UNIVERSIDADE DE SÃO PAULO

A statechart - based model for hypertext

MARIA CRISTINA FERREIRA DE OLIVEIRA
MARCELO AUGUSTO SANTOS TURINE
PAULO CESAR MASIERO

Nº 19

NOTAS

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2577

A statechart - based model for hypertext

MARIA CRISTINA FERREIRA DE OLIVEIRA
MARCELO AUGUSTO SANTOS TURINE
PAULO CESAR MASIERO

Nº 19

NOTAS DO ICMSC
Série Computação

São Carlos
Out./1995

Resumo

Este trabalho apresenta o HMBS - Modelo para Hiperdocumentos Baseado em Statecharts. Este modelo usa a estrutura e a semântica de execução dos statecharts para especificar tanto a organização estrutural como a semântica de navegação de um hiperdocumento. Os statecharts são uma extensão de máquinas de estado finito e, consequentemente, o modelo é na verdade uma generalização dos modelos para hipertexto baseados em hipergrafos.

Características importantes do modelo são a sua habilidade para modelar hierarquia de informação e sincronização da navegação simultânea por caminhos distintos do hiperdocumento; a existência de mecanismos para especificar controle de acesso e criar múltiplas versões personalizadas de um hiperdocumento; a possibilidade de gerar visões hierárquicas e de permitir acesso indexado. O statechart associado ao hiperdocumento pode ser analisado para verificar propriedades como alcançabilidade de nós e validade de caminhos. Alguns exemplos ilustrativos da aplicação do HMBS serão apresentados.

A STATECHART - BASED MODEL FOR HYPERTEXT

Maria Cristina Ferreira de Oliveira
Marcelo Augusto Santos Turine
Paulo Cesar Masiero
E-mail: {ferreira/ mast/ masiero}@icmsc.sc.usp.br

Departamento de Ciências da Computação e Estatística
Instituto de Ciências Matemáticas de São Carlos
Universidade de São Paulo
C.P. 668, 13560-970 — São Carlos, SP
Brazil

Abstract

We present a formal definition for HMBS - the Hypertext Model Based on Statecharts. HMBS uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of a hypertext. 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 of information and synchronization of simultaneous display windows within a hypertext; provision of mechanisms for specifying access control, multiple taylored versions, hierarchical views and indexed access. Analysis of the underlying machine allows verification of node reachability, valid paths and other properties. Examples of usage of HMBS are also provided.

1. Introduction

A major challenge facing the author of a hypertext[1] is that of organizing complex material in a suitable way. A systematic approach to defining the structural organization is especially important in the design of large and complex hypertexts. In this case the use of a model helps to discipline the authoring activity by encouraging the development in a structured fashion, so that the structure is designed before the actual contents are filled into the document nodes. The peculiarities of hypertexts (e.g., the role of links, the complexity of structures, the multimedia facilities, the navigation/browsing paradigm,

[1]In this article the term *hypertext* is used to denote online documents made up of a network of interconnected pieces of information (nodes). The term *hypertext system* denotes software tools used to create and browse *hypertexts*. HyperCard, NoteCards, KMS, Intermedia and Guide are examples of hypertext systems. Note that a single *hypertext* might be published in several editions, each using a different hypertext system.

etc.) are encouraging the development of brand new models [Con87]. A number of formal models for describing the design process, the structural organization and/or the browsing semantics of hypertexts have been proposed in the literature.

Garzotto et al. [Gar93] identify four distinct approaches to hypertext modeling. Namely, there are "*application-oriented*" models which explicitly model the semantics of specific application domains, such as the model employed in the g-IBIS hypertext tool [Con88,Rei91]. A second class includes models that are "*system-oriented*", rather than application-oriented. These include, for example, the Dexter Hypertext Reference Model [Hal94], and Garg's set-theoretical model [Gar88]. Such models try to identify the relevant abstractions found in a wide range of existing (and future) systems. A third class includes the "*behavioral methods*", such as the Trellis model [Sto91,Sto92] and Tompa's model [Tom89]. These are concerned with the modeling of the dynamic behaviour associated to hypertext networks and with the browsing semantics allowed for navigating through the net. A fourth class comprises less formal approaches, which emphasize preferred topological structures as building blocks to create the structure of hypertext networks. Examples of approaches in this class are the linear strucutures used in Hypercard [App94] and Guide [Bro87], and the hierarchical structures of KMS [Aks88].

The model proposed in this paper, *HMBS* (*Hypertext Model Based on Statecharts*), uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of a hypertext. In the context of the aforementioned classification, the HBMS model may be included in the class of behavioral methods, due to its ability to model the browsing semantics of hypertexts. However, it is also adequate for describing the document's organization, particularly for describing the hierarchical structure common to many hypertexts, for the hierarchy levels are directly mapped into the different levels of an underlying statechart model. Moreover, as statecharts were designed to model concurrent reactive systems, the HMBS model is suitable for describing concurrency aspects of a hypertext. It also has the advantage of being a very intuitive model, as statecharts are an extension of finite-state machines.

Zheng and Pong [Zhe92] also use *statecharts* for hypertext modeling, but they are concerned with the specification of management aspects of the hypertext system interface, rather than with the modeling of the hypertext structure and contents. That is, they are concerned essentially with the specification of the browsing semantics of hypertext systems, and not with the structural organization of the hypertexts within the system. They exemplify the use of statecharts to model the behaviour of various buttons and frames supported by Guide, a real-life production hypertext system. We consider their approach a potentially interesting use of statecharts, but in this paper we suggest the use of statecharts in a different context, to model the structural organization and the browsing semantics associated to hypertexts.

The remainder of the article is divided as follows. Section 2 describes the formal syntax used in statecharts, and the main features of the proposed model with respect to its use for specifying the structural organization and the browsing semantics of hypertexts. It also presents a definition of hierarchical views for hypertexts. Section 3 describe possible simple solutions warranted by the model to some common problems related to hypertext visualization, browsing and analysis. Section 4 discusses related work by comparing HMBS to other approaches. Section 5 presents a discussion and implications of using the HMBS model in the modeling of hypertexts. Finally, the conclusions are presented in Section 6.

2. A Statechart-Based Model

Statecharts constitute a visual formalism for describing states and transitions in a modular fashion. The statechart formalism 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, agregation and refinement of states [Har87a].

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 interface, of a hypertext 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 new information.

The *HMBS* (*Hypertext Model Based on Statecharts*) model uses the structure and execution semantics of statecharts to specify both the contents (including the linked structure) and the browsing semantics of a hypertext. This logical structure provides a layer of indirection which can be interpreted to generate a final product to be delivered to users. As in other models (e.g., Trellis [Sto89]), hypertext contents and linked structure, as well as the logical structure of the document and its mapping to specific physical representations, are effectively separated. Therefore, a single model representation may be used to generate different versions, or different presentations, of the same hypertext.

2.1. Background on Statecharts

In this section we describe the formal syntax for statecharts. It should be noted that the syntax described is actually a subset of the one presented by Harel in [Har87b], as not all of the aspects of the original definition are relevant for the problem under consideration here.

Definition 1: A Statechart structure (ST) is a 8-tuple,

$$ST = <S, \rho, \psi, \delta, V, C, E, T>$$

in which:

- $\mathbf{S} = \{s_1, s_2, \ldots, s_n\}$ is the *set of states*, $n > 0$.
- ρ: $S \rightarrow 2^S$ is a *hierarchy function* that defines the substates 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 $\rho(s) = \phi$. ρ^*, ρ^+ are extensions of ρ defined by $\rho^*(s) = \cup \rho^i(s)$, $i \geq 0$; $\rho^+(s) = \cup \rho^i(s)$, $i \geq 1$.
- ψ:$S \rightarrow \{AND, OR\}$ is a function that defines the type of each state $s \in S$ such that $\rho(s) \neq \phi$ (s is not a basic state).
- δ: $S \rightarrow 2^S$, is the *default function*, defining the set of initial states which are contained in a state s. Thus, $\delta(s)$ is the *default set* for s. For $x \in S$, if $x \in \delta(s)$ then $x \in \rho^+(s)$.
- **V** is the *set of expressions* containing variables names from a set of logical variables V_p with known initial values. The set $V = V_p$ is thus given by $V_T \cup V_F$, in which V_T is the set of (initially) *true* variables and V_F is the set of (initially) *false* variables.
- **C** is the *set of conditions*, which may be T (true), F (false) or logical expressions of the type:

  If $u, v \in V$ then $u \vee v$, $u \wedge v$, $\neg u \in C$.

- E is the *set of event expressions*, also called *labels*, which may be contained in the *set of primitive events* E_p, or may be combined with conditions:

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

- $T \subset 2^S \times E \times 2^S$ is the *set of transitions*. A transition $t = (X, e, Y)$ is composed of a source set of states X, a target set of states Y and a label $e \in E$.

A *statechart legal configuration* (SC) is defined by Harel in [Har92] as:

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 $\psi(LCA(\{x,y\})) = AND$. A set of states is orthogonal if its states are pairwise orthogonal.

A *maximal orthogonal* set is an 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. Its set of states is given by S = $\{A,A_1,A_2,B,B_1,B_{11},B_{12},B_2,B_{21},B_{22},C,D,E\}$, where A is the root state, and $\{B_{11},B_{12},B_{21},B_{22},C,D,E\}$ are the basic states. The hierarchy functions for states A and A_1 are given by $\rho(A)=\{A_1,A_2\}$, $\rho(A_1)=\{C,D\}$. The ψ functions for the same states are $\psi(A)$ = {AND}, $\psi(A_1)$ = {OR}. The default functions for these states are $\delta(A)$ = $\{B,C,B_{11},B_{21}\}$, $\delta(A_1)$ = {C}, whereas for state A_2 it is given by $\delta(A_2)$ = $\{B,B_{11},B_{21}\}$. The set V is given by $V = V_T \cup V_F$, where $V_T = \{v_1\}$, $V_F = \varnothing$; in other words, variable v_1 is assumed true for the current application. The set of event expressions is E = $\{e_1[v_1],e_2,e_3,e_4,e_5,e_6\}$, and the set of transitions is T = $\{(\{C\},e_1[v_1 = \textit{true}],\{D\})$; $(\{D\},e_2,\{C\})$; $(\{B\},e_3,\{E\})$; $(\{E\},e_4,\{B_{12},B_{22}\})$; $(\{B_{11}\},e_5,\{B_{12}\})$; $(\{B_{12}\},e_6,\{B_{22}\})\}$. Possible configurations for this statechart are $SC_1 = \{B_{11},B_{21},C\}$; $SC_2 = \{B_{11},B_{21},D\}$; SC_3 = {D,E}, etc.

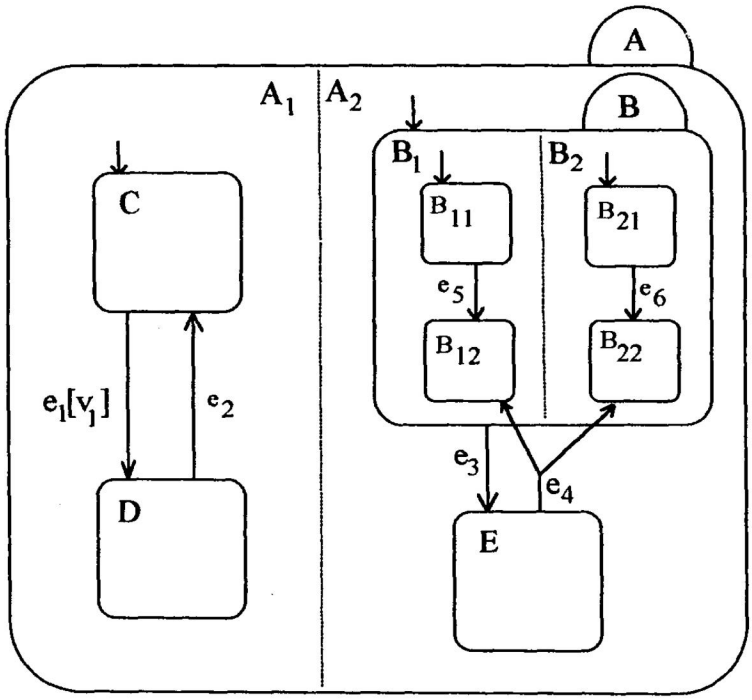


Figure 1: A Statechart example.

2.2. Definition of the Model

The HMBS model provides a mechanism for effectively separating the hypertext physical and logical structures. The model uses the structure and execution semantics of statecharts to especify both the linked (logical) structure and the browsing semantics of a hypertext. The model definition is given below:

Definition 2: A Hypertext H is a 6-tuple $H = \langle ST,P,R,M,L,V \rangle$ in which

- **ST** = <S,ρ,ψ,δ,V,C,E,T> is a statechart structure.
- **P** is a *set of pages* (document contents) corresponding to the data objects in the document. In this text we are assuming that elements in set P contain text, buth they might as well contain graphics, tables, bitmaps, executable code, audio information, or any other data objects supported by the hypertext system. The set P also includes a special *null page*, which has no associated contents. Each element in set P is associated to two pieces of information: *page-title* and *page-contents*. The first provides a unique identifier to the page, whereas the second describes its content information.
- **R** is a *set of readers* for interpreting hypertext pages. A reader is an interpreter for a page, operating on the page as a whole and typically displaying the page through some medium (e.g., text formatters, graphic decoders, program interpreters, audio and video players, data manipulation systems, etc.).
- **M:** $\mathbf{S_S \rightarrow P}$ is a *value function* mapping states into data objects (in the form of a page). Possibles mappings are defined for states in a set S_S such that

$$S_S: \{x \in S \mid \psi(x) = OR \vee \rho(x) = \phi\}$$

S_S is the subset of S comprising basic states and OR states. AND states are not mapped into data objects.

- **L**, the hypertext browsing level, is the visibility level ($0 \leq L \leq n$) used for defining the hierarchy depth when displaying pages during navigation (see Section 2.3 below).
- **V:** $\mathbf{P \rightarrow R}$ is the *vizualization relationship* which associates each hypertext page with a single *reader* that is able to interpret it.

2.3. Browsing Semantics

The browsing semantics of a hypertext is the manner in which information is to be visited and presented to a reader who is navigating through the document. In this section some examples are provided to illustrate the power and the flexibility of the statechart model to represent the browsing semantics of hypertexts.

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

The nodes of the hypertext are mapped into a set of statechart states, and the set of possible link activations are represented by the set of events and transitions allowed in the statechart. Event activation in the statechart is thus used to represent anchor (or link) activation in the hypertext. Physically, such anchors may be indicated through buttons,

menus or marked keywords within the hypertext. Thus clicking a button in the hypertext results in the triggering of an event in the underlying statechart model, consequently activating all links (transitions in the statechart) from the source node (i.e., its corresponding state in the statechart) to the destination node (state).

Within the statechart model, users are positioned at an arbitrary number of states at a time, having concurrent acess to an arbitrary number of data objects. The current statechart configuration defines a *user state* in the hypertext system, that gives the set of currently active hypertext nodes at a certain stage during browsing, indicating which page contents are displayed for viewing. Nodes, or states of the set S_S, are mapped into pages of the set P according to the given function M. A possible mapping of pages to nodes for the statechart described in Figure 1 is defined in Table 1.

Table 1: Value function (M) mapping states into pages for the statechart depicted in Figure 1.

States (S_S)	**Pages**	**Titles**
A_1	PA_1	TA_1
A_2	PA_2	TA_2
B_1	PB_1	TB_1
B_{11}	PB_{11}	TB_{11}
B_{12}	PB_{12}	TB_{12}
B_2	PB_2	TB_2
B_{21}	PB_{21}	TB_{21}
B_{22}	PB_{22}	TB_{22}
C	PC	TC
D	PD	TD
E	PE	TE

To view a page, it is necessary to invoke an associated reader that interprets and exhibits it through some medium. To visualize the pages associated to the current hypertext configuration readers must be invoked for every state in the underlying statechart's current state configuration. Pages associated to ancestral states may also be viewed in conjunction with those associated to currently active states, a facility which is useful for allowing users to simultaneously visualize different levels of the hypertext hierarchy as mapped into the statechart. The hierarchy level to be displayed is specified through the visibility level L, as described below.

If L = 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 L = 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 L = 2 *then*

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

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

Generalizing,

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

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

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

• generates an event which causes the triggering of the transition associated to that link. The source set for the transition must be an active set, that is, it must be included in the current state configuration.

• activates all states which are target sets for the transitions fired, thus generating the next state configuration and disabling the previous current state configuration.

• invokes the readers for the pages associated with those nodes corresponding to states in the new configuration and those within the scope of the visibility level L.

Assuming an initial state configuration for the statechart model shown in Figure 1 as given in Figure 2, where $SC_0 = \{B_{11}, B_{21}, C\}$, and that the set of variables $V = V_T \cup V_F$ is given by $V_T = \{v_1\}$, $V_F = \phi$, let's illustrate possible state configurations to be reached during browsing, and possible layouts for the associated display windows, according to the interpretation given to the statechart semantics.

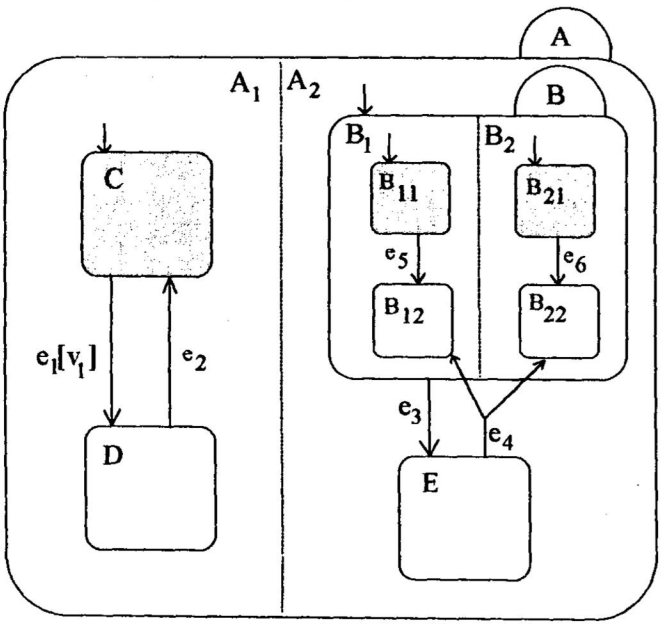


Figure 2: Possible initial configuration for the statechart model in Figure 1, $SC_0 = \{B_{11}, B_{21}, C\}$.

If a visibility level L = 0 is specified, a schematic layout for the window displaying the hypertext at this stage in browsing is presented in Figure 3. The pages associated to the three nodes in the current state configuration are displayed, and the anchors defined for each node being displayed are also shown.

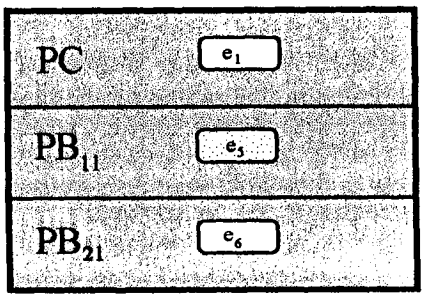


Figure 3: Schematic window layout displayed during the browsing of the hypertext when $SC_0 = \{B_{11}, B_{21}, C\}$, considering L = 0.

For example, clicking at the anchor labeled e_5 in page PB_{11} corresponds to activating the event labeled e_5 in the underlying statechart, leading to a new state configuration $SC_1 = \{B_{12}, B_{21}, C\}$ and a new layout for the display, which is ilustrated in Figure 4. As L = 0, only the pages associated to basic nodes in the current state configuration, and only the anchors associated to transitions enabled for such states, are shown.

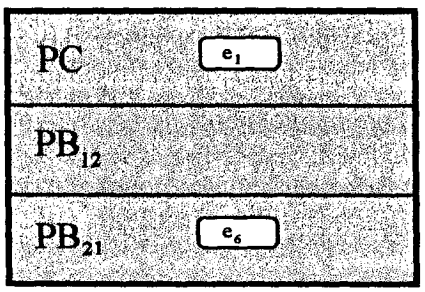


Figure 4: A possible window layout for $SC_1 = \{B_{12}, B_{21}, C\}$ and L = 0.

If L = 1 and the current state configuration is as indicated in Figure 2, a possible layout for the display window is shown in Figure 5. In this situation, the visibility level is set in order to shown not only the pages associated to the basic states in the current configuration, but also the pages associated to their parent states which are in set S_S. Thus, the hypertext hierarchical node structure up to a depth of one is displayed, as well as the anchors in each page which correspond to enabled transitions in the underlying statechart.

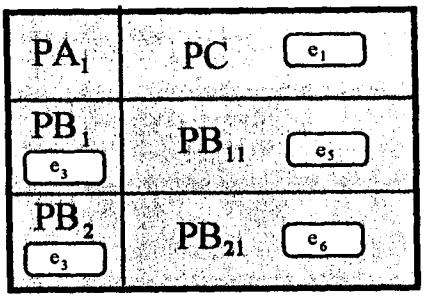


Figure 5: Window layout for $SC_0 = \{B_{11},B_{21},C\}$ and L = 1.

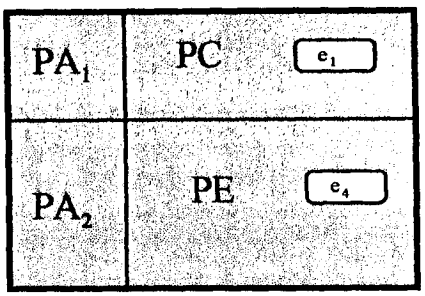


Figure 6: Window layout for $SC_2 = \{C,E\}$ and L= 1.

Observe that anchor e_3 corresponds to event e_3 in the statechart model, which is associated to state B. Because this is an AND state, there is no page associated to its corresponding node, and therefore anchor e_3 is actually shown in the pages associated to nodes (states) B_1 and B_2, which are the substates of state B. Selecting anchor e_3 in either page PB_1 or PB_2 results in leaving both pages simultaneously, and produces a new statechart configuration $SC_2 = \{C,E\}$ for which a possible display window is depicted in Figure 6.

2.4. Hierarchical Views

The hierarchy of states in a given statechart may be described by an AND/OR tree with depth[2] *d*. States can be repeatedly decomposed either into OR-components (which are, in fact, XOR-substates) or into AND-components, and the source and target states of a transition are allowed to reside on any level of the tree. The AND/OR tree for the statechart in Figure 1 is shown in Figure 7. In that Figure, each node (state) is labeled with what we shall call its *upward-level*. The upward-level of a leaf node is zero, and the upward-level of a non-leaf node is 1 plus the upward-level of its direct sons. With this definition, a node may have more than one associated upward-level. That is the case of the node associated to state A_2. Also observe that we are not associating upward-levels to AND-states, neither considering them when labeling the other nodes (OR states). This

[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.

is so because AND nodes are abstract entities that are not mapped into data objects in the model, thus making their labeling irrelevant.

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

Show-hview :

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 3 with L = 0, if the user executes *Show_hview* the hypertext system might produce the screen depicted in Figure 8, where the hierarchical view window (*Show_hview*) partially overlaps the main window.

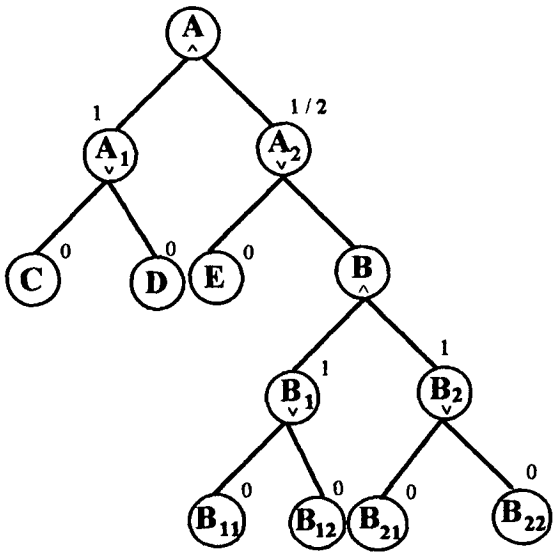


Figure 7: AND/OR tree for statechart in Figure 1.

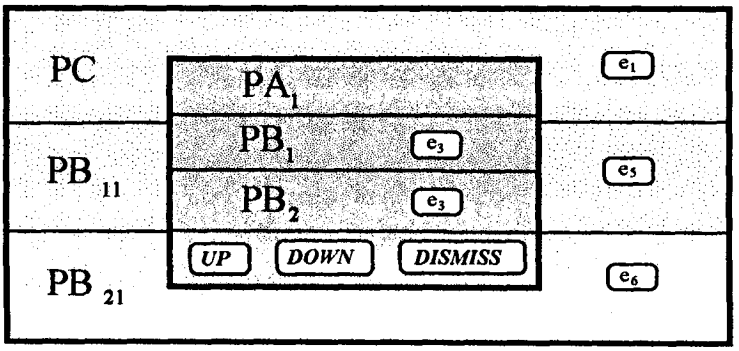


Figure 8: Activating *Show_hview* during the browsing of the hypertext.

The screen generated by *Show_hview* enables three adicional 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 > L *then* h = h-1; *Show_hview*.

Note that the execution of *Show_hview* does not modify the current statechart configuration but only provides access to the nodes which are one level up (or down) in the hierarchy, as long as at least one such a node exists. If there is no OR node up in the hierarchy (or, alternatively, no OR or basic node down), the state in the current level remains being shown. Such a situation is illustrated analysing the configuration depicted in Figure 8, and assuming that button Up was pressed. The pages shown in the hierarchy window as a result of such selection would be those associated to nodes A_1 and A_2. If button Up is pressed again these same pages remain shown in the hierarchy window.

In addition to navigating hierarchically through the document by activating the operations UP and DOWN, the user may activate the anchors enabled by the *Show_hview* operation. When an anchor brought up by a *Show_hview* operation is activated one of two possible interpretations may be taken:

- once an anchor is activated, the corresponding transition is fired and the new screen layout displays the pages associated to the new statechart configuration and the pages resultant from applying the *Show_hview* operation on this configuration. The value of h is maintained.
- once an anchor is activated, the corresponding transition occurs and the new screen layout displays the pages associated to the new statechart configuration.

The first alternative may be interesting if the hypertext is composed by a uniform hierarchical structure (e.g., 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 anchors shown as a result of a *Show_hview* operation and thus such an operation should be available only for viewing pages up or down in the hierarchy. We believe this alternative is too restrictive, however.

3. Solutions Using the Model

The HMBS model lends itself to providing simple solutions to some common problems related to hypertext visualization, browsing and analysis. Alternative solutions

based on the statechart model for some of the problems discussed in [Sto89] are described below.

3.1. Node Reachability and Display Characteristics

The HMBS model allows the determination of some parameters relevant for the physical presentation of the hypertext and for analysis of the reachability of nodes in the hypertext.

The reachability tree of a statechart is constructed to describe every possible computation sequence for the statechart [Mas94b]. At each step it is assumed that an event expression evaluates to true thus firing a transition that leads to a new state configuration. The semantic rules to compute each step are the ones defined in [Har87b]. The computation evolves in a sequence of time steps, which can be represented by the set $\{(SC_i, \gamma_i), i \geq 0\}$, where

1. $SC_0 = (X_0, \Pi_0, \Theta_0)$, where X_0 is the initial configuration, Π_0 is the set of external events generated by the environment (e.g., selection of an anchor) and Θ_0 is the set of primitive conditions whose value is true.
2. 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 system configuration reached from SC_i when the transitions in $\gamma_i, i \geq 0$ are taken.

The reachability tree is based on statechart configurations. We shall use the following notation for representing a configuration: parenthesis denote OR-superestates; square brackets denote AND-superestates; and the values 1 and 0 represent basic states, indicating whether they are active or not active, respectively. This notation preserves hierarchy and decomposition type although only the basic states are explicitly represented. The superestates are inferred from the configuration hierarchy.

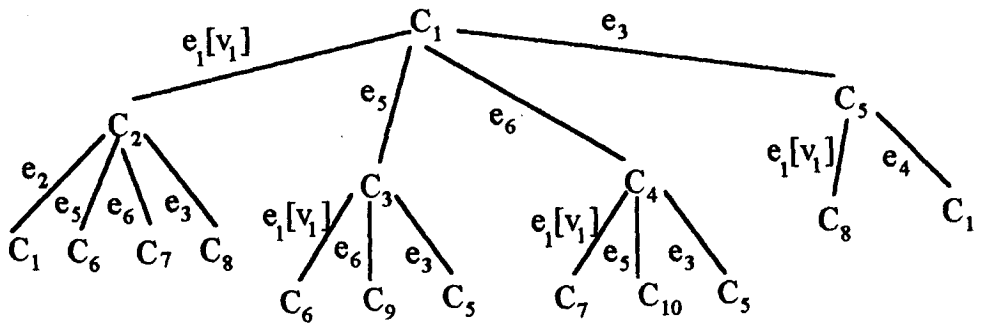
For the statechart shown in the Figure 1, the node description of the tree is:

$$[(C,D), ([(B_{11},B_{12}),(B_{21},B_{22})],E)]$$

As an illustration, Figure 9 gives the reachability tree for the statechart in Figure 2.

The reachability tree can be used to determine if portions of a hypertext can actually be reached during browsing, or whether there are any nodes that can never be reached. Given a hypertext H, and an initial state configuration SC_0, a simple way of determining if a particular page *ps* associated to a state *s* can be viewed during browsing is to compute the reachability tree for the underlying statechart model and analyze the nodes of the tree to verify whether state *s* appears in any of the state configurations generated. If the state does not appear in any reachable configuration, then the associated information cannot be viewed when the hypertext 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 the associated states, or looking for a specific set of states within any of the reachable configurations.



Where the configurations are:

C_1=[(1,0),([(1,0),(1,0)],0]
C_2=[(0,1),([(1,0),(1,0)],0]
C_3=[(1,0),([(0,1),(1,0)],0]
C_4=[(1,0),([(1,0),(0,1)],0]
C_5=[(1,0),([(0,0),(0,0)],1]
C_6=[(0,1),([(0,1),(1,0)],0]
C_7=[(0,1),([(1,0),(0,1)],0]
C_8=[(0,1),([(0,0),(0,0)],1]
C_9=[(1,0),([(0,1),(0,1)],0]
C_{10}=[(1,0),([(0,1),(0,1)],0]

Figure 9: Reachability tree for the statechart of Figure 2.

The reachability tree also enables the detection of terminal state configurations, that is, state configurations from which no other state can be reached because no transitions are enabled from them. These correspond to those leave nodes in the reachability tree that are not duplicates of other non-leave nodes. Another interesting characteristic to look for in the reachability tree is the presence of cyclical browsing paths, whose occurrence indicates that a browsing section can return to a previous state.

Yet another class of properties that can be checked for a browsing section are temporal order relationships amongst the pages visited. The semantics of the statecharts determines the set of all sequences of transitions that can be fired from a given initial state during execution of the statechart. Considering the associated hypertext, we are interested in sequences of anchor selections which may be made by a reader from an initial node. Consequently, the semantics of the statechart can be analysed to verify any browsing sequence restrictions an author wishes to impose on a hypertext. For example, a condition that any browsing section which encounters a page *py* must have previously encountered page *px*, or a condition that a browsing section which encounters a page *py* must not subsequently encounter page *px*, may both be checked for in the reachability tree.

Using the reachability tree of the underlying statechart model it is also possible to determine, for example, the maximum number of windows (or window subdivisions, depending on the browsing policy adopted) that will be required for any reading of the hypertext. Since the model associates a page $p \in P$ to every state in set $S_s \subset S$, if the

browser displays each concurrently viewed element in a single window that is subdivided to show all the elements, then the number of states in the current statechart configuration is equal to the number of window subdivisions that will be necessary for that configuration. If it displays each concurrently viewed element in a separate window, then the number of states in the current statechart configuration is equal to the number of windows that must be simultaneously displayed.

Therefore, we can analyse the nodes in the reachability tree to find the maximum number of active states for all possible configurations. For the reachability tree shown in Figure 9 this number is 3. This information can be employed to aid the determination of reasonable layouts for a display mechanism that subdivides a single window or tiles multiple windows on the screen.

Such an analysis presumes that concurrently viewed elements are specified by the hypertext author rather than the reader, and therefore the browsing semantics does not allow a reader to simultaneously initiate separate traversals of the various alternative paths leaving a node. It is also important to mention that the size of the tree generated may be critical for hypertexts with a great number of nodes.

3.2. Synchronization of Simultaneous Display

Statecharts provide a computational model adequate for expressing concurrency. Therefore, they have suitable mechanisms for representing concurrent display of multiple pages and concurrent browsing paths in a hypertext, as well as for providing authors the support for specifying sinchronization of concurrent activities.

Concurrent activity may appear for example 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. In analogy to [Sto89], who present a Petri Net that models such a situation, shown in Figure 10, we present a corresponding statechart model in Figure 11.

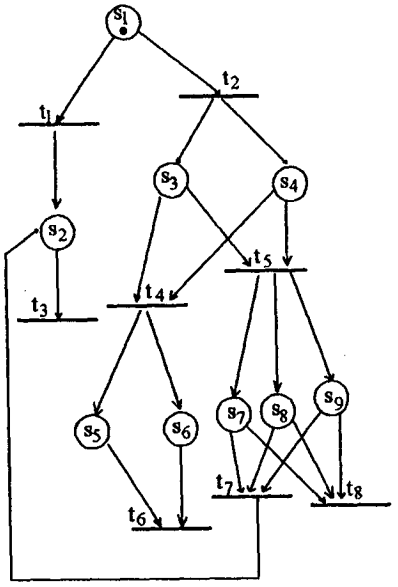


Figure 10: Petri net representation example (without hierarchy).

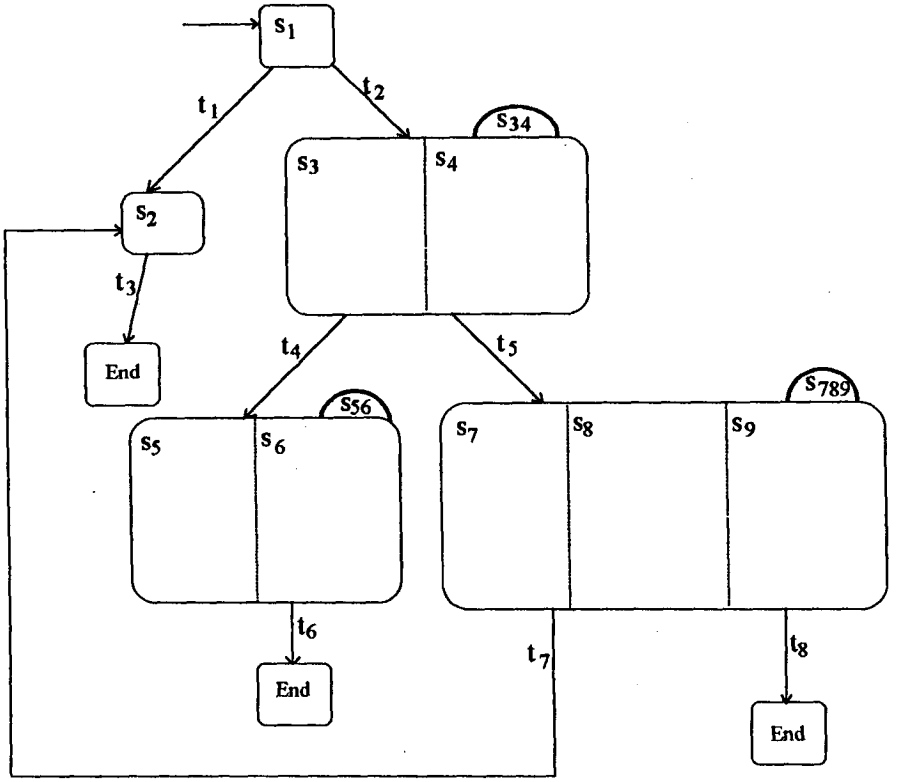


Figure 11: A statechart model with concurrent states, equivalent to the Petri Net depicted in Figure 10.

This type of concurrent activity may be described by a statechart in which a single transition (that models an anchor in the hypertext and may be physically mapped into a button, for example) branches to several parallel (AND) states (which are associated to the presentation of multiple concurrent pages), and a second transition causes all the parallel states to be simultaneously abandoned. In Figure 11 there are three AND states, s_{34}, s_{56} and s_{789}, consisting of multiple sub-states corresponding to concurrently viewed

pages. In that example, from a current state configuration equal to $SC_0 = \{s_3,s_4\}$, the firing of transition t_4 will cause both states s_3 and s_4 to be left, leading to a new configuration $SC_1 = \{s_5,s_6\}$, whereas the firing of transition t_5 will also cause both states to be left, leading to a different new configuration $SC_2 = \{s_7,s_8,s_9\}$. The end states indicate a final state with an associated *null* page. Following the semantics of the statecharts, the three pages associated to these states will become visible after transition t_5 is fired (by selecting the corresponding button). The three pages will be removed from the display when either transition t_7 or t_8 is fired. The statecharts' AND states thus provide a natural mechanism for specifying the parallelism inherent to the situation, as all the concurrent elements are explicitly depicted in the statechart, which is not the case for Petri nets.

3.3. Access Control

The set of boolean variables associated to a model in HMBS may be used to provide a simple yet effective mechanism to enforce browsing restrictions on readers of a hypertext. The HMBS model ensures that access capabilities are an integral part of the document, and allows that different browsing paths share one or more common nodes but still have interspersed mutually exclusive sections, overcoming the limitations of annotated directed graphs.

We illustrate this point using the example presented in [Sto91] which considers an imaginary situation where an employment record is to be accessed by 2 classes of readers: a privileged class (PR) and a restricted class (RR). Readers of class PR can browse the entire document, but readers of class RR may not see the sections containing job performance evaluations. A statechart dealing with this scenario is presented in Figure 12. Let the pages associated to states s_3 to s_5 contain the job evaluations; as discussed earlier the structure specifies that the three are to be displayed concurrently. The pages referring to states s_{12} and s_{13} also contain privileged information, and their contents are also to be displayed simultaneously after their evaluation have been read. The rest of the unemployment record (pages associated to states s_2, s_6 through s_9, and s_{11}) contains unrestricted information. In the statechart, access control to restricted nodes is governed by two global variables v_1 and v_2. According to the semantics of the statecharts, a transition $t_1[v_1]$ is fired only if event t_1 is activated and variable v_1 is true. Thus, for a class PR reader, the initial setting is $V_T = \{v_1,v_2\}$ and $V_F = \varnothing$, therefore enabling access to the restricted pages. For a class RR reader, the initial setting is $V_T=\phi$, and $V_F = \{v_1,v_2\}$, so that a reader is not granted access to nodes (s_3,s_4,s_5) and (s_{12},s_{13}).

Therefore, the links associated with transitions t_1 and t_7 will not be displayed for these users, and transitions t_2, t_9 and t_{10} will not be visible for them.

This approach requires the generation of two reachabitly trees, one for each initial setting of sets V_T and V_F. A third class of reader might be defined, comprising semirestricted (SR) readers which may, for example, see the salary information in the pages associated to nodes s_{12} and s_{13}, but are not allowed to see the performance evaluation kept in the pages associated to nodes s_3 to s_5. This restriction class is created with an initial setting of $V_T = \{v_2\}$, $V_F = \{v_1\}$. An examination of the reachability tree for this hypertext shows that a reader may fire transition t_7, but that transition t_1 will never be enabled, and configuration $\{s_3,s_4,s_5\}$ will never be reached.

To construct documents with access control classes, an author might proceed through the following steps: identify the access control classes for the hypertext, deciding which nodes are restricted for each class; then associate a control variable to each node that is restricted for some class; and finally define a set V_T for each control class.

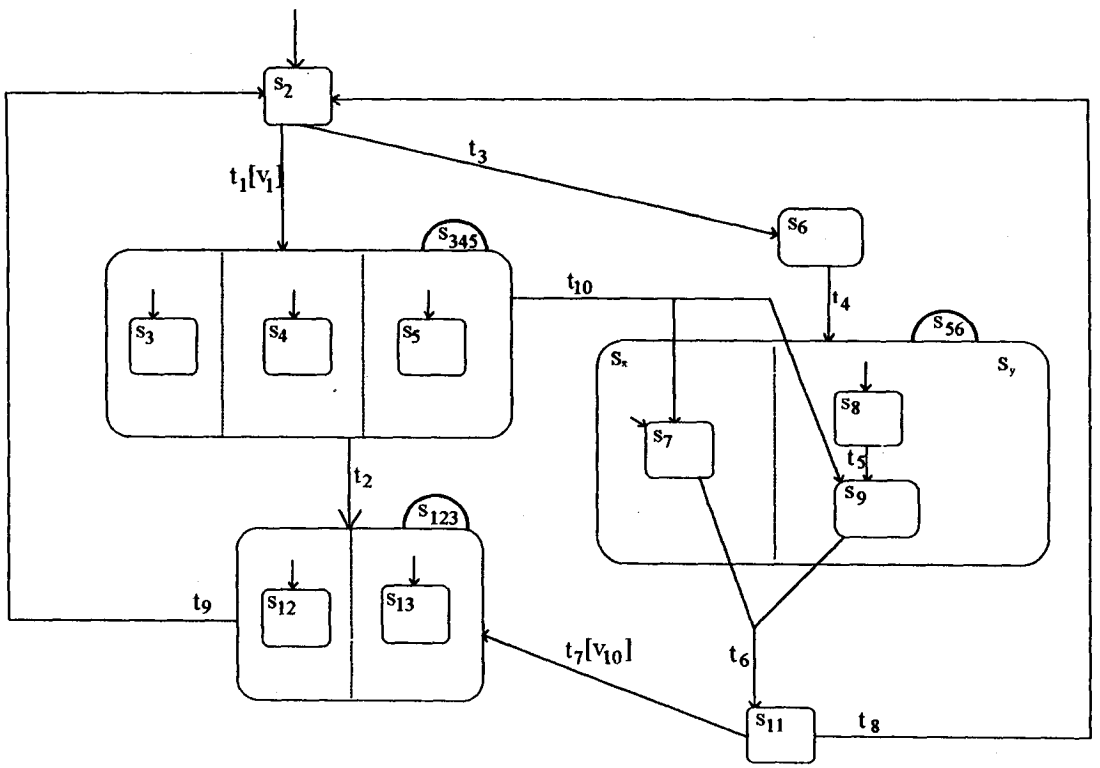


Figure 12: Statechart with access control through boolean variables.

3.4. Tailored Versions of a Hypertext

A single statechart model may be used to keep a collection of distinct tailored versions of the same hypertext. This may be achieved by delivering the same hypertext model with alternative mapping functions M and set of pages P, thus allowing the association of different pages to the nodes in set S_S. Such a solution assumes that only the contents of the nodes are altered in the different versions, while the underlying hypertext structure is maintained. This is the case, for example, when different versions of the same hypertext 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 is the same discussed in the section on access control in browsing and the same solutions apply, although the necessary reachability graph analysis may differ. Properties to be checked for 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 (i.e., all the pages of a particular version are potentially viewable).

3.5. Direct Access to Pages

An index providing direct access to pages may be constructed from the available page titles, and readers may invoke this index in order to perform searches (string search, linear search, etc.). Page titles shown in the index are restricted to those corresponding to nodes satisfying $max(upward\text{-}level) \leq L$. Once a page P is chosen which corresponds to a node N, it is shown in conjunction to the pages corresponding to nodes that with N define a valid set of active states which are chosen by default. For example, consider the statechart of Figure 2 with the M mapping depicted in Table 1, L = 1, and the AND/OR tree given in Figure 7. Assuming that during a browsing section the user has selected the page with title TB_1. The resulting display shows pages PB_1, PB_{11}, PB_2, PB_{21}, PA_1 and PC as presented in Figure 5. This is because SC = {B_{11}, B_{21}, C} is a valid state statechart configuration extended with default states that includes state B_1 as an ancestor.

An algorithm for generating a valid state configuration for a given node and page is outlined bellow:

Algorithm 1:

- Initialize a state set S as an empty set;
- Given a page P corresponding to a node N of the AND/OR tree;
- Add N to S;
- *If* N is not a leaf node *then*

    AddDefaultDescendent(N,S);

- *For* each ancestor M of N *do*
  Add M to S;
  If M is an AND node *then*
    For each direct son K of M (excluding the one in the ancestorial path) *do*
      Add K to S;
      AddDefaultDescendent(K,S);
- Display all pages associated to nodes x in S according to the HMBS browsing semantics and the visibility level L;
- Let SC be the state configuration containing the basic states of S;

Where *AddDefaultDescendent* is a recursive procedure that visits all descendent nodes of an initial node, adding its default descendents to a set of active states.

Algorithm 2:

AddDefaultDescendent(X,S)

Begin
  If X is basic *then*
    return
  Else If X is an OR node *then*
    Choose X′ as the default state of X;
    Add X′ to S;
    AddDefaultDescendent(X′,S);
  Else If X is an AND node *then*
    For each direct son X′′ of X *do*
      Add X′′ to S;
      AddDefaultDescendent(X′′,S);
End.

A more general search, without the restriction *max(upward-level)* $\leq L$ would allow direct access to all pages of a hypertext. In this case, a composition of hierarchical view and plain view (a window without a *show-hview* sub-window) would be necessary to show a chosen page associated to a node N. The basic algorithm to restore the set of active states would be the same, and the hierarchical view would correspond to a sequence UP_1 (equivalent to *Show-hview*), UP_2, UP_3, ..., UP_k, where k depends on the *upward-level* of node N and on the state configuration. Considering for example the statechart of Figure 2 with the M mapping depicted in Table 1 and the AND/OR tree given in Figure 7, and assuming that the user has selected the page with title TA_2 then:

• Assuming that state E is chosen as the default for A_2, the valid set of active states would be $\{A_2, A_1, A, E, C\}$ and the corresponding state configuration is SC = {E, C}. In this case, k = 1 (*upward-level* of A_2 following the path from state E) and pages PA_1, PA_2, PE and PC are displayed.

• Assuming that state B_{11} is chosen as the default for B_1 and B_{21} as the default for B_2, the valid set of active states would be $\{A_2, A_1, A, C, B, B_1, B_2, B_{11}, B_{21}\}$ and the corresponding valid configuration is SC = { B_{11}, B_{21}, C}. In this case, k = 2 (*upward-level* of A_2 following the path from states B_{11}, B_{21}) and the display shows the pages PA_1, PC, PA_2, PB_1, PB_2, PB_{11}, PB_{21}.

4. Related Work

The model implicit in many hypertext systems is the *labeled directed graph*, where the contents of a hypertext are associated with the contents of a set of nodes, and nodes marked with tokens are displayed. The labelling of the links associated with the users' logical and physical selections may be established via buttons, menu choices, etc. Although extremely simple, the model unfortunately does not represent the structure of the data and the browsing semantics adequately, and provides no separation between the structural organization of the nodes and their contents.

Tompa [Tom89] 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 behaviour 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 behaviour.

Stotts and Furuta [Sto89] 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 the browsing semantics of a hypertext. The Trellis model provides elegant support for solving several problems inherent to hypertext sytems, such as analysis of display complexity and node

reachability, providing concurrent browsing paths and syncronization, access control, and tailored versions of hypertexts.

However, a major shortcoming of the Petri net based model is the lack of satisfactory hierarchical structuring facilities, which difficults the task of especifying synchronization control across different levels of hierarchical structures. Only one level of hierarchical decomposition can be accomodated in the Petri net, as a hierarchical state is not another Petri net in a lower level of decomposition, but rather a set of multiple substates 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 substates that are also statecharts, and behave as so.

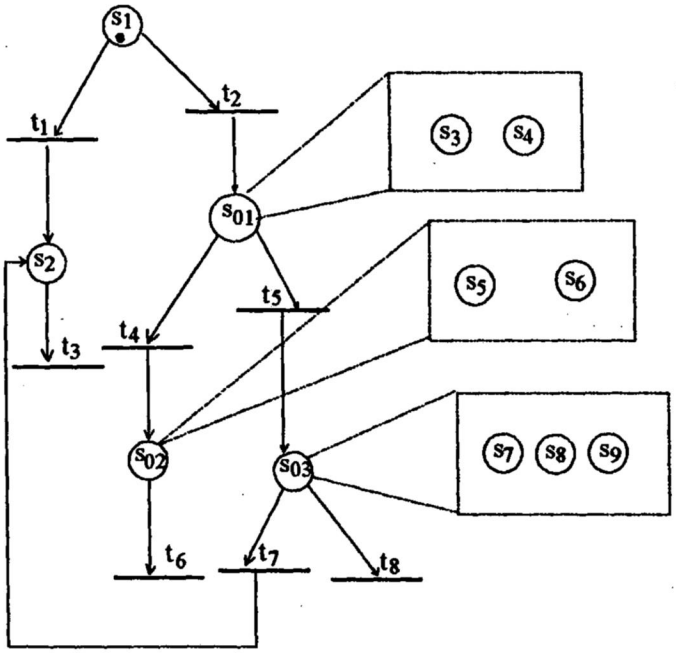


Figure 13: Petri net representation example (using hierarchy).

As an illustration, Figure 13 models the same situation depicted in figure 10 using a hierarchical Petri net representation. Comparing both representations, the statechart in Figure 11 and the Petri net in Figure 13 [Sto89], we also observe that the statechart representation is clearer, as the statechart includes suitable hierachical structuring facilities which are inherent to the model. In Figure 13, a further decomposition of state s_3 is not easily accomodated in the Petri net, and there are no mechanisms for specifying synchronization within this state decomposition. In the HBMS model, however, this decomposition is directly mapped into sub-states of state s_3, and any further level of structural decomposition is easily represented.

According to Zheng & Pong [Zhe92], the behaviour 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 undertand the browsing semantics of hypertexts 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, the HBMS model provides a more intuitive model for describing the browsing semantics of hypertexts.

HDM (Hypermedia Design Model) [Sch92, Gar93], on the other hand, is a model oriented to the description of hypertext 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 behavioural aspects of hypertext, or with operational features, HDM, as the HMBS model, is concerned with the structural organization of hypertexts and with representational issues. A distintiction between both models is that HDM is based on the definition of a schema which may be instantiated for all documents of the same type, whereas HMBS models the structural organization of each hypertext. However, nothing prevents that a single hypertext model be shared by different documents, or by different versions of the same document. Such use of statecharts is actually suggested by Masiero et al. [Mas94a], 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 mantained in a hypertext database when new documents are added to the database.

HDM shares some apparent similarities with the Entity Relationship (E-R) model and the HMBS model is based on Statecharts. It is interesting to observe that Harel, in [Har88], showed that both models, E-R and statecharts, are based on a common underlying concept called Higraph (hierarchical graph), which brings together the concepts of Euler-Venn diagrams and Hyperedges. Therefore, in principle, any HDM models are convertible to corresponding HMBS models.

5. Discussion and Implications of the Model

This paper introduced the *HMBS* model, which uses the structure and execution semantics of the statecharts visual formalism to specify both the structural organization and the browsing semantics of a hypertext. HMBS models are not restrict to hypertext systems in which a single element is visible at any time during browsing (as it is the case in directed graphs); rather, the model allows simultaneous visualization of multiple windows and provides proper support for concurrency and syncronization through the use of AND-states for modeling synchronized browsing actions.

The model is particularly suitable for modeling hypertexts that present a hierarchical structure, such as books, scientific papers, on-line manuals and reports, as the state hierarchy imposed on the underlying statechart model provides adequate

mechanisms for modeling such hierarchical organization and the possible browsing actions to be taken [Tur95].

Several extensions of statecharts which were not considered in the original HMBS model could be further explored in the context of hypertext modeling. The history mechanism of the statecharts, for example, ensures that, if an OR state is entered, control is returned to the last active state, rather than to the default state. This facility could be explored for modeling hypertext on-line help applications in which several different situations result in the activation of a single link. Such situations could be mapped into states grouped in a superestate, and the history information may be used to ensure that control is returned to the specific node in which the link was triggered.

If integer variables and actions are included in the model, and altering the values of variables during the browsing sections is allowed, other interesting hypertext applications could be considered, such as programmed learning due to execution of actions associated to the transitions. A system could select the most adequate paths for a given user considering his/her knowledge about the subject, which might be inferred with basis on a history of user choices made during one or more browsing sections, or, alternatively, based on a series of grades assigned to the user.

The broadcast mechanism could be explored in order to provide proper synchronization support for cooperative work, e.g., collaborative writing, in distributed environments. This is also due to the execution of actions that generate events which may be captured by orthogonal components. A paper being read by two readers, for instance, may be simultaneously available for both, but it is interesting that only one be allowed to control the reading flow. Execution of a HMBS model on a distributed environment raises interesting implementation issues that are out of the scope of this paper.

6. Conclusions

The HMBS model uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of a hypertext. 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.

We are initiating the construction of a prototype hypertext system in order to experiment with the ideas presented in this paper. This prototype shall include an

authoring and a browsing modes, and will be integrated into the STATSIM environment [Mas91]. STATSIM is a statechart simulation environment which integrates a graphical editor for describing and simulating statecharts. We are considering the development of an on-line help for STATSIM itself as a first potential application to be carried out using this prototype system. We also intend to incorporate a "statechart engine" into the prototype hypertext system that will enable its use within an office information system in which automatic authoring is supported for certain classes of office documents, as described in [Mas94a].

Finally, we point out that, although we restricted ourselves to the modeling of hypertexts, the applicability of the model to hyperdocuments containing information to be displayed through different media is a point that deserves further investigation.

References

[App94] Apple Computer. *Hypercard: stack design guidelines.* Addison-Wesley Publishing Company Inc., april 1994, 229p.

[Aks88] Akscyn, R.M.; McCracken, D.L. & Yoder, E.A. KMS: a distributed hypermedia system for managing knowledge organizations. *Communications of the ACM*, 31(7), p.820-835, july 1988.

[Bro87] Brown, P.J. Turning ideas into products: the Guide System. In: *Proceedings of the ACM Hypertext'87* (Chapel Hill, N.C.), p.33-40, 1987.

[Con87] Conklin, J. Hypertext: an introduction and survey. *IEEE Computer*, 20(9), p.17-41, september 1987.

[Con88] Conklin, J. & Begeman, M.L. gIBIS: a hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4), p.303-331, october 1988.

[Gar88] Garg, P.K. Abstraction mechanisms in hypertext. *Communications of the ACM*, 31(7), p.862-870, 1988.

[Gar93] Garzotto, F.; Paolini, P. & Schwabe, D. HDM — a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, 11(1), p.1-26, january 1993.

[Hal94] Halasz, F. & Schwartz, M. The Dexter hypertext reference model. K. Gronbaek & R. Trigg, Eds. *Communications of the ACM*, 37(2), p.51-62, february 1994.

[Har87a] Harel, D. STATECHARTS: a visual formalism for complex systems. *Science of Computer Programming* , 8, p.231-274, 1987.

[Har87b] Harel, D. On the formal semantics of statecharts. In: *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science*. Ithaca, New York, p.54-64, 1987.

[Har88] Harel, D. On visual formalisms. *Communications of the ACM*, 31(5), p.514-530, 1988.

[Har92] Harel, D. & Kahana, C.A. On Statecharts with overlapping. *ACM Transactions on Software Engineering and Methodology*, 1(4), p.399-421, october 1992.

[Mas91] Masiero, P.C.; Fortes, R.P. de M.; Batista Neto, J. do E. S. Editing and simulating the behavioral aspects of real-time systems (in Portuguese), Proceedings of the XVIII Integrated Hardware and Software Seminar, SEMISH, p.45-61, August 5-9, 1991.

[Mas94a] Masiero, P.C.; Oliveira, M.C.F.; Germano, F.S.R.; Santos, G.P.B. Authoring and searching in dynamically growing hypertext databases. *Hypermedia Journal*, 6(2), p.124-148, 1994.

[Mas94b] Masiero, P.C.; Maldonado, J.C.; Boaventura, I.G. A reachability tree for statecharts and analysis of some properties. *Informantion and Software Technology*, 36(10), p.615-624, 1994.

[Rei91] Rein, G.L. & Ellis, C. rIBIS: a real-time group hypertext system. *International Journal of Man-Machine Studies*, 34, p.349-367, 1991.

[Sch92] Schwabe, D; Caloini, A.; Garzotto, F. & Paolini, P. Hypertext development using a model-based approach. *Software-Practice and Experience*, 22(11), p.937-962, november 1992.

[Sto89] Stotts, P.D. & Furuta, R. Petri-net-based hypertext: document strucuture with browsing semantics. *ACM Transactions on Information Systems*, 7(1), p.3-29, january 1989.

[Sto91] Stotts, P.D. & Furuta, R. Dynamic Adaptation of hypertext structure. In: *Hypertext'91 Proceedings*, p.219-230, december 1991.

[Sto92] Stotts, P.D. & Furuta, R. Hyperdocuments as automata: trace-based browsing property verification. *ACM ECHT Conference*, p.272-281, november 30 - december 4, 1992.

[Tom89] Tompa, F.W. A data model for flexible hypertext database systems. *ACM Transactions on Information Systems*, 7(1), p.85-100, january 1989.

[Tur95] Turine, M.A.S.; Oliveira, M.C.F. & Masiero, P.C. Designing hyperdocuments with HMBS. Technical Report, ICMSC-USP, 1995 (In preparation).

[Zhe92] Zheng, Y; Pong, M. - Using statecharts to model hypertext. Proceedings of the European Conference on Hypertext Technology, Milano, ACM Press, p. 242-50, 1992.

NOTAS DO ICMSC

SÉRIE COMPUTAÇÃO

018/95 PIMENTEL, M.G.C. - Alternative operations for browsing hypertext.

017/94 ROMEIRO, N.M.L.; CASTELO FILHO, A. - Análise Comparativa de Métodos Numéricos de equações algebrico-diferenciais.

016/94 MAGALHÃES, A.L.C.C.; SIQUEIRA, M.F.; OLIVEIRA, M.C.F. - Operadores de Euler na modelagem por fronteira: conceito, aplicação, estudos de casos.

015/94 ODA, C.S.; MOREIRA, E.S. - ASNMP graphical network monitor with automatic topology discovery.

014/94 FELIPE, L.S.G.; FRANCO, N.M.B. - Sobre a ordem de convergência para as equações integrais de volterra de segunda espécie tipo Abel com soluções não suaves.

013/94 PIMENTEL, M.G.C. - A framework for user-hypertext interaction.

012/94 TURINE, M.A.S.; MENDES, M.D.C.; NUNES, M.G.V. - TEGRAM: a geometry tutoring system based on Tangram.

011/94 SPOLON, R.; SPOLON, R.; SANTANA, M.J.; SANTANA, R.H.C. - Desenvolvimento de um gerador de aplicação para simulação de sistemas discretos.

010/94 SAWAKI, J.; MONARD, M.C.; RODRIGUES, S.R. SABNAG: - um sistema baseado em conhecimento para suporte aos usuários da biblioteca NAG.

009/94 NICOLETTI, M.C.; MONARD, M.C. - Learning restricted Horn clauses: some considerations on the ij-determination concept.