A Logical Characterization of Adaptive Educational Hypermedia

Nicola Henze and Wolfgang Nejdl

ISI- Knowledge-Based Systems,
University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
{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.

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. 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 [1], and, associated, the Adaptive Web [8] 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 also 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 [5].:

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 [22] 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.

1.1 Why is a logical characterization of adaptive (educational) hypermedia required?

With Brusilovsky's definition of adaptive hypermedia, we can describe the general functionality of an *adaptive* hypermedia system, and we can compare which kind of adaptive functionality is offered by such a system.
In the literature, we can find reference models for adaptive hypermedia, e.g. the AHAM Reference Model [4], or the Munich Reference Model [20]. Both, the AHAM and Munich Reference Model, extend the Dexter Hypertext Model [16], and provide a framework for describing the different components of adaptive hypermedia systems. The focus of these reference models is on process modeling and engineering of adaptive hypermedia applications, so they are process-oriented and therefore provide process-oriented descriptions of adaptive (educational) hypermedia systems. However, a formal description of adaptive educational hypermedia which allows for a system-independent characterization of adaptive functionality is still missing. Currently, we cannot answer a request like the following: "I want to apply the adaptive functionality X in my system. Tell me what information is required with the hypermedia-documents, which interactions at runtime need to be monitored, and what kind of user model information and user modeling is required." At the moment, we can only describe the functionality with respect to a specific environment, which means we can describe the functionality only in terms of the system that implements it. We cannot compare how different systems implement them, nor can we benchmark adaptive systems. A benchmark of adaptive systems would require at least a comparable initial situation, observations about a user's interactions with the system during some defined interaction period, before the *result* of the system, the *adaptive functionality* as well as the changes in the user model could be compared. The logical definition of adaptive educational hypermedia given in this paper focuses on the components of these systems, and describes which kind of processing information is needed from the underlying hypermedia system (the document space), the runtime information which is required (observations), and the user model characteristics (user model). Adaptive functionality is then described by means of these three components, or more precisely: how the information from these three components, the static data from the document space, the runtime-data from the observations, and the processing-data from the user model, is used to provide adaptive functionality. The aim of this logical definition of adaptive educational hypermedia is to provide a language for describing adaptive functionality, to allow comparison of adaptive functionality in a well-grounded way, and to enable the re-use adaptive functionality in different contexts and systems.
We require a formalism expressing adaptive functionality in a system-independent and re-usable manner, which allows us to apply this adaptive functionality in various contexts. In the educational context, a typical scenario where re-usable adaptive functionality is required would be: Imagine a learner who wants to learn a specific subject. The learner registers to some learning repository, which stores learning objects. According to her/his current learning progress, some of the learning objects which teach the subject s/he is interested in, are useful, some of them require additional knowledge that the learner does not have so far (in accordance to his/her user model), and some might teach the subject only on the surface and are too easy for this learner. This kind of situation has been studied in adaptive educational hypermedia in many applications, and with successful solutions. However, these solutions

are specific to certain adaptive hypermedia applications, and are hardly generalizable for re-use in different applications. Another reason why adaptive functionality is not re-usable today is related to the so-called *open corpus problem* in adaptive (educational) hypermedia, which states that currently, adaptive applications work on a fixed set of documents which is defined at the design time of the system, and directly influences the way adaptation is implemented, e.g. that adaptive information like "required prerequisites" is coded on this fixed set of documents.
A first step to come to re-usable adaptive functionality is to analyze and describe adaptive functionality system-independent, and in a comparable manner, which we undertake in this paper.

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. Based on this formalization, three simple AEHS with few adaptive functionalities are described in section 3. From the many adaptive educational hypermedia systems which have been developed in the past, we have selected four exemplary systems to verify the logical definition of AEHS we propose. In this selection, three of the systems belong to the first generation (Interbook [7]) and the second generation of adaptive educational hypermedia systems (NetCoach [26] and KBS Hyperbook [18]), as well as a recent system which is also an authoring framework for adaptive educational hypermedia (AHA!2.0 [2]) (see section 4). A synopsis of the results is given in section 4.5. 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

(DOCS, UM, OBS, AC)

with

DOCS: Document **S**pace 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 **M**odel: 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: **A**daptation **C**omponent: rules for *adaptive functionality* (e.g. whether to suggest a document for learning, or for generating reasonable learning paths, etc.), rules for *adaptive functionality* (e.g. sorting the links leading to further documents according to their usefulness for a particular user, etc.), etc.

To formalize this above definition we will discuss these components in more detail.

2.1 DOCS: The Document Space

The objects of discourse in the document space are the *documents*, and, if applicable, the knowledge *topics*. Their counterpart in the logical description are the constants: 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

part_of(doc_id_1, 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_id_1, doc_id_2) .

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_id_1, topic_id_2) .

A further example are learning dependencies modeled on topics:

is_dependent(topic_id_1, topic_id_2) .

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 constants, 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, topic) or
has_property(doc_id, user_id, topic, 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:

$\forall$user_id $\forall$doc_id$_1$
($\forall$doc_id$_2$ preq(doc_id$_1$,doc_id$_2$) $\Longrightarrow$ obs(doc_id$_2$, user_id, visited))
$\Longrightarrow$ learning_state(doc_id$_1$, user_id, recommended_for_reading).

The *adaptive functionality* is a set of rules describing the runtime behavior of the system. An often used adaptive functionality is the traffic light metaphor [5] 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). Variations of the colors and their meaning in the various adaptive educational hypermedia systems exist. A rule for defining the adaptive functionality "document annotation" is given in the following:

$\forall$doc_id $\forall$user_id
learning_state(doc_id, user_id, recommended_for_learning)
$\Longrightarrow$ document_annotation(doc_id, user_id, green_icon).

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 [23]. 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: Doc*ument* **S***pace: A finite set of first order logic (FOL) sentences with constants for describing documents (and knowledge topics), and predicates for defining relations between these constants.*

UM: U*ser* **M***odel: A finite set of FOL sentences with constants 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: Obs*ervations: A finite set of FOL sentences with constants for describing observations and predicates for relating users, documents / topics, and observations.*

AC: A*daptation* **C***omponent: 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 functionalities for a user (AC).

3 Examples

In this section we will provide some examples of a prototypical AEHS to illustrate the applicability of our 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 [26], the AHA!2.0 system [2], the Interbook system [7], and the KBS hyperbook system [18].

3.1 A 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.

Simple: Document Space A set of n constants (n corresponds to the number of documents in the document space) which represent the documents:

D_1, D_2, ..., D_n.

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:

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

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

U_1, U_2, ..., U_m.

Simple: Observations One constant for the observation whether a document has been visited:

Visited.

And a set of predicates

obs(D_i, U_j, Visited) for certain D_i, U_j.

Simple: Adaptation Component One constant for describing the values of the adaptive functionality "learning_state":

Recommended_for_reading,

and two constants representing values of the adaptive functionality:

Green_Icon, Red_Icon.

Rules for describing the learning state of a document

$\forall U_i \forall D_j$
$(\forall D_k \text{preq}(D_j, D_k) \Longrightarrow \text{obs}(D_k, U_i, \text{Visited}))$
$\Longrightarrow \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading})$.

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

$\forall U_i \forall D_j$
$\text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading})$
$\Longrightarrow \text{document\_annotation}(D_j, U_i, \text{Green\_Icon})$,
$\forall U_i \forall D_j$
$\neg\ \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading})$
$\Longrightarrow \text{document\_annotation}(D_j, U_i, \text{Red\_Icon})$.

3.2 A 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: Document Space Same as the document space in Simple.

Simple 1: User Model As the user model in 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. *Simple 1* uses an additional constant for describing user characteristics:

Learned.

A document D is assumed to be learned by a user, if it has been visited,

$\forall U_i \forall D_j$
$\text{obs}(D_j, U_i, \text{Visited}) \Longrightarrow \text{p\_obs}(D_j, U_i, \text{Learned})$.

or if a document D', for which D is a prerequisite, has been visited:

$\forall U_i \forall D_j$
$(\exists D_k \text{preq}(D_k, D_j) \wedge \text{obs}(D_k, U_i, \text{Visited}))$
$\Longrightarrow \text{p\_obs}(D_j, U_i, \text{Learned})$.

These inference rules process an observation, they are therefore abbreviated by p_obs for process observation.

Simple 1: Observations Same as Simple.

Simple 1: Adaptation Component The rule describing the learning state of a document is updated as follows:

$\forall U_i \forall D_j$
$\forall D_k(\text{preq}(D_j, D_k) \Longrightarrow (\text{ obs}(D_k,U_i,\text{Visited}) \vee \text{p\_obs}(D_k,U_i,\text{Learned}))$
$\Longrightarrow \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}).$

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

3.3 A simple AEHS - Extension 2

We can extend this simple AEHS by using a knowledge graph instead of 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 (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: Document Space The document space contains all axioms of the document space of Simple, but does not contain any of the predicates. In addition, it contains a set of s constants (s corresponds to the number of topics in the knowledge space) which name the knowledge topics:

$T_1, T_2, \ldots, T_s.$

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

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

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):

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

Simple 2: User Model The user model is the same as in 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 constant

Learned.

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

$\forall U_i \forall T_j$
$(\exists D_k \text{keyword}(D_k, T_j) \wedge \text{obs}(D_k,U_i,\text{Visited})$
$\Longrightarrow \text{p\_obs}(T_j,U_i,\text{Learned}).$

Simple 2: Observations Are the same as in Simple.

Simple 2: Adaptation Component The adaptation component of Simple 2 contains two further constants (im comparison to Simple) representing new values for the learning state of a document,

Might_be_understandable, Will_become_understandable.

and two further constants 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

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ \forall \mathrm{T}_k \Big(\mathrm{keyword}(\mathrm{D}_j, \ \mathrm{T}_k) \implies \big(\forall \mathrm{T}_\ell \mathrm{depends}(\mathrm{T}_k, \ \mathrm{T}_\ell) \implies \mathrm{p\_obs}(\mathrm{T}_\ell, \ \mathrm{U}_i, \\ \mathrm{Learned}) \big) \Big) \\ \implies \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Recommended\_for\_reading}).$$

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

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ (\ \forall \mathrm{T}_k \mathrm{keyword}(\mathrm{D}_j, \ \mathrm{T}_k) \implies \\ (\ \exists \mathrm{T}_\ell \mathrm{depends}(\mathrm{T}_k, \ \mathrm{T}_\ell) \implies \mathrm{p\_obs}(\mathrm{T}_\ell, \mathrm{U}_i, \mathrm{Learned}) \ ) \ ) \\ \wedge \neg \ \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Recommended\_for\_reading}) \\ \implies \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Might\_be\_understandable}).$$

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:

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ \exists \mathrm{T}_k \mathrm{keyword}(\mathrm{D}_j, \ \mathrm{T}_k) \implies \\ (\exists \mathrm{T}_\ell \mathrm{depends}(\mathrm{T}_k, \ \mathrm{T}_\ell) \implies \mathrm{p\_obs}(\mathrm{T}_\ell, \mathrm{U}_i, \mathrm{Learned}) \ ) \\ \wedge \neg \ \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Might\_be\_understandable}) \\ \implies \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Will\_become\_understandable}).$$

Four rules describe the adaptive link annotation:

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Recommended\_for\_reading}) \\ \implies \mathrm{document\_annotation}(\mathrm{D}_j, \mathrm{U}_i, \mathrm{Green\_Icon})$$

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Will\_become\_understandable}) \\ \implies \mathrm{document\_annotation}(\mathrm{D}_j, \mathrm{U}_i, \ \mathrm{Orange\_Icon})$$

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Might\_be\_understandable}) \\ \implies \mathrm{document\_annotation}(\mathrm{D}_j, \mathrm{U}_i, \ \mathrm{Yellow\_Icon})$$

$$\forall \mathrm{U}_i \forall \mathrm{D}_j \\ \neg \ \mathrm{learning\_state}(\mathrm{D}_j, \ \mathrm{U}_i, \ \mathrm{Recommended\_for\_reading}) \\ \implies \mathrm{document\_annotation}(\mathrm{D}_j, \mathrm{U}_i, \ \mathrm{Red\_Icon})$$

3.4 Summary of first three examples

We can now easy 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 functionality (e.g. green_icon, red_icon, etc.).

System	DOCS	UM	OBS
Simple	D_1, D_2, ..., D_n.	U_1, U_2, ... U_m.	Visited.
Simple 1	D_1, D_2, ..., D_n.	U_1, U_2, ... U_m, Learned.	Visited.
Simple 2	D_1, D_2, ..., D_n, T_1, T_2, ... T_s.	U_1, U_2, ..., U_m. Learned.	Visited.

System	AC–Learning State	AC–Adaptive Link Annotation
Simple	Recommended_for_reading.	Green_Icon. Red_Icon.
Simple 1	Recommended_for_reading.	Green_Icon. Red_Icon.
Simple 2	Recommended_for_reading. Might_be_understandable. Will_become_understandable.	Green_Icon. Red_Icon. Orange_Icon. Yellow_Icon.

Table 1. Constants used in *Simple, Simple 1* and *Simple 2*.

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	AC
Simple	preq(D_i, D_j).	–	obs(D_k, U_j, Visited).	–
Simple 1	preq(D_i, D_j).	–	obs(D_k, U_j, Visited).	–
Simple 2	keyword(D_i,T_j) depends(T_j,T_k).	–	obs(D_k, U_j, Visited).	–

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

System	DOCS	UM	OBS
Simple	–	–	–
Simple 1	–	p_obs(D_i,U_j,Learned)	–
Simple 2	–	p_obs(D_i,U_j,Learned)	–

System	AC–Learning State	AC–Adaptive Link Annotation
Simple	learning_state(D_i,U_j,X), X is a constant from AC	document_annotation(D_i,D_j,Y), Y is an constant of AC
Simple 1	”	”
Simple 2	”	”

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

4 Logical Characterizations of four exemplary Adaptive Educational Hypermedia Systems

In this section, we give the logical characterization of four existing adaptive educational hypermedia systems: NetCoach [26] by Gerhard Weber et. al., AHA!2.0 [2] by Paul de Bra et. al. , Interbook [10] by Peter Brusilovsky et. al., and KBS Hyperbook [18] by Nicola Henze et. al.

4.1 NetCoach

NetCoach [26] is the successor of ELM-ART II [27] 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" [26]. This knowledge base is the "basis for adaptive navigations support" [26] in NetCoach, and authors "can create content-specific relations" between the concepts in the knowledge base [26]. We can formalize NetCoach in the following way:

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

$\mathrm{D}_1, \ldots, D_n, \mathrm{TG}_1, \ldots, \mathrm{TG}_k, \mathrm{TI}_1, \ldots, \mathrm{TI}_l$.

NetCoach uses a concept space $\mathcal{C}$ for internally representing 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 provides additional input for adaptation, 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 $\mathrm{D}_i$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.

$\mathrm{preq}(\mathrm{D}_i, \mathrm{D}_j)$ for certain $\mathrm{D}_i \neq \mathrm{D}_j$.

An infers-relations assigns a set of documents to a document $\mathrm{D}_i$that can be inferred to be learned whenever $\mathrm{D}_i$has been learned.

$\mathrm{infer}(\mathrm{D}_i, \mathrm{D}_j)$ for certain $\mathrm{D}_i \neq \mathrm{D}_j$.

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 $\mathrm{D}_i$the set of documents which are sub-documents of D_i:

$\mathrm{part\_of}(\mathrm{D}_i, \mathrm{D}_j)$ for certain $\mathrm{D}_i \neq \mathrm{D}_j$.

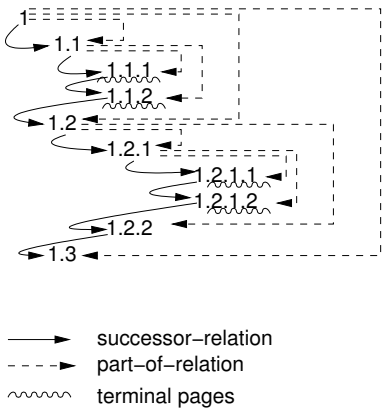

Fig. 1. The hierarchy of documents/concepts in NetCoach

Recursively, all documents that are related via part-of-relations can be calculated by calculating the transitive closure of the set of part-of-relations.
The successor-relation assigns for each document the next document in sequence. This is done by following the hierarchical structure step by step.

succ(D_i, D_j) for certain $D_i$and one $D_j \neg D_i$.

The terminal-page flag is set whenever a document has no sub-documents at all.

terminal_flag(D_i) for certain D_i

NetCoach uses "test-groups" which are sets of test-items. Test-groups need not be disjunct. Test-items and test-groups are used to "assess the user's current learning state of a concept". NetCoach explicitely distinguishes between documents and test-items [25].
A test-assignment, which assigns certain test-items (TI) or test-groups (TG) to a document, is given by:

test_assignment(D_i, TG_j) for certain $D_i$and TG_j.
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 testitems and testgroups is sufficient to know D_i. This criterion is a numerical value indicating how many distinct testitems need to be successfully mastered for knowing D_i.

criterion(D_i, value) for certain D_i.

NetCoach: Observations Observation in NetCoach are used to develop a multi-layered overlay model [26] with 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 therefore separate observations and processed observations into the components OBS and UM.
The **first layer** in NetCoach describes whether a user $\mathcal{U}$ has already visited the document page P corresponding to concept C (again, observation) is abbreviated by obs):

obs(D_j,U_i, Visited) 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:

obs(TI_k, U_i, Worked_testitem) for certain TI_k, U_i, and
obs(TI_k, U_i, Solved_testitem) for certain TI_k, U_i.

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 un-mark concepts to be known as they like, if they pass testitems for a concept the expectation that this concept is learned rises, etc.

obs(D_j, U_i, Marked) for certain D_j, U_i.

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

solved_testitems(U_i, D_j) = [].
$\forall D_j \forall U_i$
$\forall TI_k$test_assignment(D_j,TI_k) $\wedge$ obs(TI_k, U_i, solved_testitem)
$\Longrightarrow$ solved_testitems(U_i,D_j) = [solved_testitems(U_i,D_j), TI_k].

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

$\forall D_j \forall U_i$
criterion(D_j, Value) $\wedge$ length(solved_testitems(U_i, D_j)) $\geq$ Value
$\Longrightarrow$ p_obs(D_j, U_i, Tested).

The user model infers observations about visited documents to the according prerequisite 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.

$\forall D_k \forall U_i$
$\exists D_j$(infer(D_j,D_k) $\wedge$ p_obs(D_j, U_i, Tested)
$\Longrightarrow$ p_obs(D_k, U_i, Inferred_Known)

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. criterion(D_j, Value) for 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.

$\forall D_j \forall U_i$
p_obs(D_j, U_i, Tested)
$\vee$ (criterion(D_j,0) $\wedge$ (obs(D_j, U_i, Visited) $\vee$ p_obs(D_j, U_i, Inferred_Known)
$\vee$ obs(D_j, U_i, Marked))
$\Longrightarrow$ p_obs(D_j, U_i, Learned).

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 haven been learned by this user:

$$\forall D_j \forall U_i$$
$$\forall D_k(\ \text{preq}(D_j,D_k) \Longrightarrow \text{p\_obs}(D_k,U_i,\text{Learned})\ )$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Green\_Ball})$$

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:

$$\forall D_j \forall U_i$$
$$\exists D_k(\ \text{preq}(D_j,D_k) \wedge \neg\ \text{p\_obs}(D_k,U_i,\ \text{Learned})\ )$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Red\_Ball})$$

Which is equivalent to

$$\forall D_j \forall U_i$$
$$\neg\ \text{document\_annotation}(D_j,\ U_i,\ \text{Green\_Ball})$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Red\_Ball})$$

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:

$$\forall D_j \forall U_i$$
$$\text{terminal\_flag}(D_j)$$
$$\wedge\ \big(\ \text{p\_obs}(D_j,\ U_i,\ \text{Tested}) \vee (\ \text{criterion}(D_j,\ 0) \wedge \text{obs}(D_j,\ U_i,\ \text{Visited})\ )\ \big)$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Yellow\_Ball})$$

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

$$\forall D_j \forall U_i$$
$$\neg\ \text{terminal\_flag}(D_j)$$
$$\wedge\ (\forall D_k \text{part\_of}(D_j,\ D_k) \Longrightarrow \text{p\_obs}(D_k,\ U_i,\ \text{Learned}))$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Yellow\_Ball})$$

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.

$$\forall D_j \forall U_i$$
$$\text{terminal\_flag}(D_j)\ \wedge\ \text{obs}(D_j,\ U_i,\ \text{Inferred\_Known})\ \wedge\ \neg\ \text{p\_obs}(D_j,\ U_i,\ \text{Learned})$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Orange\_Ball})$$

$$\forall D_j \forall U_i$$
$$\neg\ \text{terminal\_flag}(D_j) \wedge (\exists D_k \text{part\_of}(D_j,\ D_k) \wedge \neg\ \text{p\_obs}(D_j,\ U_i,\ \text{Learned})\ )$$
$$\Longrightarrow \text{document\_annotation}(D_j,\ U_i,\ \text{Orange\_Ball})$$

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):

$$\text{learning\_goal(Name)} = [\text{D}_1, \ldots, \text{D}_g].$$

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

$$\begin{aligned}
&\text{learning_goal_complete(Name)} = [\,].\\
&\forall \text{D}_k\\
&\neg\ \text{member}(\text{D}_k, \text{learning_goal_complete(Name)})\\
&\wedge\ (\text{learning_goal}(\text{D}_1, \ldots, \text{D}_k \ldots, \text{D}_g)\\
&\qquad \vee\ (\exists \text{D}_\ell \text{part_of}(\text{D}_\ell, \text{D}_k) \wedge \text{learning_goal}(\text{D}_1, \ldots, \text{D}_\ell \ldots, \text{D}_g))\ )\\
&\Longrightarrow \text{learning_goal_complete(Name)} = [\text{learning_goal_complete(Name)}, \text{D}_k].
\end{aligned}$$

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

$$\begin{aligned}
&\text{sequence_learning_goal(Name)} = [\,].\\
&\forall \text{D}_k\\
&\neg\ \text{member}(\text{D}_k, \text{sequence(Name)})\\
&\wedge\ \text{learning_goal_complete}(\text{D}_1, \ldots, \text{D}_k \ldots, \text{D}_n)\\
&\wedge\ \neg\ (\ \exists \text{D}_\ell \text{learning_goal_complete}(\text{D}_1, \ldots, \text{D}_\ell \ldots, \text{D}_n) \wedge \text{succ}(\text{D}_\ell, \text{D}_k)\ )\\
&\Longrightarrow \text{sequence_learning_goal(Name)} = [\text{sequence_learning_goal(Name)}, \text{D}_k].
\end{aligned}$$

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 $\text{U}_i$as the next best page.

$$\begin{aligned}
&\forall \text{D}_j\\
&\text{learning_goal_complete}(\text{D}_1, \ldots, \text{D}_j \ldots, \text{D}_n)\\
&\wedge\ \text{document_annotation}(\text{D}_j, \text{U}_i, \text{Green_Ball})\\
&\wedge\ (\ \neg\ \exists \text{D}_k \text{learning_goal_complete}(\text{D}_1, \ldots, \text{D}_k \ldots, \text{D}_n) \wedge \text{succ}(\text{D}_k, \text{D}_j))\\
&\Longrightarrow \text{next_best_page}(\text{D}_j, \text{U}_i)
\end{aligned}$$

If the user $\text{U}_i$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 $\text{U}_i$as the next best page.

$$\begin{aligned}
&\forall \text{D}_j \forall \text{U}_i\\
&\text{document_annotation}(\text{D}_j, \text{U}_i, \text{Green_Ball}) \wedge \neg\ (\exists \text{D}_k \text{succ}(\text{D}_k, \text{D}_j)\\
&\Longrightarrow \text{next_best_page}(\text{D}_j, \text{U}_i)
\end{aligned}$$

4.2 AHA!2.0

The AHA!2.0 [2] system is the successor of the AHA! [3] system which started to be developed in 1996. AHA!2.0 is based on the AHAM reference model [4]. AHA!2.0 is a framework for authoring adaptive hypermedia applications and provides a runtime-environment for so authored applications. In the following, we will describe the main techniques available in AHA!2.0.

AHA!2.0: Document Space In AHA!2.0, domain and adaptation information are not separated: "In AHA! the author defines concepts, along with requirements that determine under which conditions the user is 'ready' to access the concept, and generate rules that specify how the browsing behavior of the user translates into user model updates" ([2], page 2).
AHA!2.0 implements adaptation strategies, access strategies, etc. by means of so-called *concepts*. Concepts might be abstract (e.g. coding user characteristics, for describing knowledge, etc.) or used to describe a certain page. Each page in AHA!2.0 needs to be described with at least one of these concepts.
Thus, the document space in AHA!2.0 consists of documents (identified via the URI) and concepts:

$D_1, \ldots, D_n, C_1, \ldots, C_m$.

Each concept can have some of the following attributes which further describe the concept. Here, and in the following, we use as names for the relations the names given in [2]. A concept can have a *name*, and a *description*. In case the concept is non-abstract, it must be associated with the document D it describes:

resource(C, D)

Each concept C has a *requirements expression* used to decide on the suitability of the concept. The expression can be a boolean value, or a complex expression that needs to be evaluated, e.g. like $x > 10$. For more information on expressions we refer the reader to AHA!2.0. For describing purposes, we use the following requirements-flag which is true whenever for a user U the expression associated with the requirements of the concept are is evaluated as being true.

req(C, U).

Furthermore, each concept has one or several *attributes* which indicate whether they are be persistent (like e.g. in the user model for indicating that a pages has been visited) or temporary (like e.g. the "access" attributes which is only evaluates to true if a user is currently visiting/accessing the page, as described in the adaptation component of AHA!2.0).

attributes(C, Att).

with Att(has-Name, has-Type, is-Persistent, is-System, is-Changeable), has-Name $\in$ {access, suitability, knowledge, visited, interest}, has-Type $\in$ {boolean, integer, string}, is-Persistent $\in$ {true, false}, is-System $\in$ {true, false}, and is-Changeable $\in$ {true, false}. In addition, the concept has one default value for each of these attributes. The default value is again an AHA!2.0 expression. The values determine various parameters for the event-condition-action rules that belong to this attribute (the metadata for the event-condition-action rules will be described later). E.g. the has-Name part together with the is-System parameter identifies whether the event-condition-action rules that belong to this attribute should register each access-event (has-Name="access", is-System="true"), or whether it should register an access-event permanently (has-Name="visited", is-System="true"). The value has-Name="knowledge" identifies that the following rule will update the knowledge a user has on this concept, etc.
Each attribute can be attached with event-condition-action rules, e.g. to define how access results in user model updates, or in display characteristics, etc. The different parts of the rules are coded in AHA!2.0 in so called *List-items*.
Each event is generated by the runtime-environment of AHA!2.0 and is associated with the link anchor of a page (see section 4.2 on observations in AHA!2.0).

Each action of these event-condition-action rules belongs to an attribute *Att*, and applies some expression *_expression* to a certain attribute *_attribute* of some concept *_concept*.

action(Att, U, _expression, _attribute, _concept).

N.B.: The action rule may be related to other concepts than the one on which the access-event has been registered, the binding of the action rule to an attribute of some concept given in the action-part of the rule makes this possible.
Further, the action might have an *execution condition*:

req(action(Att, U, _expression, _attribute, _concept))

and a boolean flag *isPropagating* which indicates whether this action is allowed to trigger actions to other attributes of this concept:

isPropagating(action(Att, U, _expression, _attribute, _concept)).

In this way, AHA!2.0 defines all the metadata which is used to construct the event-condition-action rules by means of *attributes* (to define what kind of event will trigger the rule), the *execution condition* (to define the conditions for the rule), the *action* (to define what should be executed) and *propagation flag* to decide whether the execution of this rule will trigger further rules. The runtime-engine of AHA!2.0 uses this metadata to construct event-condition-action rules and to execute them.

Last, the documents in AHA!2.0 can be structured as fragments, where each fragment is a certain part of the document. These fragments are identified in the document via surrounding execution conditions. If a document D consists of of l fragments, $D = \bigcup_{j=1}^{l} F_j$, then for all of these l fragments a separate execution_condition is introduced:

execution_condition(F_j) $\forall j \in \{1, \ldots, l\}$

AHA!2.0: Observations AHA!2.0 uses as observations whether a user is currently accessing a page: This is an access-event generated by the runtime-system which is only temporarily valid and terminates whenever the user accesses another page. All concepts that describe this page will receive this access-event.

access(D, U)

Further, AHA!2.0 registers all these access-events permanently as observations whether the user has visited some page.

access(D, U) obs(D, U, Visited).

$\forall \mathrm{D}_j \forall \mathrm{U}_i$
access(D_j, U_i) $\Longrightarrow$ obs(D_j, U_i, Visited).

AHA!2.0: User Model The user model of AHA!2.0 is an overlay model: For every concept from the document space, an entry is reserved in the user model. For each attribute of a concept the user model stores the *type* of the attribute (boolean, integer, or string), the *value*, and a flag to indicate whether the attribute value is persistent or not.
A specific concept called "personal" is used by the user model to represent information about the user independent from any application domain. E.g. in this personal

concept, information about a user U_i's preferred colors for the adaptive annotation of links is stored.
The observations made in AHA!2.0 are inferred by firing the event-condition-action-rules.
In case an access-event has been generated and propagated to be valid for some concept C_1, an action to process this observation to update the user model will be executed:

access(C, U) $\wedge$ attributes(C, Att) $\wedge$
req(action(Att, U, _expression, _attribute, _concept)) $\Longrightarrow$
execute_action(Att, U, _expression, _attribute, _concept).

If the action which is executed allows propagation, actions of other attributes $\tilde{Att}$ that belong to the same concept C as the attribute that maintains the action will be executed (in the current AHA!2.0, some propagation constraints are introduced that guarantee termination) :

execute_action(Att, U, _expression, _attribute, _concept) $\wedge$
isPropagating(action(Att, U, _expression, _attribute, _concept)) $\wedge$
attributes(C, Att) $\wedge$ attributes (C, $\tilde{Att}$) $\Longrightarrow$
execute_action($\tilde{Att}$, U, _expression, _attribute, _concept).

N.B. the code of the event-condition-action rule is provided by the author of the system in the definition of a concept – therefore in the document space of AHA!2.0.

AHA!2.0: Adaptation Component The adaptive engine of AHA!2.0 is running whenever a page has been accessed, this means that the evaluate_access becomes true for some concepts of the document space.

Adaptive link annotation A link to a document D is a recommended for reading (a *goodlink* in the terminology of AHA!2.0), if it has not been visited so far, and it is described by a concept C whose requirements req(C, U) are fulfilled for the user U:

$\forall$U $\forall$D
$\exists$C resource(C, D) $\wedge$ req(C, U) $\wedge$ $\neg$ obs(D, U, visited)
$\Longrightarrow$ document_annotation(U, D, Good_link).

A link to a document D is *neutral* if it has been visited so far, and it is described by a concept C whose requirements are fulfilled:

$\forall$U $\forall$D
$\exists$C resource(C, D) $\wedge$ req(C, U) $\wedge$ obs(U, D, visited)
$\Longrightarrow$ document_annotation(U, D, Neutral_link).

A link to a document D is *bad* if D is described by a concept C whose requirements are not fulfilled:

$\forall$U $\forall$D
$\exists$C resource(C, D) $\wedge$ $\neg$ req(C, U)
$\Longrightarrow$ document_annotation(U, D, Bad_link).

A link to a document D is *active* if the user U is clicking on the link, e.g. the access-event access(D, U) is true:

$\forall$U $\forall$D
$\exists$C resource(C, D) $\wedge$ access(D,U)
$\Longrightarrow$ document_annotation(U, D, Active_link).

A link to a document D is *external* if there is no concept available which describes the D:

$\forall$U $\forall$D
$\neg$ ($\exists$C resource(C, D)) $\wedge$ $\neg$ obs(D, U, visited)
$\Longrightarrow$ document_annotation(U, D, External_link).

A link to a document D to a page is *external, visited* if there is no concept available which describes D, and there is an observation that U has previously visited the corresponding page:

$\forall$U $\forall$D
$\neg$ ($\exists$C resource(C, D)) $\wedge$ obs(D, U, visited)
$\Longrightarrow$ document_annotation(U, D, Externalvisited_link).

Adaptive content generation In addition, the AHA!2.0 adaptation engine processes the XHTML-Code of the resource which a link points to, and evaluates any occurring <if> tags for allowing the conditional inclusion of fragments.

4.3 Interbook

Interbook [10] 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:

Interbook: Document Space Interbook uses domain concepts which are "elementary pieces of knowledge for the given domain" [7]. 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, ..., D_n, TI_1,..., TI_l, C_1, ..., 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. $D_j$is part of D_i:

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

Recursively, all documents that are related via part-of-relations can be calculated by calculating the transitive closure of the set of part-of-relations.
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 can be derived from the successor relation by successor (D_j,D_i) $\Longrightarrow$ predecessor(D_i,D_j). $D_j$is the successor of D_i:

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 sub-documents at all.

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 $D_i(TI_k)$.

preq(D_i, C_j) for certain D_i, C_j.
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).

out(D_i, C_j) for certain D_i, C_j.
out(TI_k, C_j) for certain TI_k, C_j.

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

obs(D_j, U_i, Visited) for certain D_j, U_i.

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

obs(TI_k, U_i, Solved) for certain TI_k, U_i.

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.

$\forall C_j \forall U_i$
$\exists D_k$obs(D_k, U_i, Visited) $\wedge$ out(D_k, C_j)
$\Longrightarrow$ p_obs(C_j, U_i, Beginner_knowledge)

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.

$\forall C_j \forall U_i$
$\exists D_k \exists D_\ell \quad \neg(D_k = D_\ell) \wedge$ obs(D_k, U_i, Visited) $\wedge$ obs(D_ℓ, U_i, Visited)
$\Longrightarrow$ p_obs(C_j, U_i, Intermediate_knowledge)

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

$\forall C_j \forall U_i$
$\exists TI_k out(TI_k, C_j) \wedge obs(TI_k, U_i, Solved)$
$\Longrightarrow p\_obs(C_j, U_i, Expert\_knowledge)$

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.

$\forall C_j \forall U_i$
$\neg\ p\_obs(C_j, U_i, Expert\_knowledge)$
$\wedge \neg\ p\_obs(C_j, U_i, Intermediate\_knowledge)$
$\wedge \neg\ p\_obs(C_j, U_i, Beginner\_knowledge)$
$\Longrightarrow p\_obs(C_j, U_i, No\_knowledge)$

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:

$\forall D_j \forall U_i$
$\forall C_k(out(D_j, C_k) \Longrightarrow p\_obs(C_k, U_i, Expert\_knowledge)$
$\Longrightarrow document\_annotation(D_j, U_i, Big\_checkmark).$

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

$\forall D_j \forall U_i$
$\forall C_k(out(D_j, C_k) \Longrightarrow p\_obs(C_k, U_i, Intermediate\_knowledge)$
$\wedge \neg\ document\_annotation(D_j, U_i, Big\_checkmark)$
$\Longrightarrow document\_annotation(D_j, U_i, Normal\_checkmark).$

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

$\forall D_j \forall U_i$
$\forall C_k(out(D_j, C_k) \Longrightarrow p\_obs(C_k, U_i, Beginner\_knowledge)$
$\wedge \neg\ document\_annotation(D_j, U_i, Normal\_checkmark)$
$\Longrightarrow document\_annotation(D_j, U_i, Small\_checkmark).$

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:

$\forall D_j \forall U_i$
$\forall C_k(\ preq(D_j, C_k) \Longrightarrow p\_obs(C_k, U_i, Beginner\_knowledge)$
$\Longrightarrow document\_annotation(D_j, U_i, Green\_ball)$

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.

$\forall D_j \forall U_i$
$\forall C_k(\ \text{out}(D_j, C_k) \Longrightarrow \text{obs}(C_k, U_i, \text{Visited})\ )$
$\Longrightarrow \text{document\_annotation}(D_j, U_i, \text{White\_ball})$

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:

$\forall D_j \forall U_i$
$\exists C_k(\ \text{preq}(D_j, C_k) \wedge \text{p\_obs}(C_k, U_i, \text{No\_knowledge})$
$\Longrightarrow \text{document\_annotation}(D_j, U_i, \text{Red\_ball})$

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.

$\text{prerequisite\_based\_help\_concepts}(D_i) = [\ ].$
$\forall C_j$
$\neg\ \text{member}(C_j, \text{prerequisite\_based\_help\_concepts}(D_i)) \wedge \text{preq}(D_i, C_j)$
$\Longrightarrow \text{prerequisite\_based\_help\_concepts}(D_i) =$
$[\text{prerequisite\_based\_help\_concepts}(D_i), C_j].$

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

$\text{prerequisite\_based\_help\_documents}(D_i) = [\ ].$
$\forall D_j \forall C_k$
$\neg\ \text{member}(D_j, \text{prerequisite\_based\_help\_documents}(D_i))$
$\wedge\ \text{member}(C_k, \text{prerequisite\_based\_help\_concepts}(D_i))$
$\wedge\ \text{out}(D_j, C_k)$
$\Longrightarrow \text{prerequisite\_based\_help\_documents}(D_i) =$
$[\text{prerequisite\_based\_help\_documents}(D_i), D_j].$

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 p_obs(D,U_i,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

$\text{prerequisite\_concepts}(D_i) = [\ ].$

$\forall C_j$
$\text{preq}(D_i,\ C_j) \Longrightarrow \text{prerequisite\_concepts}(D_i) = [\text{prerequisite\_concepts}(D_i),$
$C_j]$

$\forall C_j \forall C_k \forall D_\ell$
$\text{member}(C_j, \text{prerequisite\_concepts}(D_i)) \wedge \text{out}(D_\ell, C_k)$
$\wedge\ \text{preq}(D_\ell, C_k) \wedge \neg\ \text{member}(C_k, \text{prerequisite\_concepts}(D_i))$
$\Longrightarrow \text{prerequisite\_concepts}(D_i) = [\text{prerequisite\_concepts}(D_i), C_k]$

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 bound to a document D_i, this set of required documents is also binded to this D_i:

$\forall D_j$
$\exists C_k(\text{out}(D_i,C_k) \vee \text{member}(C_k, \text{prerequisite\_concepts}(D_i))) \wedge \text{out}(D_j, C_k)$
$\Longrightarrow \text{required\_documents}(D_i, D_j)$.

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.

4.4 KBS Hyperbook

The KBS hyperbook system [18] 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 [17]. KBS Hyperbook implements the adaptation component on top of an existing, concept-based hypermedia system.
We can formalize KBS Hyperbook in the following way:

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

$D_1, \ldots, D_n$.

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

role(D_i, Lecture) for certain D_i,
role(D_i, Lecture_Note) for certain D_i,
role(D_i, Course) for certain D_i,
role(D_i, Exercise) for certain D_i,
role(D_i, Project_Description) for certain D_i,
role(D_i, Example) for certain D_i,
etc.

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

$C_1, \ldots, C_s$.

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

role(C_i, Introduction) for certain C_i,
role(C_i, Concept) for certain C_i.

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 therefore 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.

KBS Hyperbook: Observations KBS Hyperbook stresses the importance of *active learning*. For this purpose, KBS Hyperbook employs constructivist learning strategies [14]. 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, Expert_knowledge) for certain C_j, U_i,
obs(C_j, U_i, Advanced_knowledge) for certain C_j, U_i,
obs(C_j, U_i, Beginner_knowledge) for certain C_j, U_i,
obs(C_j, U_i, Novice_knowledge) for certain C_j, U_i.

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 [18]. 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:

$\forall C_j \forall U_i\, P(C_i = F) + P(C_i = A) > P(C_i = E) + P(C_i = N)$
$\quad \Longrightarrow$ p_obs(C_j,U_i, Known)
$\forall C_j \forall U_i\, P(C_i = E) + P(C_i = F) > P(C_i = A) + P(C_i = N)$
$\quad \Longrightarrow$ p_obs(C_j,U_i, Well_known)
$\forall C_j \forall U_i\, P(C_i = E) > P(C_i = F) + P(C_i = A) + P(C_i = N)$
$\quad \Longrightarrow$ p_obs(C_j,U_i, Excellently_known)
$\forall C_j \forall U_i\, P(C_i = A) + P(C_i = N) > P(C_i = E) + P(C_i = F)$
$\quad \Longrightarrow$ p_obs(C_j,U_i, Partly_known)
$\forall C_j \forall U_i\, P(C_i = N) > P(C_i = E) + P(C_i = F) + P(C_i = A)$
$\quad \Longrightarrow$ p_obs(C_j,U_i, Not_known)

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:

$\forall C_j \forall U_i$
p_obs(C_j, U_i, Known) $\vee$ p_obs(C_j, U_i, Well_known)
$\vee$ p_obs(C_j, U_i, Excellently_known)
$\Longrightarrow$ p_obs(C_j, U_i, Child_Known)

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:

$\forall C_j \forall U_i$
p_obs(C_j, U_i, Well_known) $\vee$ p_obs(C_j, U_i, Excellently_known)
$\Longrightarrow$ p_obs(C_j, U_i, Parent_Known)

KBS Hyperbook: Adaptation Component

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

$\forall D_j \forall U_i \forall C_k \forall C_\ell$
(keyword(D_j, C_k) $\Longrightarrow$
(depends(C_k,C_ℓ) $\Longrightarrow$ p_obs(C_ℓ, U_i, Child_known)))
$\Longrightarrow$ document_annotation(D_j, U_i, Green_ball) .

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:

$\forall D_j \forall U_i \forall C_k$
(keyword(D_j, C_k) $\Longrightarrow$ p_obs(C_k, U_i, Parent_known))
$\Longrightarrow$ document_annotation(D_j, U_i, White_ball) .

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

$\forall D_j \forall U_i$
$\neg$ (document_annotation(D_j, U_i, Green_ball))
$\wedge \neg$ (document_annotation(D_j, U_i, White_ball))
$\Longrightarrow$ document_annotation(D_j, U_i, Red_ball).

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.

learning_goal(U_i) = (C_1, ..., 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.

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: p_obs(C_j, U_i, Known) or p_obs(C_j, U_i, Well_known) or p_obs(C_j, U_i, 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, ..., C_n].

$$\begin{aligned}&\text{candidate\_for\_sequence}(H, (c_1, \ldots, c_n)) \longleftarrow \\ &(c_{H_1}, \ldots, c_{H_n}) \subset (c_1, \ldots, c_n) \wedge \text{ index } ((c_{H_1}, \ldots, c_{H_n}), H) \\ &\wedge ((c_{H_1}, \ldots, c_{H_m}) \subset (c_1, \ldots, c_n) \wedge \text{ index } ((c_{H_1}, \ldots, c_{H_m}), H) \\ &\Rightarrow (c_{H_1}, \ldots, c_{H_m}) \subset (c_{H_1}, \ldots, c_{H_n}))\end{aligned}$$

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

- $H \in \text{final\_list}(c_1, (c_1, \ldots, c_n)) \longleftarrow \text{ index } (c_1, H)$
- $\forall (c_1, \ldots, c_i) \subset (c_1, \ldots, c_n)$
  $(H \in \text{final\_list}((c_1, \ldots, c_i), (c_1, \ldots, c_n)) \longleftarrow$
  $\text{index } (c_i, H) \wedge \neg (H \in \text{final\_list}((c_1, \ldots, c_i), (c_1, \ldots, c_n))))$

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.

$$\begin{aligned}&\forall \mathrm{C}_i \\ &\text{role}(\mathrm{C}_i, \text{ Introduction}) \vee \neg \ (\ \exists \mathrm{C}_j \text{depends}(\mathrm{C}_i, \mathrm{C}_j) \ ) \\ &\Longrightarrow \text{in\_glossary}(\mathrm{C}_i).\end{aligned}$$

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 :

$$\begin{aligned}&\text{information\_index}([\mathrm{C}_1, \ldots, \mathrm{C}_g]) = [\ ] \ . \\ &\forall \mathrm{C}_i \forall \mathrm{D}_j \\ &\text{member}(\mathrm{C}_i, [\mathrm{C}_1, \ldots, \mathrm{C}_g]) \wedge \text{keyword}(\mathrm{D}_j, \mathrm{C}_i) \\ &\wedge \neg \text{ member}(\mathrm{C}_i, \text{information\_index}(\text{learning\_goal}(\mathrm{C}_1, \ldots, \mathrm{C}_g)) \\ &\Longrightarrow \text{information\_index}([\mathrm{C}_1, \ldots, \mathrm{C}_g]) = [\text{information\_index}([\mathrm{C}_1, \ldots, \mathrm{C}_g]), \mathrm{D}_j]\end{aligned}$$

4.5 Synopsis of four exemplary described AEHS

This chapter provides synoptical tables of the logic-based characterization of the adaptive educational hypermedia systems NetCoach [26], Interbook [7], AHA!2.0 [2], and KBS hyperbook [18]. The constants 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, ..., D_n, TG_1, ..., TG_k, TI_1, ..., TI_ℓ.	U_1, ..., U_m, Learned, Inferred_Known, Tested.	Visited, Solved_Testitem, Marked.
AHA!2.0	D_1, ..., D_n, C_1, ..., C_s.	U_1, ..., U_m.	Visited.
InterBook	D_1, ..., D_n, TI_1, ..., TI_ℓ, C_1, ..., C_s.	U_1, ..., U_m, Learned, Beginner, Intermediate, Expert, No_knowledge.	Visited, Solved.
KBS Hyperbook	D_1, ... D_n, C_1, ..., C_s.	U_1, ..., 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.	–
AHA!2.0	Good_link, Neutral_link, Bad_link, Active_link, External_link, Externalvisited_link	– –
Interbook	Small_Checkmark, Normal_Checkmark, Big_Checkmark, Green_Ball, White_Ball, Red_Ball.	– –
KBS Hyperbook	Green_Ball, White_Ball, Red_Ball.	–

Table 4. Constants used in NetCoach, AHA!2.0, Interbook and KBS Hyperbook.

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 now see that the above characterized systems are very similar in their way of employing adaptive functionality - all provide adaptive navigation support (with respect to Brusilovsky's taxonomy of adaptive hypermedia technologies [6]));
- 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 completely different from calculating this characteristic by compiling the transitive closure of prerequisites. However, all the input and output data for the algorithms are clearly described. Therefor, we can take the algorithms as encapsulated building blocks, and the characterization of the interface of these algorithms is described in the logical formalism;
- 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, testitems and knowledge concepts, NetCoach uses documents, test-groups and testitems, etc.;
- we can compare how much the adaptation information is coded already in the document space (like e.g. in NetCoach or AHA!2.0), or whether the document space does only contain few input information for the adaptation (like e.g. in KBS Hyperbook);
- the rules in the adaptation component show which data is processed by the system for making decisions about particular adaptive functionality; decisions;

System	DOCS
NetCoach	preq(D_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)
AHA!2.0	resource(C_i,D_j) (resource $D_j$belonging to C_i) req(C_i, U_j) (requirements of $C_i$which $U_i$needs to fulfill) attributes(C_i,Att_l) (attributes of C_i) action(Att_l, U_j, _expression, _attribute, _concept) (action part of the rule) req(R_i, U_j) (requirements of R_i which $U_i$needs to fulfill) req(action(Att, U, _expression, _attribute, _concept)) (execution condition of the action) isPropagating(action(Att, U, _expression, _attribute, _concept)) (flag indicating whether the execution of an action is propagated) execution_condition(F_k) (conditional execution of fragments of a document D_j)
InterBook	preq(D_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)
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}	–
AHA!2.0	–	access(D_i, U_j) obs(D_i, U_j, X), X ∈ {Visited}	–
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, AHA!2.0, Interbook and KBS Hyperbook.

System	DOCS	UM	OBS
NetCoach	–	Rules to infer p_obs(D_i, U_j, X), X ∈ {Inferred_Known, Learned, Tested}	–
AHA!2.0	–	Event-condition-action rules to update the user model with the values/expressions given in the action-part of the rule	–
InterBook	–	Rules to infer p_obs(C_i, U_j, Learned, X), X ∈ {Expert, Intermediate, Beginner, No_knowledge}.	–
KBS Hyperbook	–	Rules to infer p_obs(C_i, U_j, Learned, X), X ∈ {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 ∈ {Green_Ball, Red_Ball, Yellow_Ball, Orange_Ball}.
AHA!2.0	Rules to infer document_annotation(D_i, U_j, X), X ∈ {Good_link, Neutral_link, Bad_link, Active_link, External_link, Externalvisited_link}.
InterBook	Rules to infer document_annotation(D_i, U_j, X), X ∈ {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 ∈ {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, …, D_ℓ)
AHA!2.0	–
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 …, C_n], U_j), glossary(D_i) learning_goal([C_1 …, C_n]), next_reasonable_goal(U_j) information_index([C_1 …, C_n])

System	AC-Adaptive Content Selection
NetCoach	–
AHA!2.0	Rules to evaluate the execution_condition(F_j) for each fragment F_j of a document D
InterBook	–
KBS Hyperbook	–

Table 6. Rules used in NetCoach, AHA!2.0, Interbook and KBS Hyperbook.

- 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.

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: The *document space* codes in most cases the way how the adaptation is realized. This observation can be directly related to the so-called *open corpus problem* in adaptive hypermedia [19, 6]. So far, adaptive hypermedia systems have been working on a closed set of documents (closed corpus); the documents are fixed at the design time of the system, alterations or modifications are hardly to process. This widely-used closed-corpus explains why the document space *can* carry all this adaptation-related information. On the other hand, this approach

cannot allow to open up the document space or even working in open environments like the Web.
We have seen that the *observations* used by the chosen adaptive educational hypermedia systems are very similar. They monitor whenever a user accesses some document - the "visited observation". If the systems also have an assessment of learners, other observations like "solved-test" are required, which are attached to some specific subset of the document space. This means that the systems do not differ so much in the way they monitor the runtime-interaction of the user, and we can conjecture that some "standard observations" can be introduced and used by adaptive educational hypermedia systems.
The *user modeling* components describe only the user characteristics and some update rules. More sophisticated user modeling approaches like e.g. fuzzy methods, or probabilistic reasoning cannot be described in FOL. The definition of the user model component provides a description on the characteristics the adaptive systems maintain, and which information is required to trigger an update of the user model. The way how this update is realized is not visible in this description - and is not required as the user modeling component does not interact with the other components of the systems directly.
We have seen, that, in contrary to our intentions motivated by the transfer of Reiter's approach [22] to educational hypermedia, we were not able to generalize the diversity of rules for *adaptation* for a meta-description of adaptation. However, 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 [21], SCORM [24] 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 in the document space, the observation component, and for a user's characteristics in the user model. These sources can now be used as a candidate set for meta-data for adaptation.
With our approach, we have described adaptive functionality in a re-usable way: if e.g. the "traffic light annotation" of documents should be implemented, the catalogue of described AEHS can be used to check the requirements for meta-information in the document space, observation information, and user model characteristics. The decision on which adaptive functionality to implemented can be made on estimations on the required overhead in these three parts.
With the emerging semantic web, there is even more the need for comparable, reusable adaptive functionality. If we consider adaptive functionality as a service on the semantic web, we need re-usable adaptive functionality, able to operate on an open corpus - which the web is. Some first approaches to bring adaptive functionality to the semantic web are considered e.g. in [9, 11, 13, 15].

6 Conclusion and Outlook

This paper 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.

In current work, we are applying adaptation functionality as described in this paper to semantic web applications. An demonstrator implementation using the TRIPLE language has been provided in [12], and currently we are developing a personalized search tool in e-Learning [13] and a *personal reader tool* which demonstrates re-usable adaptive functionality in Semantic Web services.

Acknowledgment
We would like to thank Peter Brusilovsky, Gerhard Weber, and Paul de Bra for the discussions on their adaptive educational hypermedia systems, and the anonymous reviewers for their comments on the draft version of this paper.

References

1. Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web. *Scientific American* (May 2001).
2. Bra, P. D., Aerts, A., Smits, D., and Stash, N. AHA! version 2.0: More adaptation flexibility for authors. In *Proceedings of the AACE ELearn'2002 conference* (Oct. 2002), pp. 240–246.
3. Bra, P. D., and Calvi, L. Aha! an open adaptive hypermedia architecture. *New Review of Hypermedia and Multimedia 4* (1998), 115–139.
4. Bra, P. D., Houben, G.-J., and Wu, H. AHAM: A dexter-based reference model for adaptive hypermedia. In *ACM Conference on Hypertext and Hypermedia* (Darmstadt, Germany, 1999), pp. 147–156.
5. Brusilovsky, P. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction 6*, 2-3 (1996), 87–129.
6. Brusilovsky, P. Adaptive hypermedia. *User Modeling and User-Adapted Interaction 11* (2001), 87–110.
7. Brusilovsky, P., Eklund, J., and Schwarz, E. Web-based Educations for All: A Tool for Development Adaptive Courseware. In *Proceedings of the Seventhns International World Wide Web Conference, WWW'98* (1998).
8. Brusilovsky, P., and Maybury, M. *The Adaptive Web.* Communications of the ACM, 2002.
9. Brusilovsky, P., and Nijhawan, H. A framework for adaptive e-learning based on distributed re-usable learning activities. In *In: M. Driscoll and T. C. Reeves (eds.) Proceedings of World Conference on E-Learning, E-Learn 2002* (Montreal, Canada, 2002).
10. 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).
11. Conlan, O., Lewis, D., Higel, S., O'Sullivan, D., and Wade, V. Applying adaptive hypermedia techniques to semantic web service composition. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)* (Budapest, Hungary, 2003).
12. Dolog, P., Henze, N., Nejdl, W., and Sintek, M. Towards an adaptive semantic web. In *Principles and Practive of Semantic Web Reasoning (PPSWR'03)* (Mumbay, India, December 2003).
13. Dolog, P., Henze, N., Nejdl, W., and Sintek, M. Personalization in distributed e-learning environments. In *International World Wide Web Conference* (New York, USA, May 2004).
14. Duffy, T., and Jonassen, D., Eds. *Constructivism and the Technology of Instruction.* Lawrence Erlbaum Associates, 1992.
15. Frasincar, F., and Houben, G. Hypermedia presentation adaptation on the semantic web. In *Proccedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)* (Malaga, Spain, 2002).
16. Halasz, F., and Schwartz, M. The Dexter hypertext reference model. *Communications of the ACM 37*, 2 (1994), 30–39.

17. Henze, N., and Nejdl, W. Extendible adaptive hypermedia courseware: Integrating different courses and web material. In *Proccedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000)* (Trento, Italy, 2000).
18. Henze, N., and Nejdl, W. Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems 12* (2001).
19. Henze, N., and Nejdl, W. Knowledge modeling for open adaptive hypermedia. In *Proccedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)* (Malaga, Spain, 2002).
20. Koch, N. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process.* PhD thesis, Ludwig-Maximilians-Universitt Mnchen, 2001.
21. LOM: Draft Standard for Learning Object Metadata, 2002. http://ltsc.ieee.org/wg12/index.html.
22. Reiter, R. A theory of diagnosis from first principles. *Artifical Intelligence 32* (1987).
23. Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.
24. SCORM: The sharable content object reference model, 2001. http://www.adlnet.org/Scorm/scorm.cfm.
25. 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).
26. 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).
27. 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).