-defined relation between documents) successor(D_i, D_j) (reading order) part_of(D_i, D_j) (chapter structure) terminal_flag(D_i) (whether a document has no further sub-documents) test_assignment(D_i, X), X ∈ {Testslot, Testitem} (relates documents with testslots and testitems) </preq(d<sub>		
InterBook	<preq(d<sub>i, C_j) (prerequisite knowledge) out(D_i, C_j) (concepts inferred to be learned by studying D_i) succ(D_i, D_j) (reading order) terminal_flag(D_i) (whether a document has no further sub-documents) part_of(D_i, D_j) (chapter structure) </preq(d<sub>		
KBS Hyperbook	keyword(D _i , C _j) assigns some concepts each document depends(C _i , C _j) learning dependencies between concepts role(D _i , X), X ∈ {Course, Goal, Lecture, Example, etc.} role of the document D _i role(C _i , X), X ∈ {Introduction, Concept} role of the concept C _i		
System	UM	OBS	AC
NetCoach	–	obs(D _i , U _j , X), X ∈ {Visited, Solved_Testitem, Marked}	
ELM-ART II	–	obs(D _i , U _j , X), X ∈ {Visited, Solved_Testitem}	–
InterBook	–	obs(D _i , U _j , X), X ∈ {Visited, Solved}	–
KBS Hyperbook	–	obs(C _i , U _j , Marked, Value), Value ∈ {Expert, Advanced, Beginner, Novice}	–

Table 5: Predicates used in NetCoach, ELM-ART II, Interbook and KBS Hyperbook.

System	DOCS	UM	OBS
NetCoach	–	Rules to infer p_obs(D_i, U_j, X), $X \in \{\text{Inferred_Known, Learned, Tested}\}$	–
ELM-ART II	–	Rules to infer p_obs(D_i, U_j, X), $X \in \{\text{Inferred_Known, Tested}\}$	–
InterBook	–	Rules to infer p_obs($C_i, U_j, \text{Learned}, X$), $X \in \{\text{Expert, Intermediate, Beginner, No_knowledge}\}$.	–
KBS Hyperbook	–	Rules to infer p_obs($C_i, U_j, \text{Learned}, X$), $X \in \{\text{Known, Well_known, Excellently_known, Partly_known, Not_known, Child_known, Parent_known}\}$.	–
System	AC - Adaptive Link Annotation		
NetCoach	Rules to infer document_annotation(D_i, U_j, X), $X \in \{\text{Green_Ball, Red_Ball, Yellow_Ball, Orange_Ball}\}$.		
ELM-ART II	Rules to infer document_annotation(D_i, U_j, X), $X \in \{\text{Green_Ball, Red_Ball, Yellow_Ball, Orange_Ball}\}$.		
InterBook	Rules to infer document_annotation(D_i, U_j, X), $X \in \{\text{Green_Ball, White_Ball, Red_Ball, Small_Checkmark, Normal_Checkmark, Big_Checkmark}\}$.		
KBS Hyperbook	Rules to infer document_annotation(D_i, U_j, X), $X \in \{\text{Green_Ball, White_Ball, Red_Ball}\}$.		
System	AC-Adaptive Link Generation		
NetCoach	Rules to infer next_best_page(D_i, U_j), learning_goal(X), curriculum_sequencing(D_1, \dots, D_ℓ)		
ELM-ART II	Rules to infer next_best_page(D_i, U_j) (+ a tutoring component for the lisp domain)		
InterBook	Rules to infer prerequisite_based_help(D_i, U_j), learning_goal(D_i), reading_sequence(D_i, U_j), teach_me(D_i).		
KBS Hyperbook	Rules to infer learning_sequence($[C_1 \dots, C_n], U_j$), glossary(D_i) learning_goal($[C_1 \dots, C_n]$), next_reasonable_goal(U_j) information_index($[C_1 \dots, C_n]$)		

Table 6: Rules used in NetCoach, ELM-ART II, Interbook and KBS Hyperbook.

4 Overview: Metadata required in the studied AEH systems

This section discusses which Metadata Information is used in the adaptive hyper-media systems presented in this report.

4.1 NetCoach

4.1.1 Metadata:

documents NetCoach distinguishes between three different kinds of documents: 'normal' documents, testgroups, and testitems.

concepts -

metadata on documents **inf(D)**: information about the inference range of document \mathcal{D}

preq(D): information about the prerequisites of document \mathcal{D}

succ(D): the successor of a document \mathcal{D}

part-of(D): which documents are part of this document \mathcal{D}

criterion(D): numerical value indicating which amount of training is required for learning this document

testgroups(D): The set of testgroups assigned to document \mathcal{D}

testitems(D): The set of testitems assigned to document \mathcal{D}

+ **structure: on Documents: Section, Subsection, Subsubsection, etc**

This is something like a recurring structure for a hierarchy. The position in this hierarchy determines the role of the concept. Terminal pages are important, too.

4.1.2 User's Observations

$visited(D, \mathcal{U})$: Whether a page D has been visited by user \mathcal{U} .

$worked_on_testitem(t, \mathcal{U})$: Whether a user \mathcal{U} has worked on a testitem $t \in \mathcal{T}_{\mathcal{I}}$.

$successfully_worked_on_testitem(t, \mathcal{U})$: Whether a user \mathcal{U} has successfully worked on a testitem $t \in \mathcal{T}_{\mathcal{I}}$.

$marked(C, \mathcal{U})$: Whether a user \mathcal{U} has marked concept C has already known.

4.2 ELM-ART II

4.2.1 Metadata

documents ELM-ART II distinguishes between two different kinds of documents: 'normal' documents, and testitems.

concepts -

metadata on documents **preq(D)**: information about the prerequisites of document \mathcal{D}

out(D): information about the outcome of document \mathcal{D}

rel(D): information about the whatever related documents to document \mathcal{D}

succ(D): the successor of a document \mathcal{D}

part-of(D): which documents are part of this document \mathcal{D} required for learning this document

testitems(D): The set of testitems assigned to document \mathcal{D}

4.2.2 Observations

visited(P,U) : Whether a page P has been visited by user U .

solved(t,U) : Whether a user U has solved a test-item $t \in T$.

+ **hierarchical structure, terminal pages**

4.3 Interbook

documents Interbook distinguishes between two different kinds of documents: 'normal' documents, and testitems.

concepts Controlled Vocabulary of knowledge concepts

metadata on documents **preq(D)**: information about the prerequisites of document D

out(D): information about the outcome of document D

testitems(D): The set of testitems assigned to document D

+ **hierarchical structure, terminal pages**

4.3.1 Observations

visited(P,U) : Whether a page P has been visited by user U .

solved_successfull(T,U) : Whether a user U has successfully worked on a test $T \in T$.

4.4 KBS Hyperbook

4.4.1 Metadata

documents Documents in KBS Hyperbook have a conceptual role: they can be exercises, projects, examples, lecture notes, course notes, glossary entries, or topics.

concepts Controlled Vocabulary of knowledge concepts.

metadata on documents **index(D)**: information about the content of document D

metadata on knowledge concepts *learning_dependency(KI_i, KI_j)* \iff KI_i is required to understand KI_j

4.4.2 Observations

Each observation expresses the grade of knowledge the user has on a KI. KBS Hyperbook uses four grades *expert knowledge*, *advanced knowledge*, *beginner's knowledge*, *novice's knowledge*.

knowledgegrade(C,U) The grade of knowledge a user has on a knowledge concept C .

5 Discussion

In this report, we have proposed a component-based definition of adaptive educational hypermedia systems that uses first-order logic to characterize AEHS. With this approach

- we can easily compare the adaptive functionality of the AEHS: we can e.g. derive that the above characterized systems are very similar in their way of employing adaptive functionality (all make adaptive navigation support, no adaptive presentation support (with respect to Brusilovsky's taxonomy of adaptive hypermedia technologies [2]));
- we hide a lot of functionality behind the rules, e.g. KBS Hyperbook uses a Bayesian Network to calculate the `Inferred_known` characteristic. This is technically very different to calculating this characteristic by compiling the transitive closure of prerequisites. But, logically, it has the same functionality: to estimate the user's current knowledge state based on some input parameters (the observations);
- we can describe the taxonomy of concepts used by the systems in document spaces, the user models, the observations, and the adaptation component. E.g. in case of the document space, we can derive that Interbook uses documents, testitens and knowledge concepts, ELM-ART II uses documents and testitens, etc.;
- the rules in the adaptation component show which data is processed by the system for making decisions about particular adaptive treatment; decisions;
- thus we can encapsulate adaptive functionality in order to support transfer of functionality between AEHS,
- and to support the more wide-spread use of adaptation in web-based educational systems, e.g. by employing Web-services that provide adaptive features.

During the application of the proposed characterization of AEHS, it turned out that the documents and their relations play a decisive role for the way how adaptation components draw conclusions. We have seen, that, in contrary to our intentions motivated by the transfer of Reiter's approach [13] to educational hypermedia, we were not able to generalize the diversity of rules for adaptation for a meta-description of adaptation. However, we claim that a logical characterization of adaptive educational hypermedia is a way to find solutions of current open questions in this area. E.g. currently, there is no catalogue of "metadata for adaptation" which could be used in LOM [12], SCORM [15] or other catalogues of metadata for education. The main objection is that adaptive educational hypermedia systems are "too different" to generalize for a meta-data driven description. From the above characterizations we can derive which meta-data is needed by the characterized AEHS: We can derive which sources for input data are used in the different systems from DOCS and OBS. These sources can now be used as a candidate set for meta-data for adaptation. Further, our approach contributes to solutions for the open document space problem [11, 2]: If we consider adaptive functionality as a query in open environments (as it has been done e.g. in [6]) it turns out that a decisive task is to determine the characteristics of adaptive functionality in order to define useful queries.

6 Conclusion

This report proposes a component-based definition of adaptive educational hypermedia based on first-order logic. We have shown the applicability of such a formal

description language for adaptive educational hypermedia in various examples. We claim that this logical characterization of adaptive educational hypermedia enables comparison of adaptive functionality in a well-grounded way, promotes the transfer of adaptive functionality to other educational hypermedia and web-based systems, defines rule-based descriptions of adaptivity, and supports the understanding of the role of metadata for adaptation.

Acknowledgment

We would like to thank Peter Brusilovsky and Gerhard Weber for the discussions on their adaptive educational hypermedia systems.

References

- [1] BRUSILOVSKY, P. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction* 6, 2-3 (1996), 87–129.
- [2] BRUSILOVSKY, P. Adaptive hypermedia. *User Modeling and User-Adapted Interaction* 11 (2001), 87–110.
- [3] BRUSILOVSKY, P., EKLUND, J., AND SCHWARZ, E. Web-based Educations for All: A Tool for Development Adaptive Courseware. In *Proceedings of the Seventh International World Wide Web Conference, WWW'98* (1998).
- [4] BRUSILOVSKY, P., AND MAYBURY, M. *The Adaptive Web*. Communications of the ACM, 2002.
- [5] BRUSILOVSKY, P., SCHWARZ, E., AND WEBER, G. A tool for developing adaptive electronic textbooks on WWW. In *Proceedings of WebNet'96 - World Conference of the Web Society* (Boston, MA, USA, June 1996).
- [6] DOLOG, P., GAVRILOAIE, R., NEJDL, W., AND BRASE, J. Integrating adaptive hypermedia techniques and open rdf-based environments. In *International World Wide Web Conference* (Budapest, Ungarn, May 2003).
- [7] DUFFY, T., AND JONASSEN, D., Eds. *Constructivism and the Technology of Instruction*. Lawrence Erlbaum Associates, 1992.
- [8] HENZE, N. *Adaptive Hyperbooks: Adpatation for Project-Based Learning Resources*. PhD thesis, University of Hannover, 2000.
- [9] HENZE, N., AND NEJDL, W. Extendible adaptive hypermedia courseware: Integrating different courses and web material. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000)* (Trento, Italy, 2000).
- [10] HENZE, N., AND NEJDL, W. Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems* 12 (2001).
- [11] HENZE, N., AND NEJDL, W. Knowledge modeling for open adaptive hypermedia. In *Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)* (Malaga, Spain, 2002).
- [12] LOM: Draft Standard for Learning Object Metadata. <http://ltsc.ieee.org/wg12/doc.html>.
- [13] REITER, R. A theory of diagnosis from first principles. *Artificial Intelligence* 32 (1987).

- [14] RUSSELL, S., AND NORVIG, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- [15] SCORM: The sharable content object reference model, 2001. <http://www.adlnet.org/Scorm/scorm.cfm>.
- [16] T.BERNERS-LEE, HENDLER, J., AND LASSILA, O. The semantic web. *Scientific American* (May 2001).
- [17] WEBER, G., AND BRUSILOVSKY, P. ELM-ART: An Adaptive Versatile System for Web-based Instruction. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems 12* (2001).
- [18] WEBER, G., KUHL, H.-C., AND WEIBELZAHL, S. Developing adaptive internet based courses with the authoring system NetCoach. In *Proceedings of the Third Workshop on Adaptive Hypermedia, AH2001* (2001).
- [19] WEBER, G., AND SPECHT, M. User modeling and adaptive navigation support in WWW-based tutoring systems. In *Proceedings of the Sixth International Conference on User Modeling, UM97* (Sardinia, Italy, 1997).