A Statechart-Based Model for Modeling Hypermedia Applications

MARIA CRISTINA FERREIRA DE OLIVEIRA
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - Campus de São Carlos
Email: cristina@icmc.sc.usp.br
MARCELO AUGUSTO SANTOS TURINE
Departamento de Computação e Estatística
Universidade Federal do Mato Grosso do Sul
Email: mast@dct.ufms.br
PAULO CESAR MASIERO
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - Campus de São Carlos
Email: masiero@icmc.sc.usp.br

This paper presents a formal definition for HMBS — the Hypermedia Model Based on Statecharts. HMBS uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of hypermedia applications. Statecharts are an extension of finite state machines and the model is thus a generalization of hypergraph-based hypertext models. Some of the most important features of HMBS are its ability to model hierarchy and synchronization of information; provision of mechanisms for specifying access structures, navigational contexts, access control, multiple tailored versions and hierarchical views. Analysis of the underlying statechart machine allows verification of page reachability, valid paths and other properties, thus providing mechanisms to support authors in the development structured applications.

Categories and Subject Descriptors: F.1.1.[**Computation by Abstract Devices**]: Models of Computation — *relation among models*; I.7.2. [**Text Processing**]: Document Preparation — *hyperdocument/hypermedia*; H.3.3. [Information Storage and Retrieval]: Information Search and Retrieval — *search process*; H.3.4**. [Information Storage and Retrieval**]: Systems and Software — *information networks*; H.5.1. [**Information Interfaces and Presentation**]: Multimedia Information Systems — *hyperdocument navigation and maps.*

General Terms: Design, Languages, Theory, Modeling.

Additional Key Words and Phrases: Hypermedia, Statecharts, Navigational Models, Browsing Semantics, HMBS model.

1. INTRODUCTION

A major challenge facing the author of hyperdocuments and hypermedia applications in general is that of suitably organizing complex material. Systematic approaches for defining the structural organization are especially important in the design of large and complex applications. In this case, the use of a model helps to discipline the authoring activity by encouraging development in a structured fashion, so that the structure is designed before the actual contents are filled into the nodes. The peculiarities of hypermedia (e.g., the role of links, the complexity of structures, the multimedia facilities, the navigation/browsing paradigm, etc.) are encouraging the development of brand new models and design approaches [Bieber and Vitali 1997, Balasubramanian et al. 1997, Rossi et al. 1997]. A number of formal and semi-formal models and methods for describing the design process, the structural organization and/or the browsing semantics of hyperdocument systems and applications have been proposed in the literature.

Garzotto et al. [1993] identify four distinct approaches towards hypertext modeling. Namely, there are "*application-oriented"* models that explicitly model the semantics of specific application domains, such as the model employed in the g-IBIS hypertext tool. A second class includes models that are "*system-oriented"*, rather than application-oriented. These include, for example, the Dexter Hypertext Reference Model [Halasz and Schwartz 1994], and Garg's set-theoretical model [Garg 1988]. Such models try to identify the relevant abstractions found in a wide range of existing (and future) systems. A third class includes the "*behavioral models*", such as the Trellis model [Stotts and Furuta 1989] and Tompa's model [Tompa 1989]. These are concerned with modeling the dynamic behavior associated to hypertext networks and with the browsing semantics adopted for navigating them. A fourth class comprises less formal approaches, which emphasize preferred topological structures as building blocks for defining the structure of hypertext networks, for example, the structures used in HyperCard, Guide and KMS. This classification may now be updated to include the so-called "application design methods" that, in addition to a set of models provide a consistent set of steps that guide authors in process of prepararing the application. This class includes HDM - Hypermedia Design Model [Garzoto et al. 1993], OOHDM - Object-Oriented HDM [Schwabe et al. 1996, Rossi et al. 1997], RMM - Relationship Management Methodology [Isakowitz et al. 1995, Isakowitz et al. 1997] and EORM - Enhanced Object-Relationship Model [Lange 1994].

The model proposed in this paper, *HMBS* (*Hypermedia Model Based on Statecharts*), uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of hypermedia applications. In the context of the aforementioned classification, HBMS may be included in the class of behavioral models, due to its ability to model the browsing semantics of such applications. The browsing semantics adopted is fully compatible with the structural organization defined for the application, and therefore it provides navigational mechanisms that highlight the hierarchical structure common to many hypermedia applications. The hierarchy mechanism provided includes parallel and sequential decompositions with associated semantics. Moreover, as statecharts were designed to model concurrent reactive systems, the HMBS model is suitable for describing concurrency aspects of an application related to the simultaneous presentation of multiple units of information.

Zheng and Pong [1992] also use *statecharts* for hypertext modeling, but they are concerned with specifying the hypertext system user interface behavior, rather than with modeling application structure and contents. They exemplify the use of statecharts to model the behavior of various buttons and frames supported by Guide, a real-life production hypertext system. Although theirs is a potentially interesting use of statecharts, in this paper we adopt a different approach, using them to model the structural organization and the browsing semantics of hypermedia applications. The resulting model may be used for different purposes, such as structuring and analyzing the information, considering design alternatives, organizing the synchronization of multimedia data and also for planning the user interface.

The remainder of this paper is divided as follows. Section 2 describes the formal syntax used in statecharts, introduces an example application and presents the main features of the proposed model. Section 3 describes simple solutions warranted by the model to some common problems related to hypermedia navigation, browsing and analysis. Some brief remarks are presented on the integration of HMBS into a general hypermedia design method. Section 4 discusses related work by comparing HMBS to other approaches. Finally, the conclusions are presented in Section 5.

2. A STATECHART-BASED MODEL

Statecharts constitute a visual formalism for describing states and transitions in a modular fashion. It extends the classical formalism of finite state machines and state transition diagrams by incorporating the notions of hierarchy, orthogonality (concurrency), a broadcast mechanism for communication between concurrent components, composition, aggregation, and refinement of states [Harel 1987a, Harel and

Naamad 1996]. Statecharts provide an effective notation for the specification and design of large and complex reactive systems. In addition to a concise and intuitive visual notation, they have associated formal syntax and semantics, thus enabling the specification of behavioral aspects of systems in a clear, yet rigorous, manner, and providing means for formal verification and validation of models.

The human-computer interaction management module, or user interface, of a hypermedia system may be considered as a reactive system, as it must interactively attend to external events given in the form of user requests during browsing. For example, the activation of an anchor may result in a reconfiguration of the display to show the new set of pages available for navigation. The *HMBS* model uses the structure and execution semantics of statecharts to specify both the contents (including the linked structure) and the browsing semantics of hypermedia applications. This logical structure can be interpreted to generate a final product to be delivered to users. As in other models (e.g., Trellis [Stotts and Furuta 1989]), application contents and linked structure, as well as their mapping into specific physical representations, are effectively separated. Therefore, a single model representation may be used to generate different versions, or different presentations, of the same application.

In Section 2.1 we briefly introduce a subset of the formal syntax for statecharts[1] and provide an informal description of its operational semantics [Harel 1987a, 1987b]. In Section 2.2 we describe a hypothetical example application that shall be used throughout this paper as a running example to illustrate the HMBS model and its concepts. The model is introduced in Section 2.3.

2.1 Background on Statecharts

A *Statechart structure* (*ST*) is a 11-tuple, $ST = <S,\rho,\psi,\gamma,\delta,V,C,E,A,R,T>$, in which:

(a) $S=\{s_1,s_2,...,s_n\}$ is the *set of states*, $n > 0$.

(b) $\rho: S \rightarrow 2^S$ is a *hierarchy function* that defines the sub-states of each state. If $\rho(x)=\rho(y)$ then $x=y$. There exists a unique state $r \in S$, known as the root of the statechart, such that $\forall s \in S\ r \notin \rho(s)$. A state s is *basic* if $\rho(s)=\phi$. The functions ρ^*, ρ^+ are extensions of ρ defined, respectively, by $\rho^*(s)=\cup\rho^i(s)$, $i \geq 0$ and $\rho^+(s)=\cup\rho^i(s)$, $i \geq 1$.

(c) $\psi: S \rightarrow \{AND,OR\}$ is a function that defines the hierarchical decomposition type of each state $s \in S$ such that $\rho(s) \neq \phi$ (s is not a basic state).

(d) $\gamma: H \rightarrow S$ is history function that defines the history symbols associated to OR states. $\gamma(H)=y$ if $y \in S$ and $\psi(y)=\{OR\}$. If $h_1 \in H$ and $h_2 \in H$ and $\gamma(h_1)=\gamma(h_2)$ then $h_1 = h_2$

[1] Other concepts from the original definition might be included, but not all of them are relevant to the problem under consideration.

(e) δ: $S \rightarrow 2^S$ is the *default function* that defines for a state *s* a set of its sub-states known as the *default set* for *s*. Unless otherwise specified, $\delta(s)$ are the sub states entered when s is activated, and the states in $\delta(s)$ are called *initial states*. For $x \in S$, if $x \in \delta(s)$ then $x \in \rho^+(s)$.

(f) V_P is a *set of primitive variables*, and V is a *set of expressions* defined inductively as: if *k* is number then $k \in V$; if $v_1 \in V_P$, $v_2 \in V_P$ and op$\in \{+,-,*,/\}$ then v_1 op $v_2 \in V$.

(g) C_P is a *set of primitive conditions*, and C is a *set of conditions* which include T (*true*), F (*false*) and the following logical expressions: if $c_1 \in C_P$, $c_2 \in C_P$ and op$\in \{=, >, <, \neq, \leq, \geq\}$ then c_1 op $c_2 \in C$; if $c_1 \in C$ and $c_2 \in C$ then $u \vee v$, $u \wedge v$ and $\sim u \in C$.

(h) E_p is the set *of primitive events*, and E is the *set of event expressions* that contains E_p and combinations of primitive events and conditions from the *set of conditions* C. Conditions in labels are placed within brackets:

If $e \in E$ and $c \in C$ then e[*c*]$\in$ E

(i) A is the *set of actions*, and may be defined inductively as: If $c \in C_p$, $d \in C$ then c:=d$\in$ A; If $v_1 \in V_p$, $v_2 \in V$ then v_1:=$v_2 \in$ A.

(j) $R \subset E \times A$ is the *set of labels* formed by set of ordered pairs (e,a) | e$\in$ E and a$\in$ A. Informally, if *e*[*c*]/*a* is a label of a transition *t*, then *t* is triggered if event $e \in E$ is activated with condition $c \in C$ true. Therefore, the action $a \in A$ is executed when *t* is taken.

(k) $T \subset 2^S \times R \times 2^{S \cup H}$ is the *set of transitions*. A transition t = (*X*, *r*, *Y*) is composed of a source set of states *X*, a target set of states *Y* and a label $r \in R$.

The operational semantics of statecharts relies on a discrete time model that assumes that system states have an associated duration, whereas transitions are instantaneous. An "execution step" is composed of several micro-steps, with a single transition being evaluated at each micro-step, thus allowing the simultaneous evolution of multiple concurrent state machines. At each execution step, the statechart assumes a new state configuration corresponding to the set of currently active basic states. The sub-states of an OR decomposition are not allowed to be active simultaneously, whereas the sub-states of an AND decomposition are simultaneously active as long as their father remains active.

A *statechart legal configuration* (SC) has been defined by Harel [1987a]. The *least common ancestor* (LCA) of a set of states X, denoted LCA(X), is a state *x* for which $X \subseteq \rho^*(x)$ (i.e., *x* is a common ancestor) and $\forall s \in \rho^+(x) \rightarrow X \not\subset \rho^*(s)$ (i.e., it is the least such). Two distinct states *x* e *y* are *orthogonal*, denoted $x \perp y$, if *x* = *y* (i.e., a state *x* is considered to be orthogonal to itself) or ψ(LCA({*x*,*y*}))=AND. A set of states is orthogonal if its states are pairwise orthogonal. A *maximal orthogonal* set is an

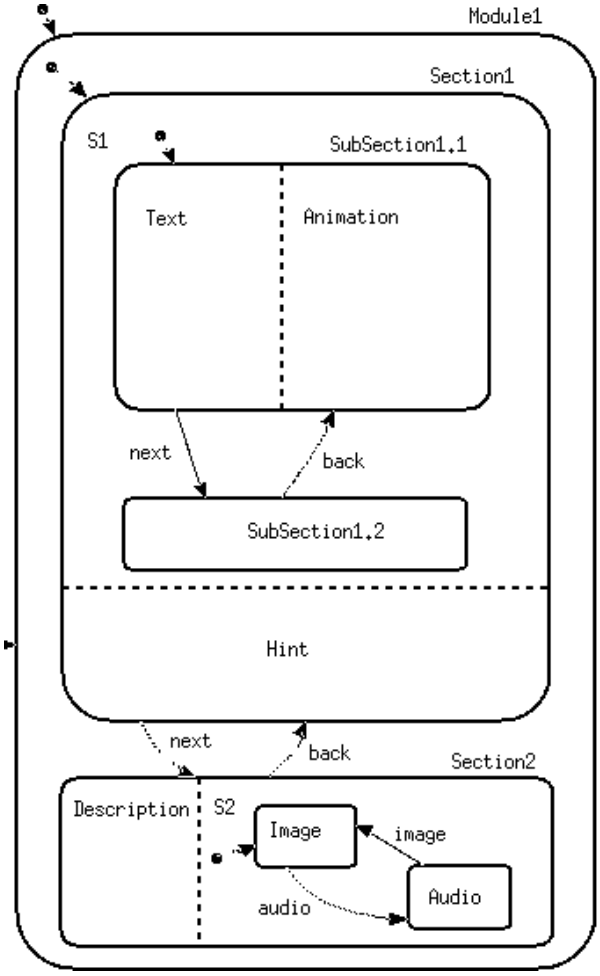


Fig. 1. A statechart example.

orthogonal set that cannot be extended (by adding elements) to a larger orthogonal set. A *legal configuration* is a maximal orthogonal set of *basic states*. A statechart's legal configuration represents a possible set of currently active states of the statechart.

Figure 1 illustrates a simple statechart model. The set of states is given by S={*Module1*, *Section1*, *S1*, *SubSection1.1*, *Text*, *Animation*, *SubSection1.2*, *Hint*, *Section2*, *Description*, *S2*, *Image*, *Audio*}, where *Module1* is the root state, and the basic states are {*Text*, *Animation*, *SubSection1.2*, *Hint*, *Description*, *Image*, *Audio*}. The hierarchy functions for states *Module1* and *Section1*, for example, are given by ρ(*Module1*) = {*Section1*, *Section2*}, ρ(*Section1*) = {*S1*, *Hint*}. The ψ functions for the same states are ψ(*Module1*) = {OR}, ψ(*Section1*) = {AND} (an AND decomposition is visually indicated by a dashed line separating its orthogonal sub-states). The default function for state *Module1* is given by δ(*Module1*) = {*Section1*, *SubSection1.1*}. The set of event expressions is given by E={next, back, audio, image}, and the set of transitions

is T={({*SubSection1.1*},next,{*SubSection1.2*}), ({*SubSection1.2*}, back, {*SubSection1.1*}), ({*Section1*}, next, {*Section2*}), ({*Section2*}, back, {*Section1*}), ({*Image*}, audio, {*Audio*}), ({*Audio*}, image, {*Image*})}. Some of the possible legal configurations for this particular statechart are SC_0={*Text,Animation,Hint*}, SC_1={*SubSection1.2,Hint*} and SC_2={*Description, Image*}.

The hierarchy of states in a given statechart may be described by an AND/OR tree, as states can be repeatedly decomposed either into OR-components or into AND-

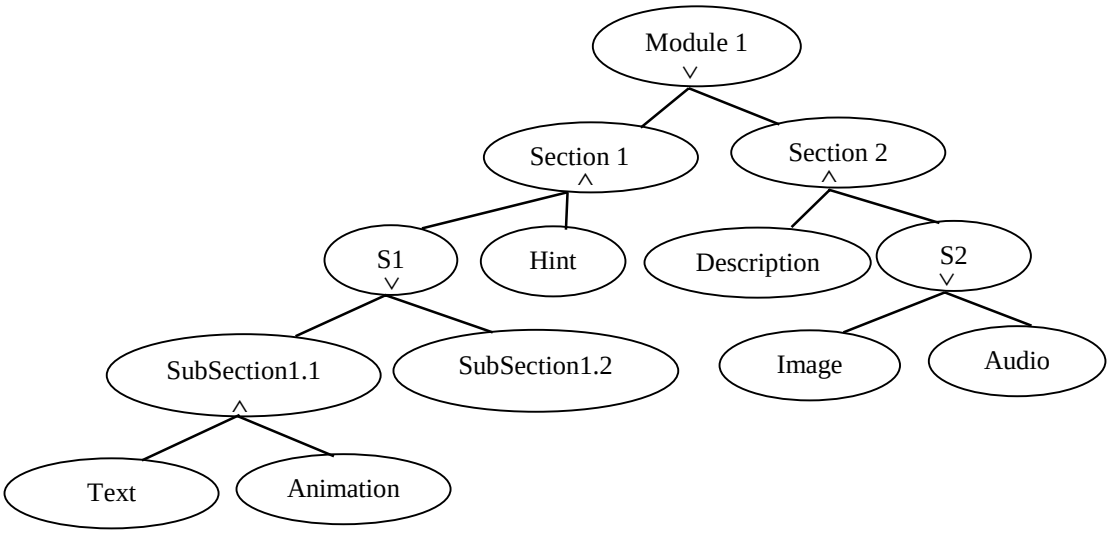


Fig. 2. AND/OR tree for statechart in Figure 1.

components. The AND/OR tree for the statechart in Figure 1 is shown in Figure 2. The symbols "∧" and "∨" in the nodes denote AND an OR decompositions, respectively.

2.2 Model Definition

The HMBS model provides a mechanism for effectively separating the hypermedia physical and logical structures. The model definition is given below:

A *Hypermedia application H* is a 7-tuple *H=<ST,P,m,L,pl, ae,N>* in which

(a) ST=<S,ρ,ψ,γ,δ,V,C,E,A,R,T> is a statechart structure.

(b) P is a *set of pages* (contents) containing units of information. HMBS adopts the terminology *page* to denote atomic items of information that will be exhibited as a whole. Elements from P may contain text, graphics, tables, bitmaps, executable code, video, audio or any other static or dynamic data objects supported by the hypermedia system. The set P also includes a special *null page*, with no associated contents. Each element in set P has two associated attributes: *title* and *contents*. The first provides a unique identifier to the page, whereas the second corresponds to its content information.

(c) m: $S_s \rightarrow P$ is a *value function* mapping states from a set $S_s \subset S$ into HMBS pages. Possible mappings are defined for states in set S_S: $\{x \in S \mid \psi(x)=OR \vee \rho(x)=\phi\}$, where S_S is the subset of S comprising basic and OR states. AND states are not mapped into pages, being used to indicate page composition. A possible mapping of states into pages of a hypothetical hypermedia application for the statechart described in Figure 1 is illustrated in Table I.

(d) L is a *set of presentation channels* for interpreting pages. A channel is an abstract device responsible for exhibiting content information. It acts as an interpreter that displays a page's content through some medium (e.g., text formatters, graphic decoders, program interpreters, audio and video players, data manipulation systems, etc.). The channel device supports the specification of some user interface properties related to pages. For example, text channels allow the setting of font type and size; image channels allow the setting of the position and resolution of an image, and so on.

(e) pl: $P \rightarrow L$ is the page *visualization function* which associates each page with a single *channel* that is able to interpret it.

(f) *ae*: $Anc_P \rightarrow E$ defines a function that associates anchors from a page *p* ($a_p \in Anc_p$) to statechart events that control the firing of transitions. A single event may appear in multiple transitions, but a transition must have a unique event. The page containing the anchor to be mapped into an event must be mapped into a state (or sub-state) in the source set of a transition containing the event. In this way, HMBS supports reuse of link components, as transitions and events may be reused to define anchors in different navigational context. Such restrictions may be formally described as:

Let $t \in T$ be a statechart transition in the form "t = (X,r,Y)", and let $rot:T \times E \rightarrow \{true,false\}$ be a function that verifies whether an event "e_n" is in the label of a transition "t_m", such that $rot(t_m,e_n) = true$ if "e_n" is in the label of "t_m" and $rot(t_m,e_n) = false$ otherwise. Thus, a function "ae" may be defined as:

$ae(a_p)=e \leftrightarrow (\exists\ p \in P \mid p=<c,t,Anc_p> \wedge a_p \in Anc_p) \wedge e \in E \wedge (\exists\ t \in T \mid rot(t,e)=true) \wedge (\exists\ s \in S \mid m(s)=p) \wedge (\exists\ s_1 \in S \mid s \in \rho^*(s_1))$.

(g) N is the hypermedia application browsing level, a natural number defining the visibility level that is used to control the hierarchy depth when displaying HMBS pages during navigation (see Section 2.3 below).

Table I. Mapping functions "m" (states into pages) and "pl" (pages into channels) and page information (content, title and anchors) for the statechart structure depicted in Figure 1.

States (S_S)	Page Title	Page Content	Anchors (Anc_p)	Channel (L)

Module1	TModule1	CModule1	{ }	L_T (Text)
S1	TS1	CS1	{nextSection2 }	L_T (Text)
Text	TText	CText	{nextSubSection1.2, nextSection2}	L_T (Text)
Animation	TAnimation	CAnimation	{ }	L_V(Video)
SubSection1.2	TSubSection1.2	CsubSectionm1.2	{backSubSection1.1, nextSection2}	L_T (Text)
Hint	THint	CHint	{nextSection2}	L_T (Image)
Description	TDescription	CDescription	{backSection1}	L_T (Text)
Image	TImage	CImage	{backSection1, audio}	L_I (Image)
Audio	Taudio	Caudio	{backSection1, image}	L_A (Audio)

Figure 3 illustrates how the statecharts' structural elements (states and transitions) are associated to the hypermedia navigational (pages and anchors) and presentation objects by the mappings defined in HMBS.

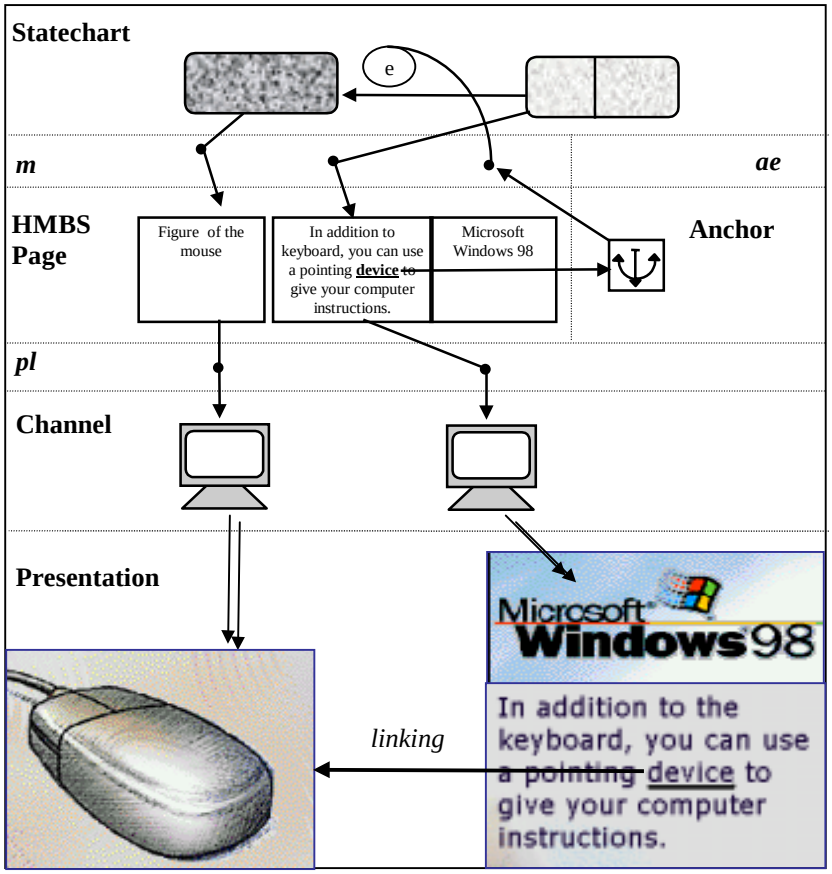


Fig. 3. Relationship between HMBS objects.

2.3 Browsing Semantics

The browsing semantics of a hypermedia application is the way in which information is to be visited and presented to a reader who is navigating through it, i.e., it refers to the dynamic properties of a reader's experience when navigating through a document [Stotts and Furuta 1989]. In this section the browsing semantics associated with the HMBS model is introduced, and some examples are provided to illustrate the power and the flexibility of the statechart model to represent the browsing semantics.

2.3.1 Browsing Semantics Definition

A hypermedia application consists of a statechart structure representing the logical structure of the document in terms of nodes and links, the document's physical structure which are the *human-consumable* components (pages), and the associated display mechanisms. The execution semantics of the underlying statechart provides the model for browsing the application.

A set of statechart states is mapped into pages and the set of events and transitions represent the set of possible link activations. Link activation results in the triggering of an associated event in its underlying statechart model, consequently firing the corresponding transition from a source state (mapped to the page that is the source of the link) to a destination state (mapped to the target page of the link). The current statechart configuration defines a *user configuration* that determines the set of currently accessible pages at a certain stage during browsing. As a valid state configuration may include several states, readers may have concurrent access to multiple pages. Information is presented by exhibiting all the pages associated with states in the *user configuration* defined by the current state configuration, with the states of the set S_S being mapped into pages of the set P according to the given function m

To view a page, the hypermedia system must invoke an associated channel that interprets and exhibits it through some medium. C*hannels* are invoked for each page within the current user configuration. A link activation triggers a corresponding transition enabled by the associated event in the statechart, whose firing results in a new state configuration that defines a new set of pages to be exhibited. Pages associated to ancestral states may also be viewed in conjunction with those associated to basic states in the current configuration, a useful facility for allowing users to simultaneously access application information at different hierarchical levels. The hierarchy level to be displayed is specified through the visibility level N, as described below.

If N=0 *then*

display all pages p in set D_0, $D_0=\{p \mid x \in S_S \wedge m(x) = p \wedge \rho^0(x) \cap SC \neq \phi\}$

If N=1 *then*

display all pages p in set $D_0 \cup D_1$,

$$D_1 = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho^1(x) \cap SC \neq \phi\}$$

If N=2 *then*

display all pages p in set $D_0 \cup D_1 \cup D_2$,

$$D_2 = \{p \mid x \in S_S \wedge M(x) = p \wedge \rho^2(x) \cap SC \neq \phi\}$$

Generalizing,

If N=*n then* display all pages p in set D_{all}, given by

$$D_{all} = \bigcup_{i=0,\ldots,n} Di \text{ where } D_i = \{p \mid x \in S_S \wedge M(x) = p \wedge \rho^i(x) \cap SC \neq \phi\}$$

Under the statechart model, browsing is thus interpreted as follows. When a user selects a link, the hypermedia system browser performs the following actions:

(a) generates an event that causes the triggering of the transitions associated to that link. The source set for the transitions must be an active set; that is, it must be included in the current state configuration.

(b) activates all states that are target sets for the transitions fired, thus generating the next state configuration and disabling the previous one.

(c) invokes the channels for the pages associated with the states in the new configuration and within the scope of the visibility level N.

We shall illustrate the browsing semantics interpretation by considering a hypothetical application: suppose we are to deliver instructional material for training a company's employees in the use of a newly acquired software product. Material consists of an initial presentation of the course and related information, followed by a sequence of training modules plus a set of evaluation tests. Modules are organized in Sections and Sub-sections that may contain text, audio, video and figures. Employees are to take the course remotely, studying the material at their own pace and gradually taking the tests to assess their learning.

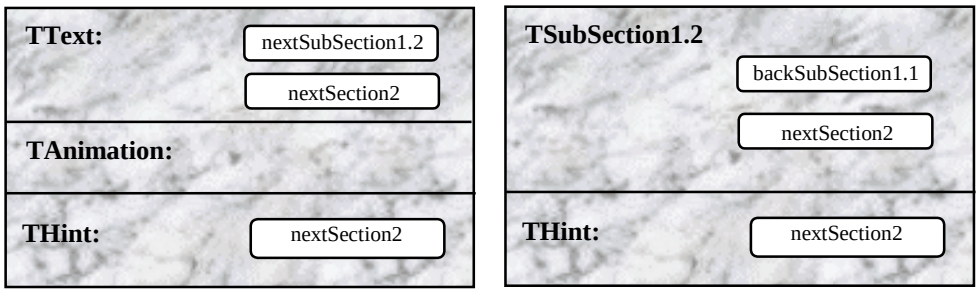
In this scenario, the statechart shown in Figure 1 may depict the organization of one particular module of the instructional material. This module may contain two sections, with Section 1 comprising two subsections and a text that is to be exhibited all the time the reader remains in Section 1. Subsection 1.1 contains a text and an animation to be exhibited simultaneously, whereas Subsection 1.2 contains text. Section 2 contains a textual description that remains on screen during an exhibition of an image or, alternatively, an audio.

Assuming an initial state configuration SC_0 = {*Text*, *Animation*, *Hint*}, and the mapping functions “m” and “pl” as shown in Table I, let us illustrate possible user configurations to be reached during browsing according to the given interpretation. Such

configurations are illustrated with hypothetical layouts for the display window, arbitrarily assuming that pages are mapped into window frames and that anchors signaling available links are represented as buttons. The model does not impose any restrictions on the nature of the interface objects associated to pages or links, however.

If a visibility level N = 0 is specified, a schematic layout for the window displaying the application at this stage during browsing is presented in Figure 4(a). The pages associated to the three basic states in the current state configuration (SC_0) are exhibited. The event labeled 'next' in transition ({*SubSection1.1*},*next*,{*SubSection1.2*}) of the statechart of Figure 1 has been mapped into the anchor labeled *nextSubSection1.2* that appears in page with title *TText*, associated to state *Text* sub-state of *SubSection1.1*, which has no associated page. If anchor *nextSubSection1.2* is selected, then the event *next* is produced, triggering the transition. Consequently, state *SubSection1.1* is left and, following the statechart semantics, so are both states *Text* and *Animation* simultaneously. A new state configuration SC_1 = {*SubSection1.2, Hint*} is reached, resulting in a new layout for the display, illustrated in Figure 4(b).

As N = 0, only the pages associated to basic states in the current state configuration and the anchors associated to transitions enabled for such states are shown. In our instructional application example, it could mean that only topics at the lowest hierarchical organizational level are being exhibited. The policy of not associating pages to AND states, but rather using them to denote concurrent presentation of pages, follows the same principle adopted by Drusinsky and Harel [Drusinsky and Harel 1988] in their algorithm devised to implement programmable logic arrays specified by statecharts.

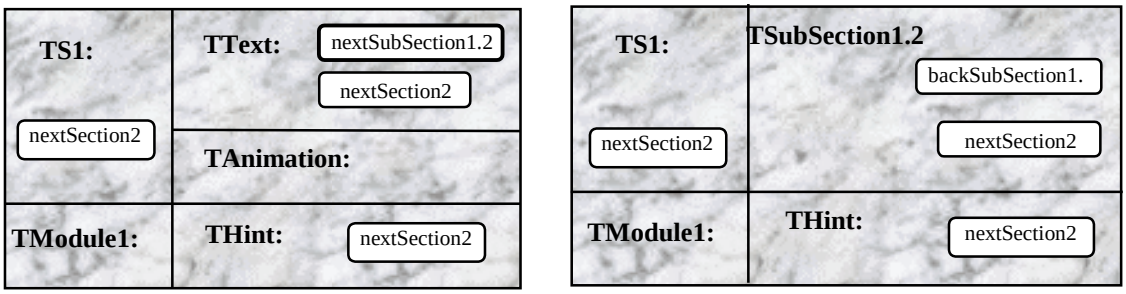


(a) SC_0={*Text, Animation, Hint*} (b) SC_1 = {*SubSection1.2, Hint*}

Fig. 4. Possible window layouts displayed during the browsing of the hypermedia application for the given configurations and N=0.

If N = 1 and the current state configuration is SC_0 = {*Text, Animation, Hint*}, a possible layout for the display window is depicted in Figure 5(a). In this situation, the visibility level is set in order to show not only the pages associated to the basic states in

the current configuration, but also the pages associated to their parent states in set S_S. The first ancestor of states *Text* and *Animation*, which is not an AND state is *S1* (see Figure 1), whose associated page is exhibited, and the first ancestor of state *Hint*, which is not an AND state is *Module1*. Both ancestors *SubSection1.1* and *Section1* are AND states with no associated pages. Thus, the application hierarchical node structure up to a depth of one is displayed. Selecting anchor *nextSubSection1.2* in page *TText* leads to configuration SC_1, for which a possible display window for N = 1 is depicted in Figure 5(b). When presenting instructional material as in the example provided, setting up a hierarchical level equal to one allows, for example, the simultaneous exhibition of content attributed to Subsections and to Sections of the material.



(a)SC_0={*Text,Animation,Hint,S1,Module1*}(b)SC_1={*SubSection1.2,S1,Hint, Module1*}

Fig. 5. Possible window layouts displayed during the browsing of the hypermedia application for the given configurations and N = 1.

2.4 Hierarchical Views

Mechanisms for navigating through the hierarchical structure of the application are highly desirable, particularly for providing “structured navigation”, according to the terminology introduced in [Thüring et al. 1995]. This type of navigation is strongly supported by the model, and in this section we describe how operations for exploring hierarchy in hypermedia applications may be implemented at the interface level of any hypermedia system that supports HMBS.

To better explore the hierarchical organization of the statecharts, at the beginning of a browsing session we may set up an auxiliary variable h = N+1, where N is the desired browsing level. We may then define a *Show-hview* operation as a hierarchical view function that shows the pages associated to the states immediately above those in level N. Formally:

Show-hview :

Let *d* be the depth[2] of the statechart's AND/OR tree;

If $h \leq d$ *then*

display all pages *p* in set D_h, $D_h = \{p \mid x \in S_S \wedge m(x) = p \wedge \rho^h(x) \cap SC \neq \phi\}$

Considering Figure 4(a) with N = 0, if the user executes *Show_hview* the hypermedia system might produce a hypothetical screen depicted in Figure 6, which exhibits a hierarchical view window (*Show_hview*) partially overlapping the main window.

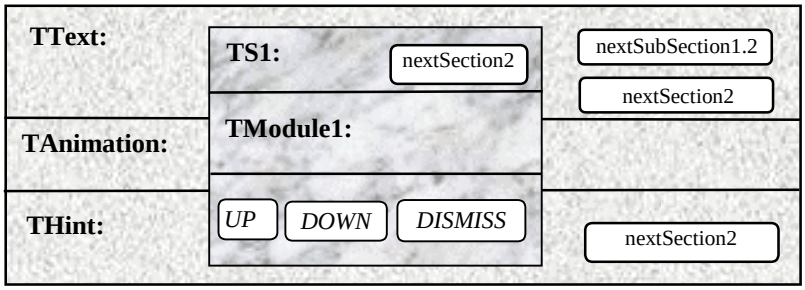


Fig. 6. Activating *Show_hview* during browsing.

The *Show_hview* operation enables three additional operations: UP, DOWN and DISMISS. The only action associated to the latter is the removal of the *Show_hview* window from the display. The *UP* and *DOWN* operations alter the current value of *h*, thus changing the pages displayed as a result of the hierarchical view operation:

UP: *If* $h < d$ *then* $h = h+1$; *Show_hview*.

DOWN: *If* $h > N$ *then* $h = h-1$; *Show_hview*.

Note that execution of *Show_hview* does not modify the current statechart configuration but only provides access to pages associated to states which are one level up (or down) in the hierarchy, as long as at least one such a state exists. If there is no OR state up in the hierarchy (or, alternatively, no OR or basic state down), the page associated to the state in the current level remains shown. Such a situation is illustrated analyzing the configuration depicted in Figure 6, and assuming that operation UP was activated. The pages shown in the hierarchy window as a result of such activation would be those associated to state *Module1* (the common ancestor of both states *S1* and *Module1* itself). If it were activated again this same page remains in the hierarchy window.

In addition to navigating hierarchically through the document by activating operations UP and DOWN, the user has a new set of enabled links as a result of the *Show_hview* operation, corresponding to those anchors in the new pages displayed. When one of such

[2] The *depth* (or height) of a binary tree is defined as the maximum level of its leaves. The level of a node is defined recursively as 1 plus the level of its parent. The root node has level zero by definition.

anchors is selected and the corresponding event is activated one of two possible interpretations may be taken:

(a) the associated transition fires and the new screen layout displays the pages associated to the new statechart configuration and the pages resulting from applying the *Show_hview* operation on this configuration. The value of *h* is maintained.

(b) the associated transition fires and the new screen layout displays the pages associated to the new statechart configuration.

The first alternative may be interesting if the application presents a uniform hierarchical structure (e.g., if it is a book). However, it may disorient the user if that is not the case. We prefer the second approach. A third option would be not to allow users to activate links enabled as a result of a *Show_hview* operation and thus such an operation should be available only for inspecting pages up or down in the hierarchy. We believe this alternative is too restrictive, however.

2.5 Access to Pages

Initial access to pages may be granted either by starting browsing from the set of pages associated to the initial configuration of the underlying statechart, or by selecting a page from an index of page titles. In the latter case the selected page will be exhibited in conjunction with other pages that together define a visualization context. This context may be assembled by an algorithm from the set of default states composing a valid state configuration that includes the one associated to the selected page. For example, after selecting from the index the page with title *TText* (refer to Figure 1 and Table I), and considering N = 0, the set of pages shown would be given by {*TText*, *TAnimation*, *THint*}. If page *TSection2* is selected and N = 1, the next set of pages would be {*TSection2*, *TModule1*}. When a reader selects a title associated to a page/state hierarchically above N, a system might either show only the pages satisfying the current display level, or use the *Show-hview* operation to display the selected page in addition to those that satisfy the display level.

Most of the common access structures proposed in the literature, such as indexes, guided tours and indexed guided tours can be modeled in HMBS using a state that represents the access structure with outgoing transitions to states that correspond to the pages to be visited. Pages belonging to multiple navigation contexts [Schwabe et al. 1996] may have conditions attached to their incoming transitions within a specific context, enabling authors to prevent readers from changing context during a browsing section. Such a situation is in Section 3.3.

3. SOLUTIONS USING THE MODEL

The HMBS model lends itself to providing simple solutions to some common problems related to visualization, browsing and analysis in hypermedia applications. Alternative solutions to some of the problems discussed in [Stotts and Furuta 1989] are described below.

3.1 Node Reachability and Display Characteristics

The HMBS model allows the determination of some parameters relevant to the physical presentation of the application and for analyzing page reachability and browsing sequence restrictions. A statechart reachability tree describes every possible computation sequence for the statechart [Masiero et al. 1994b]. At each step it is assumed that an event expression evaluates to true thus firing a transition that leads to a new state configuration computed according to the semantic rules defined in [Harel 1987b]. The computation evolves in a sequence of time steps, which can be represented by the set $\{(SC_i, \gamma_i), i \geq 0\}$, where

(a) $SC_0 = (\mathrm{X}_0, \Pi_0, \Theta_0)$, where X_0 is the initial configuration, Π_0 is the set of external events generated by the environment and Θ_0 is the set of primitive conditions valued true.

(b) γ_i is a step taken in SC_i, at time step $\sigma_{i+1}, i \geq 0$.

We then have a sequence of state configurations, where SC_{i+1} is the configuration reached from SC_i when the transitions in $\gamma_i, i \geq 0$ are taken.

Each node in the reachability tree describes a valid statechart configuration. We shall use the following notation for representing a configuration: parentheses denote OR-super-states; square brackets denote AND-super-states; and the values 1 and 0 represent active or inactive basic states, respectively. This notation preserves hierarchy and decomposition type, although only basic states are explicitly represented. The super-states are inferred from the configuration hierarchy. For the statechart shown in the Figure 1, the node description of the tree is in the form:

([([*Text*,*Animation*], *SubSection1.2*),*Hint*],[*Description*,(*Image*,*Audio*)])

As an illustration, Figure 7 gives the reachability tree for this statechart from the configuration SC_0 = {*Text*, *Animation*, *Hint*}. Such a tree can be used to verify whether there are any non-reachable pages. Given an application H, and an initial state configuration SC_0, a simple way to determine whether a particular page *ps* associated to

a state *s* is reachable during browsing is to compute the statechart's reachability tree and verify the occurrence of *s* in any of the generated configurations. If *s* does not occur, its

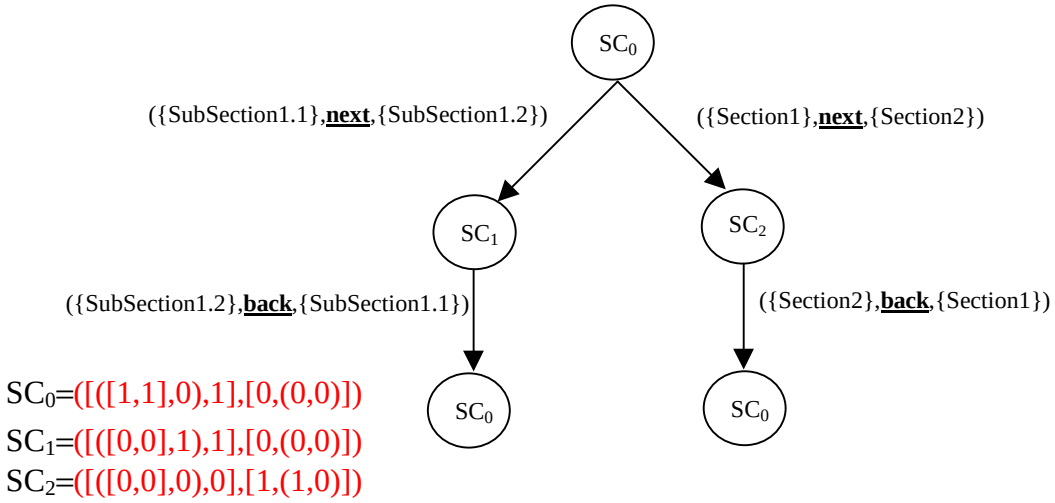


Fig. 7. Reachability tree for the statechart of Figure 1.

associated information cannot be viewed when the application is browsed starting from SC_0. Similarly, it is possible to determine whether certain groups of pages can be viewed simultaneously by looking for state configurations containing their associated states.

The reachability tree also enables the detection of configurations from which no other page can be reached because no links are enabled from them. These correspond to state configurations represented in leaf nodes of the reachability tree that are not duplicates of other non-leaf nodes. Another interesting characteristic to search for in the tree is the presence of cyclical browsing paths, whose occurrence indicates that the reader can return to a previously reached configuration during a browsing section. Documents that grow without careful organization, like Web pages, could be analyzed through the generation of a reachability tree from their models and have its anomalies detected. A somewhat similar use of statecharts and reachability trees is suggested by Fortes et al. [1996].

Yet another class of properties that can be checked for in a browsing session are temporal ordering relationships amongst the pages visited. From the statechart semantics, one can determine the set of all sequences of transitions that can be fired from a given initial configuration. We may use such information to anticipate sequences of link traversals, which may be taken by a reader from a given initial set of pages. Consequently, the semantics of the statechart can be analyzed to verify any browsing sequence restrictions imposed by authors. For example, a condition that any browsing session that reaches a page *py* must have previously reached page *px*, or a condition that

a browsing session which reaches a page *py* must not subsequently reach a page *px*, could both be checked for.

The reachability tree of the underlying statechart model also enables to determine, for example, the maximum number of simultaneous windows (or window frames, depending on the policy adopted at the user interface level) required for displaying the application. Since the model associates a page $p \in P$ to every state in set $S_s \subset S$, if the browser displays the contents of each concurrently viewed page in a frame, then the number of active states in the current statechart configuration equals the number of frames required to display that configuration. One can analyze the nodes in the reachability tree to find, for example, the maximum number of active states for all possible configurations. Such information can aid the determination of reasonable layouts for the display.

The size of the generated tree may be critical for statecharts with a great number of states, implying that applications with a great number of pages may require special treatment.

3.2 Synchronization of Simultaneous Display

Statecharts provide a computational model adequate for expressing concurrency. Therefore, they include suitable mechanisms for representing concurrent display of multiple pages and concurrent browsing paths within an application, as well as for providing authors with adequate support for specifying synchronization of concurrent activities.

Concurrent activity may appear in the form of several (presumably related) pages to be simultaneously displayed in a window, or in multiple windows, or yet via different media. For example, the activation of an anchor through the selection of a button may cause a browser to display a text frame, a related picture and also to play some audio. These pages essentially constitute a unit, and a single action from the reader should cause the simultaneous dismissal of all three pages from the display. This type of concurrent activity may be modeled by using parallel (AND) states, which are associated with the presentation of the multiple concurrent pages.

Consider our example application introduced in Section 2 (Figure 1 and Table I). Suppose the reader is the configuration depicted in Figure 4(a), given by pages {*TText,TAnimation,THint*}, and selects anchor 'nextSection2', thus generating the configuration {*TDescription,TImage*}. When the reader reaches this configuration she is simultaneously presented with a textual description and an image, contained in pages associated to the sub-states of the AND state *Section2* (*Description* and *Image*, which is

the default sub-state of *S2*). Following the statechart semantics, both pages associated to these sub-states will become visible after the transition associated to anchor 'nextSection2' is fired, and both will be removed from the display when the anchor 'backSection1' is selected, causing a return to the previous configuration. The statecharts' AND states thus provide a natural mechanism for specifying the parallelism inherent in the situation, as all the concurrent elements are explicitly depicted in the statechart.

3.3 Access Control

Boolean variables provide a simple yet effective mechanism to enforce browsing restrictions on readers, for example imposing access restrictions to different classes of users or defining navigational contexts [Schwabe et al. 1996]. HMBS ensures that access capabilities are an integral part of the delivered application. We illustrate this point using the example introduced in Section 2. Let us consider a report with assessment and frequency records for the employees taking the course, kept as a separate document. Access to these records is provided to two classes of readers: a privileged class (PR) formed by the instructors who can browse the entire document, and a restricted class (RR) formed by company managers who are not allowed to check the assessment information. A statechart modeling this scenario is presented in Figure 8. Suppose the pages associated to states *Assessments* and *GeneralComments* contain the privileged information. The remaining pages (associated to states *Description*, *Frequency* and *Final*) contain unrestricted information.

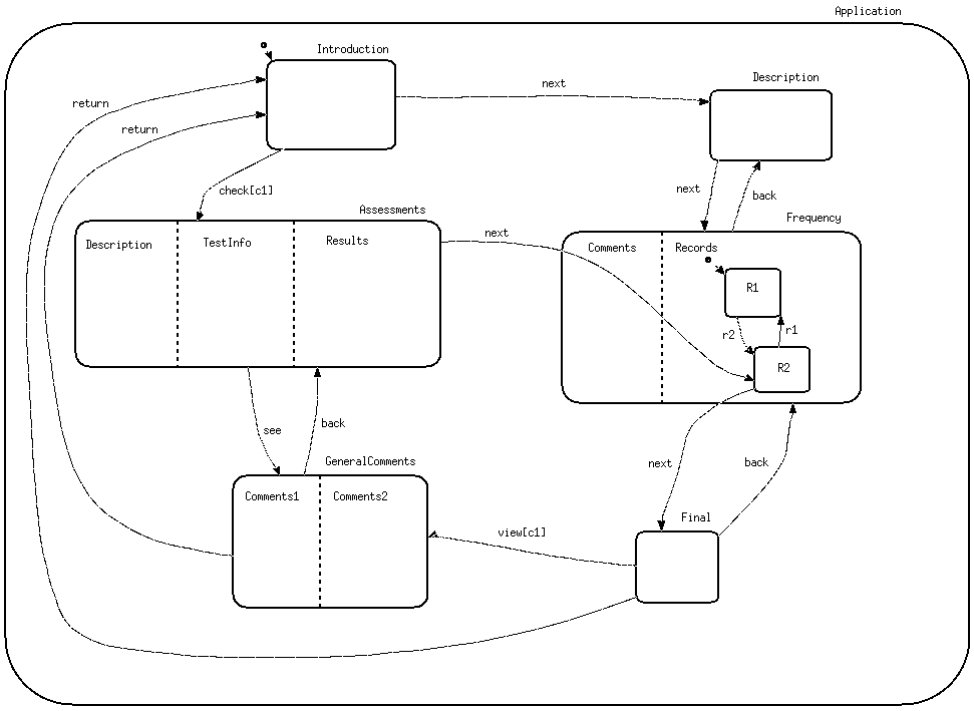


Fig. 8. Statechart model with access control through Boolean variables.

In the statechart, conditional transitions may govern access control to restricted nodes. According to the statechart semantics, if c_1 is a condition, a transition *check*[c_1] is fired only if event *check* is activated and c_1 evaluates to true. Thus, for a class PR reader, the initial setting is C = {c_1 = true}, therefore enabling access to the restricted pages. For a class RR reader, the initial setting is C = { c_1 = false}, so that this reader is not granted access to the pages associated to states *Assessment* and *GeneralComments* or to any links departing from them. A property analysis of such an application would require the generation of multiple reachability trees, one for each initial setting of C.

To construct applications with controlled access to pages, authors might adopt the following procedure: identify the relevant access control classes, deciding which pages are restricted for each class; associate a condition to each restricted class and a conditional transition to each state that corresponds to a restricted page; and finally define a set C for each control class.

Conditions can also be used to define navigational contexts. For example, suppose there are two alternative tests for learning assessment in our instructional material example. Both tests are displayed as guided tours containing multiple questions, some of which are shared. One test is targeted at beginners and the other one is targeted at experienced learners. Figure 9 depicts the modeling of such distinct tours controlled by conditions cn_1 and cn_2, respectively. In this case, an initial setting of C = {cn_1 = true, cn_2 = false} enables navigation through the *beginners* test, whereas V = {cn_1 = false, cn_2 = true} enables navigation through the *advanced* one. The links associated to transitions in each tour are enabled only if the corresponding variable evaluates to true.

3.4 Tailored Versions

A single statechart model may be used to keep a collection of distinct tailored versions of the same application. This may be achieved by delivering a single structural model with alternative mapping functions 'm' and 'pl', thus allowing the association of different pages to the states in set S_S. Such a solution assumes that only page contents are altered, while the underlying structure of the application is maintained. This is the case, for example, when versions of the same application in different languages are to be delivered.

If alternative structural organizations are required, the issues of representation and specification of appropriate alternative choices for content are the same discussed in the section on access control on browsing and the same solutions apply, although the reachability graph analysis may differ. Properties to be checked for in tailored versions

are, for example, that *exactly one* of a set of states is reachable (i.e., the page for only one of the versions is ever viewable); and that every state in a particular set is reachable

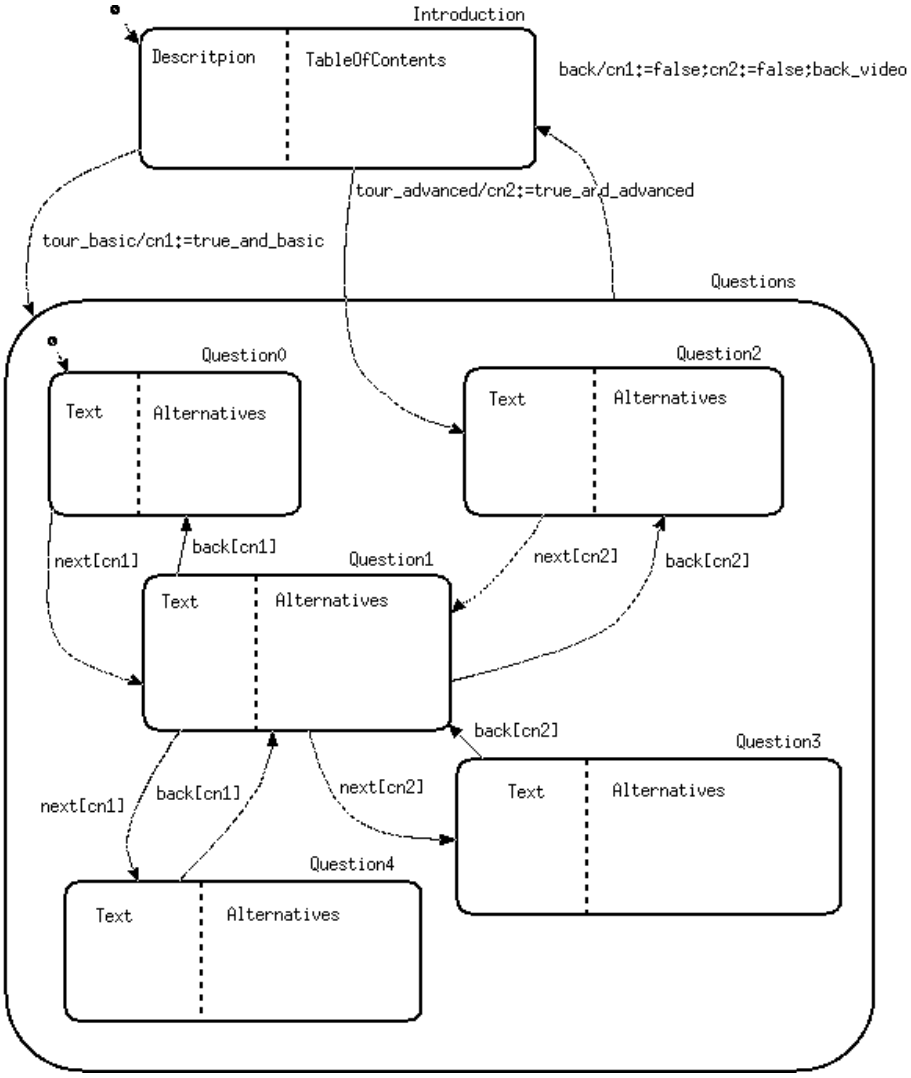


Fig. 9. Two navigational contexts specified using Boolean variables.

(i.e., all the pages of a particular version are potentially viewable).

3.5 Exploring Additional Statechart Features

Several properties of statecharts could be further explored in the context of hypermedia modeling. The history mechanism, for example, ensures that, if an OR state is entered, control is returned to the last active state, rather than to the default one. This facility could be explored for modeling situations where different user configurations lead to the activation of the same link, as it is common in on-line help applications. In our running

example, a similar situation occurs if the reader is allowed to consult the definition of a particular term whenever it appears. If the pages containing such a link are mapped into states grouped in a super-state and the link is mapped into a history transition, the history mechanism ensures that the reader will be returned to the particular page where the link was activated whenever she returns to the super-state. This facility allows a considerable reduction on the number of transitions required by a model specification.

Actions associated to transitions allow assignment of variables during a browsing session, an interesting property to be explored in applications such as programmed learning and course evaluation. Variables can register user choices during one or more browsing sessions, or may register grades assigned according to their answers to proposed questions. The application could then select the most adequate paths for a particular user by inferring her knowledge on the subject from the information registered. The broadcast mechanism can provide proper synchronization support for multimedia applications, as actions may generate events, which may be captured by orthogonal components. As an example, a video that ends before its soundtrack may broadcast a signal to stop the sound and proceed to other pieces of the presentation. Such a use of statecharts is described in [Paulo et al. 1997; 1998; 1999].

3.6 Integration of HMBS into a Design Method

HMBS may be integrated into a higher-level design method, providing a systematic procedure for the construction of hypermedia applications with the support of a mathematical formalism.

An HMBS model provides a structural description of a hypermedia application, where each state corresponds to an atomic item of information. Before creating the application, however, its authors must go through a design process, which usually starts with a conceptual model of the application domain [Schwabe et al. 1996, Isakowitz et al. 1995]. From this initial model, design proceeds to the construction of a general navigational model that provides a navigation-oriented view of the application expressed in terms of access primitives and information units. The navigational model provides the roadmap for the construction of the HMBS instantiated model. Well-defined rules may be devised to guide the process of transforming conceptual access structures from the navigational model into concrete statechart constructions in the HMBS model. Designers must describe the overall structure of the application in terms of states and transitions (pages and links), and then proceed to associate content to pages. Through its presentation channels, the model supports the definition of some interface properties at this stage of the development (e.g., font size and type for a text page, volume for an audio page, etc.).

However, HMBS is not a model for representing interface states and an interface definition stage is required in the design process, although some decisions usually taken at that stage of the design may be anticipated once the model is adopted.

If an HMBS model is input into a system that implements a statechart machine, it may be simulated and evaluated to check for properties before design proceeds. An environment with such functionality is described in [Turine et al. 1999]. Further design stages such as interface definition and implementation must be carried out if implementation into a commercially available platform is desired. In this case, approaches for translating HMBS models into a description suitable for a proprietary system or translation into a WWW publication language might be considered.

4. RELATED WORK

The model implicit in many hypertext systems is the *labeled directed graph*, where the information content of the application is associated with the contents of a set of nodes, and nodes marked with tokens are displayed. Although extremely simple, the model does not adequately represent the structure of the data and the browsing semantics, and provides no separation between the structural organization of the nodes and their contents.

Tompa [1989] proposes a *hypergraph* formalism to model generic hypertext structures that enables formal identification of commonalities in these structures (nodes, links, labels) and direct reference to “groups of nodes” having a common link semantics (*hyperedges)*. This model facilitates the separation of structure from content, includes set oriented browsing semantics, and incorporates arbitrarily many layers of personalized and system structures. A dynamic behavior is also specified through the notion of node marking. Although the model provides a natural representation for the overall structure of a document and for decomposing it in fragments, its use of token marking to model the browsing semantics is not satisfactory. This is mainly because the approach of token marking is too general, as there are no restrictions on the marking process and the hypergraphs behave as general finite state machines. Moreover, the graphical representation associated to hypergraphs does not clarify their dynamic behavior.

Stotts and Furuta [1989] proposed the Trellis model, based on *Petri nets*. This model essentially provides a unifying formalism for describing and reasoning about many of the features of existing hypertext systems. It uses the Petri net structure and execution semantics not only to represent the structure but also to specify application browsing semantics. The Trellis model provides elegant support for solving several problems inherent in hypertext systems, such as analysis of display complexity and node

reachability, providing concurrent browsing paths and synchronization access control, and tailored versions of applications.

However, a major shortcoming of the Petri net based model is the lack of satisfactory hierarchical structuring facilities, which makes difficulty the task of specifying synchronization control across different levels of hierarchical structures. Only one level of hierarchical decomposition can be accommodated in a Petri net, as a hierarchical state is not another Petri net in a lower level of decomposition, but rather a set of multiple sub-states which must be either interpreted as a concurrent state or as a sequential state. On the other hand, statecharts allow the decomposition of states into sub-states that are also statecharts, and behave as such. Whereas with Petri nets successive decomposition of states is not easily accommodated and no mechanisms are provided for specifying synchronization within such state decompositions, in statecharts any level of structural decomposition is easily represented and has an associated semantics.

According to Zheng and Pong [1992], the behavior associated to the “token marking” of places is not easily visible from the static graphical representation of Petri nets, and therefore it is not intuitive to specify or to understand the browsing semantics of applications using this concept. The execution semantics of the statecharts, on the other hand, does not use tokens, and can be easily captured from its graphical representation. Therefore, we believe the HBMS model provides a more intuitive model for describing browsing semantics.

HDM [Garzotto et al. 1993], on the other hand, is a model oriented to the description of hypermedia applications. It shares with the Trellis and Tompa's models the idea of abstracting the structure of the nodes from its contents. However, while these latter models are more concerned with behavioral aspects of hypermedia, or with operational features, HDM, as a design model, is concerned with the structural organization of applications and with representational issues. Although this concern with structure is also present in HMBS, the latter is not a design model: it provides mechanisms for describing the application’s structure, but authors need the additional support of a design method to derive a proper structure.

In our view, a major advantage of using a statechart-based specification model is that it provides a mathematical model with associated semantics and algorithms, yet it has an easy and intuitive visual notation associated. It also has the advantage of being a very intuitive model, as statecharts are an extension of finite-state machines, and modeling based on states and events is well established within the computer science community. Formal models may provide a standpoint for encouraging application portability. Moreover, as statecharts were designed to model concurrent reactive systems, HMBS is

suitable for describing concurrency aspects inherently associated to the hypermedia navigation.

OOHDM [Schwabe et al. 1996], RMM [Isakowitz et al. 1995, Díaz and Isakowitz 1995] and EORM [Lange 1994] are more recent examples of the class of application design methods. These methods follow the overall steps for a design approach comprising conceptual modeling of the application, navigational and interface modeling and a final implementation stage followed by evaluation and testing. Such a basic framework has been proposed in [Nanard and Nanard 1995], and is being adopted widely. A different use of statecharts is proposed in OOHDM, which employs *ADVCharts* [Carneiro et al. 1994]. ADVCharts are a generalization of statecharts [Harel et al. 1997] and ObjectCharts [Coleman et al. 1992] targeted at the specification of interface behavior during the user interface design stage. ADVCharts support the specification of the navigational transformations that occur at the user interface and their impact on the navigational objects.

As it has been argued in Section 3.5, the proposed model may be integrated into an application design method, providing a formalism suitable for describing navigational models that explore the hierarchical structure of the information by providing structure-oriented information. The association of a rigorous formalism with an application-oriented model may bring considerable benefits for designers. A formal model provides coherent semantic interpretation for structural decomposition and browsing semantics. Algorithms associated with the formalism may be used to support the generation of correct and non-ambiguous specifications. Moreover, they provide valuable tools that may be used for analyzing applications, supporting the evaluation of certain quality aspects, such as structuring and faulty or redundant browsing paths. They also may simplify the processes of automating access control, generating alternative browsing paths and providing structured navigation.

5. CONCLUSIONS

The HMBS model uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of a hypermedia application. Statecharts are a well-known extension to conventional state-transition diagrams based on finite state machines whose outstanding features are hierarchy of states, ability to specify parallelism and a communication mechanism via broadcasting. They also feature a rich set of special notations for augmenting the notational power of state-transition diagrams, thus allowing the specification of complex problems in very concise diagrams.

The model is particularly suitable for modeling applications with a hierarchical structure, such as books, scientific papers, on-line tutorials and manuals, instructional material, etc. This is so because the state hierarchy imposed on the underlying statechart model provides adequate mechanisms for modeling hierarchical organization and for exploring possible browsing actions to be taken [Turine et al. 1997]. As such, it is adequate for generating applications that support "structured navigation", e.g. in applications oriented towards the teaching/learning process where the reader is browsing to get knowledge about a certain topic or subject ("reading for comprehension", according to Thüring et al [1995]), rather than with the goal of locating information. Moreover, as a statechart-based model, it is associated to rich mathematical artillery that may be explored to improve application quality. However, an obvious overhead of using an HMBS-based system is that the author interacting with such a system must be familiar with the model and its underlying formalism, unless they can be hidden within a more "friendly" metaphor.

We constructed a prototype hypermedia system called *HySCharts* that incorporates an authoring and a browsing modes [Turine et al. 1999]. It reuses the graphical statechart editor and simulator of the STATSIM statechart simulation environment [Masiero et al. 1991]. In the authoring mode authors may specify HMBS models and verify them through programmed execution of the underlying statechart. In the browsing mode, readers may browse applications according to the browsing semantics defined for the model. *HySCharts* could be extended to incorporate in its authoring mode a "statechart engine" capable of supporting automatic authoring for certain classes of documents that present a fixed structure. In that case, rather than modeling the structural organization of a single application, a model would be shared by different instances of the same document type. Such use of statecharts is actually suggested by Masiero et al. [1994a], who propose a method for analyzing office applications which relies on a model based on statecharts to record the flow of documents within the system. Each type of document is associated with a statechart, and that enables the automatic update of the links maintained in a hypermedia database when new documents are added.

The model as presented here does not provide mechanisms for specifying the synchronization requirements typical of hypermedia applications. Paulo et al. [1997;1998;1999] propose an extension that caters for such requisites, including the specification of the temporal sequencing and synchronization restrictions inherent to the simultaneous presentation of dynamic data streams.

ACKNOWLEDGMENTS

The authors acknowledge the financial support of CNPq and FAPESP. We are also very grateful to the anonymous reviewers from TOIS for their helpful comments on a previous version of this article.

REFERENCES

Balasubramanian, V., Bashian, A. and Porcher, D. 1997. A large-scale hypermedia application using document management and web technologies. In: *Proc.* ACM *Hypertext'97*, Southampton, UK, April 6-11, 134-145.

Bieber, M. and Vitali, F. 1997. Toward support for hypermedia on the world wide web. *IEEE Computer* (January), 62-70.

Brown, P.J. 1987. Turning ideas into products: the Guide System. In: *Proc. ACM Hypertext´87*, Chapel Hill, N.C., USA, 33-40.

Carneiro, L.M.F., Coffin, M.H., Cowan, D.D. and Lucena, C.J.P. 1994. ADVcharts: a visual formalism for highly interactive systems. In: Harrison, M.D., Johnson, C. (eds.): *Software Engineering in Human-Computer Interaction*, Cambridge University Press.

Coleman, D., Hayes, F. and Bear, S. 1992. Introducing object-charts or how to use statecharts in object-oriented design. *IEEE Transactions on Software Engineering*, 18(1), (January), 9-18.

Díaz, A. and Isakowitz. T. 1995. RMCase: computer-aided support for hypermedia design and development. In*: Proc. Int. Workshop on Hypermedia Design 1995*, F. Garzotto (Ed.), June, Montpellier, France, Springer Verlag 15p.

Drusinsky, D. and Harel, D. 1988. Using statecharts for hardware description and synthesis. *IEEE Transactions on Computer-Aided Design* 8, 7 (July), 798-806.

Fortes, R.P.M., Garcia Neto, A. and Nicoletti, M.C. 1996. A formal approach to consistency and reuse of links in hypermedia applications. In: Proc. *Workshop on Formal Methods in Human Computer Interaction: Comparison, Benefits, Open Questions*, ACM CHI'96, Vacouver, Canada, April 14-15, pp. 33-36.

Garg, P.K. 1988. Abstraction mechanisms in hypertext. *Communications of the ACM* 31, 7, 862-870.

Garzotto, F., Paolini, P. and Schwabe, D. 1993. HDM — a model-based approach to hypertext application design. *ACM Transactions on Information Systems* 11, 1 (January), 1-26.

Halasz, F. and Schwartz, M. 1994. The Dexter hypertext reference model. *Communications of the ACM* 37, 2 (February), 51-62.

Harel, D. 1987a. On the formal semantics of statecharts. In: *Proc. of the 2nd IEEE Symposium on Logic in Computer Science*. Ithaca, New York, 54-64.

Harel, D. 1987b. On the formal semantics of statecharts. In: *Proc. II IEEE Symposium on Logic in Computer Science*. Ithaca, New York, 54-64.

Harel, D. and Naamad, A. 1996. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology* 5, 4 (October), 293-333.

Harel, D. and Gery, E. 1997. Executable object modeling with Statecharts. *IEEE Computer* 30, 7 (July), p.31-42.

Isakowitz, T., Stohr, E.A. and Balasubramanian, P. 1995. RMM: a methodology for structured hypermedia design. *Communications of the ACM* 38, 8 (August), 34-44.

Isakowitz, T., Kamis, A. and Koufaris, M. 1997. Extending the capabilities of RMM: Russian Dolls and Hypertext. In: *Proc. XXX Annual Hawaii Int. Conf. on System Sciences*, (HICSS'97).

Lange, D.B. 1994. An object-oriented design method for hypermedia information systems. In: *Proc. XXVII Annual Hawaii Int. Conf. on System Sciences*, Maui, Hawaii, January 4-7, v.III, 366-375.

Marshall, C.C. and Irish, P.M. 1989. Guided-tours and on-line presentations: how authors make existing hypertext intelligible for readers. In: *Proc. ACM Hypertext'89*, November 1989, 15-26.

Masiero, P.C., de Oliveira, M.C.F., Germano, F.S.R. and Santos, G.P.B. 1994a. authoring and searching in dynamically growing hypertext databases. *Hypermedia Journal* 6, 2, 124-148.

Masiero, P.C., Maldonado, J.C. and Boaventura, I.G. 1994b. A reachability tree for statecharts and analysis of some properties. *Informantion and Software Technology* 36, 10, 615-624.

Nanard, J. and Nanard, M. 1995. Hypertext design environments and the hypertext design process. *Communications of the ACM* 38, 8 (August), 49-56.

Paulo, F.B., Masiero, P.C. and de Oliveira, M.C.F. 1997. Hypercharts: extended statecharts to support hypermedia specification. In: *Proc. III IEEE Int. Conf. on Engineering of Complex Computer Systems ICECCS'97,* Como, Italy, (September), 1997, 152-161.

Paulo, F.B., Turine, M.A.S., de Oliveira, M.C.F. and Masiero, P.C. 1998. XHMBS: a formal model to support hypermedia apecification. In: *Proc. ACM Hypertext'98,* Pittsburg, PA, USA, June 20-24, 161-170.

Paulo, F.B., Masiero, P.C. and de Oliveira, M.C.F. 1999. Hypercharts: an extension of statecharts to support hypermedia specification. *IEEE Transactions on Software Engineering*, 25(1) (January/February), 33-49.

Rossi, G., Schwabe, D. and Garrido, A. 1997. Design reuse in hypermedia applications development. In: *Proc. ACM Hypertext'97*, Southampton, UK, April 6-11, 57-66.

Schwabe, D., Rossi, G., Barbosa, S.D.J. 1996. Systematic hypermedia application design with OOHDM. In: *Proc*. ACM *Hypertext'96*, Washington DC, USA, March, 116-128.

Stotts, P.D. and Furuta, R. 1989. Petri-net-based hypertext: document strucuture with browsing semantics. *ACM Transactions on Information Systems* 7, 1 (January), 3-29.

Stotts, P.D. and Furuta, R. 1992. Hyperdocuments as automata: trace-based browsing property verification. *Proc. ACM ECHT, European Conference on Hypertext Technology*, November 30 - December 4, 272-281.

Thüring, M., Hannemann, J. and Haake, J.M. 1995. Hypermedia and cognition: designing for comprehension. *Communications of the Association of Computer Machinery* 38, 8 (August), 57-66.

Tompa, F.W. 1989. A data model for flexible hypertext database systems. *ACM Transactions on Information Systems* 7, 1 (January.), 85-100.

Turine, M.A.S., de Oliveira, M.C.F. and Masiero, P.C. 1997. Designing structured hypertext with HMBS. In: *Proc. ACM Hypertext'97,* Southampton, UK, April 6-11, 1997, 241-256.

Turine, M.A.S., de Oliveira, M.C.F. and Masiero, P.C. 1999. HySCharts: a hyperdocument authoring and browsing environment based on statecharts. To appear in *Multimedia Tools and Applications*, Kluwer Academic Publishers.

Zheng, Y. and Pong, M. 1992. Using statecharts to model hypertext. In: *Proc. ACM EHCT, European Conference on Hypertext Technology*. Milano, 242-250.