Applying caT's Programmable Browsing Semantics to Specify World-Wide Web Documents that Reflect Place, Time, Reader, and Community

Richard Furuta
Center for the Study of Digital Libraries
and
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, USA
furuta@csdl.tamu.edu

Jin-Cheon Na
Division of Information Studies
School of Communication & Information
Nanyang Technological University
31 Nanyang Link
Singapore 637718
tjcna@ntu.edu.sg

ABSTRACT

In this paper we discuss application of caT, which extends the Trellis Petri-net-based model of document/hypertext, towards specification of Web-browsable documents that respond to their reader's characteristics, browsing activities, use environment, and interactions with other readers. The Petri net basis provides both a graphical representation of the nodes and links in the hypertext and also an automaton-based specification of the browsing behaviors encountered by readers examining the hypertext. Providing Web-browsable responsive hypertexts in the caT context requires consideration of the structures that might be designed in support of the application and also of the mechanism for translating from caT's custom interfaces' multi-window presentation to a composite that can be viewed using a standard Web browser.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia; I.7.2 [**Document and Text Processing**]: Document Preparation—*Hypertext/hypermedia*

General Terms

Design, Theory

Keywords

caT, Trellis, Petri-net-based hypertext, context-aware hypertext

1. INTRODUCTION

As the reading audience for the World-Wide Web increases in number and diversity, and as the viewing venues broaden from wired workstations to wireless networks and miniaturized displays, so also does the need develop for Web-based document/hypertext collections that can be reshaped to respond to their user and use environments. The information of relevance to a use may vary with time of day or with location (e.g., direction to see a book on a library's shelf may be useful if you are in the library building, but is not likely to be useful if you are traveling in an airplane). The presentation of information may differ depending on device and network characteristics. Additional resources may be available, or may be unavailable, if other readers are also viewing the same document collection.

Specification of such hypertextual collections requires mechanisms that can reflect these characteristics of the external use environment. Engineering such collections further requires that the mechanisms lend themselves to specification reuse and that they provide a "hook" that makes verification and validation of the document's properties feasible. The goal of the work described in this paper is to enable a principled specification of such responsive hypertexts. The mechanisms used are application of programmable browsing semantics, as introduced in Trellis and extended in caT, to be described next.

Beginning in the late 1980's, Furuta and Stotts' Trellis project [24] investigated hypertext applications specified by Petri nets. Trellis defines not only a hypertext's nodes and links, but also the its browsing semantics—where *browsing semantics* mean the document behaviors (e.g., the sequence of information elements displayed and links made available) encountered during a reader's browsing session. This is enabled by the dual nature of Petri nets—both graph structure (representing the nodes and the links) and also associated automaton semantics. Applying the automaton's semantics defines the browsing semantics.

Trellis browsing semantics are called "programmable," since localized change to the specification can be used to change browsing behavior. In addition, the specific browsing behavior of a hypertext specification depends on its initial state (i.e., which of its components were designated to be active when browsing began), so selection of a different initial state also could change the behavior (a virtual change to the specification rather than a physical change). The early Trellis papers demonstrate how these characteristics can be used, for example, to implement different access categories, either enabling or denying access to selected portions of the hypertext depending on the desired access policy. A key point is that the policy was defined by the hypertext's *specification* and not as attributes defined and interpreted by a system's *implementation*. Consequently, the mechanism allows the flexible specification and

DocEng'02, November 8–9, 2002, McLean, Virginia, USA.

modification of user-related access policies. A subsequent Trellis paper [11] demonstrates use of the mechanisms to define the roles and activities of participants in a group meeting, and how such definitions can be modified to reflect different meeting protocols.

Trellis specifications incorporate parallelism—multiple content elements can be active simultaneously, and consequently be displayed simultaneously. One application of parallelism is to represent different information threads, active simultaneously but traversed independently. Another is to represent synchronous traversal of separate content elements—for example separate media components of a multimedia presentation. Furthermore, the automaton semantics provide a hook that allows the application of analysis techniques developed by others in different domains to questions of interest in the hypertext context (e.g., can this node be reached, are there any nodes without outgoing links, etc.).

Trellis browsing semantics provide a mechanism by which a hypertext can *respond* to events in its environment. As examples, Trellis hypertexts can be designed to dynamically permit or limit access to information, to select different information for presentation on initial and subsequent accesses, and to respond differently if multiple readers are viewing a section of a hypertext simultaneously. We will discuss these points in more detail later in the paper.

Over the past few years, we have extended the Trellis model to further support hypertext responsiveness. We call the revised model caT (context-aware Trellis) [19], reflecting its additional support for incorporating, into the specification, characteristics of the external environment in which the hypertext is being used (e.g., reader's physical location, time of day, user characteristics such as age and job title, etc.). In addition, the caT extensions permit the hypertext state to better encapsulate information about the reader as an individual, augmenting the less specific support for representing classes of readers provided in Trellis.

The natural mode for authoring caT hypertexts proceeds from the structure to the content (i.e., top-down)—in other words, the link graph structure of the hypertext is developed first, and then content is associated with the nodes. On the other hand, the natural mode for authoring World-Wide Web documents seems to be to create the content within Web pages first and then to embed links to other Web pages. Although technologies such as XML provide a framework that tends to regularize the organization within pages, the overall specification remains bottom-up in character.

Indeed, one might argue that it is the node that is the primary abstraction within the World-Wide Web while it is the link structure that is primary in caT. We believe that this characteristic makes it more convenient to achieve manageable hypertext collections in caT than in more conventional World-Wide Web specifications.

The remainder of the paper is organized as follows. Section 2 provides an informal overview of the portions of the caT model necessary to understand the paper's examples (further details can be found in other papers; e.g., [19]). We also illustrate how we present caT hypertexts for viewing on the World-Wide Web. Section 3 returns to the topic of responsive hypertexts, illustrating their characteristics through example. Related work is identified in Section 4. Finally, Section 5 provides discussion and conclusions.

2. caT AND TRELLIS

This section contains an informal introduction to caT's Petri-net-based hypertext model. A formal description can be found in [19].

A Petri net (see, e.g., [21, 22]) is represented graphically as a bipartite directed graph, in which the circular nodes are called places and the bar nodes are called transitions. A dot in a place represents a token, and a place containing one or more tokens is said to be marked. When each place incident on a transition is marked that transition is enabled. An enabled transition may fire by removing one token from each of its input places and putting one token into each of its output places.

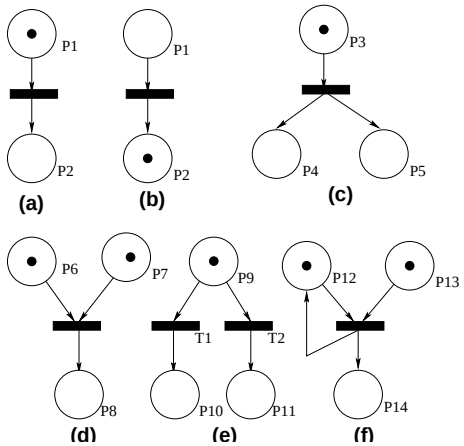

Figure 1: Petri net examples.

Figure 1 gives a few examples of Petri nets. Net (a) shows a simple case of a single place, which is marked, incident on a transition. When the transition is fired, the resulting specification is that in net (b). When the transition in net (c) is fired, the token in place `P3` is removed and one token is deposited into each of the places `P4` and `P5`; hence this can be thought of as a fork. The transition in net (d) will not become enabled until both places `P6` and `P7` are marked; this is a join. On firing, the net will contain a single token in place `P8`. Net (e) encodes a choice. Both transitions `T1` and `T2` are enabled, but only one can fire, since firing will consume the token in place `P9`. Finally, net (f) presents a slightly more complex example. Firing the transition will remove the tokens in places `P12` and `P13`, depositing tokens into `P12` and `P14`. Thus after firing, place `P12` remains marked.

Petri nets are mapped to hypertexts by associating content with places and links with transitions. When a place is marked, the corresponding content is displayed, and when a transition is enabled, the corresponding link can be selected. Consequently, nets (a) and (b) correspond to the display of a sequence of content elements, net (c) to two content elements following in parallel from a single element, net (d) to a single content element replacing two predecessors, net (e) to two links, either of which may be selected, and net (f) to a case in which one content element remains visible while the second is replaced by a new content element.

As a notational convenience, the actual form of Petri net used is the Colored Petri Net (CPN) [16]. Tokens in the CPN have a "color", and predicates on the arcs specify what color (or colors) must be present before the transition is enabled as well as the color that is produced once the transition is fired. In addition, transitions are augmented with non-negative timing values, (τ_r, τ_m), $\tau_r \leq \tau_m$. τ_r, the release time, represents the amount of time that must elapse after the transition is enabled before the system permits its firing. τ_m, the maximum latency time, represents the maximum amount of time that the transition will be permitted to be enabled; the system fires the transition automatically after time τ_m passes. Time values can be thought of as defining a range, a delay and a timeout, for the availability of an event. $(0, \infty)$ is the common case (available for firing immediately and never fired by the system). Other interesting values include $(0, 0)$, which fires itself immediately on enabling, and (∞, ∞), which can never be fired.

The mechanisms described so far in this section were defined in Trellis, and consequently have been carried forward into caT's definition. caT has expanded the definition in a number of ways, to be described in the remainder of the section.

In the caT model, a token has optional local variables in addition to its color value. In addition, global variables are found in a user profile. Conditional predicates, referring to the local and global variables, are associated with the transitions and must be satisfied before the transition is permitted to fire. Assignment statements, associated with the arcs from transition to place, set and modify the variables' values. Hooks are included in the system's implementation to allow invocation of an external fuzzy logic engine when necessary in the evaluation of predicates and in computing new values for variables.

Following Jensen's lead [16], caT incorporates a form of hierarchical Petri net—essentially the specification is built as a collection of subnets, tied together in a hierarchy. In caT's representation, a transition in a higher-level net may be mapped to a separate subnet. The transition that is expanded to the subnet is called the *substitution transition*. Additionally, places in the higher-level net are mapped to places in the subnet—generally, the input to the subnet corresponds to places that lead into the substitution transition, and the output from the subnet corresponds to places that come from the substitution transition. When a token arrives in a mapped place, it appears simultaneously in both nets. Consequently, tokens in the higher-level net are first conveyed to the subnet, and then back from the subnet to the higher-level net. Variables passed from the higher-level net to the subnet can be used to specify the mapping of content to place, so the same subnet definition can be used multiple times if a particular structure is used multiply in a hypertext.

2.1 Specifying composite nodes for dynamic Web page generation

In caT, a *template file* specifies how active content elements are to be displayed and how links are to be embedded when the reader browses the document on the Web. Content elements associated with a place have types that are used on display to determine how (or whether) the element should be rendered. The template file is simply another content type. It takes control when its associated place is marked; consequently multiple template files can be active at different times during a browsing session. The node that contains the template file can be viewed as specifying the directions by which a virtual composite node is constructed from the active atomic content elements (i.e., those associated with marked places); only the active atomic content elements specified in the template file are used for generating the display. Thus, the content of the composite node changes dynamically based on the current net state.

The template file is essentially an HTML page with placeholders indicating where information is to be embedded when the corresponding places and transitions are marked and enabled, respectively (see [19] for a definition of the template file's syntax). A template file includes a `<Template>` construct, which is followed by any number of `<Transition>` and `<Place>` constructs in any order. Additional text can be included as well; this text is copied verbatim into the output page.

Attributes associated with the `<Template>` construct specify global settings that control whether links are shown in a separate frame, the appearance of the links, if shown, and whether debugging controls are also included. The `<Transition>` construct maps active transitions in the net to corresponding anchors in the Web page. This construct's attributes also can be used to specify what the anchor's text will be when the associated transition

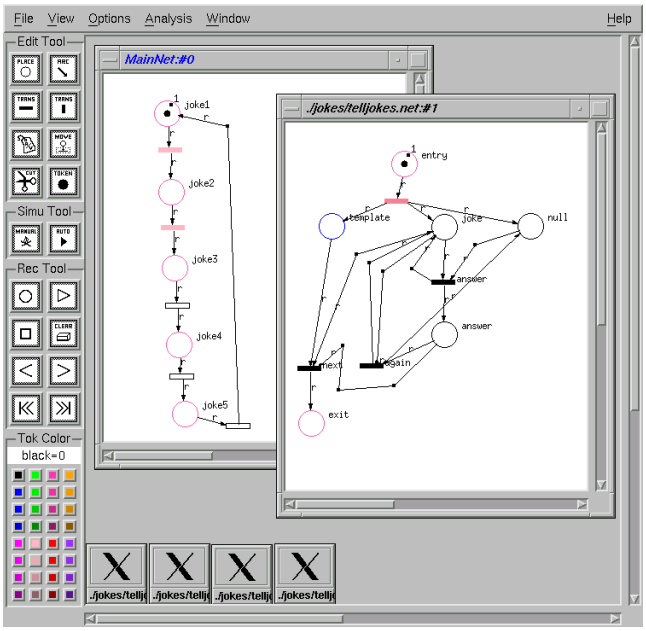

Figure 2: χTed display showing a hierarchical net specification.

```
<body bgcolor=#ffffff>
<h1>
<Template multiple_frame="no">
<Place name="joke" link_display="none">
<br>
<Transition name="answer"
activelink_name="Click here for the answer"
inactivelink_name="">
<Place name="answer">
</h1>
</body>
```

Figure 3: Template file associated with the subnet.

is active or inactive; if unspecified, the transition's name is used as the anchor text. Finally, the `<Place>` construct specifies a target place and display options for the output links of the place. The `<Place>` construct is replaced by the content of the target place when the place is active and suppressed otherwise. Attributes specify whether anchors should be generated automatically (i.e., if they are not specified within the content using `<Transition>` constructs), where they should be placed relative to the content, whether they should be sorted by anchor text, and whether they should appear multiple times if more than one arc leads to the transition.

We present a brief example to illustrate some of these mechanisms. Figure 2 shows the caT version of the χTrellis Petri net editor χTed with two subnets visible. The net shown in the window labeled `MainNet:#0` is the top-level definition, and invokes five instances of a subnet. One of these instances is shown in the other window and the other four are minimized. In the example, top-level place "joke1" is mapped to subnet place "entry" and "joke2" to subnet place "exit". Since "joke1" is marked, so also is "entry". The subsequent subnet instance maps "joke2" to "entry" and "joke3" to "exit", and so on. Each mapping in the main net also assigns different values to attributes "jokefile" and "answerfile".

Within the subnet, the value of attribute "jokefile" is specified to point to the content of place "joke" and the value of "answer-

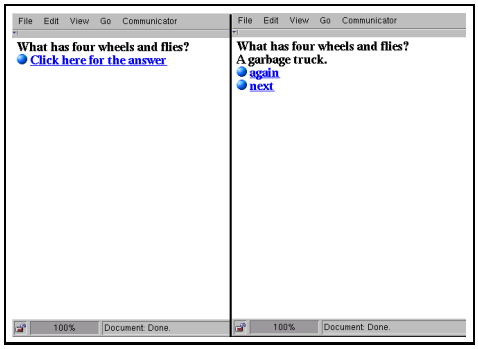

Figure 4: Two successive snapshots of Web browser output.

file" to point to the content of place "answer". The transition adjacent to the initially-marked "entry" has timing values of $(0, 0)$. Consequently the token in "entry" is redistributed immediately to places "template", "joke" and "null". The contents for "template" are identified to the system as a template file; the complete template specification is shown in Figure 3. Place "null" is used to modify the behavior of the specification and does not have displayable content associated with it. Note that the author of the template file has decided to specify the appearance of the link associated with transition "answer", but has used the defaults for the transitions "next" and "again", which are adjacent to place "answer". Figure 4, left, shows the Web browser display when the net is in this state.

When the "answer" link is selected, the tokens from "joke" and "null" are removed, and tokens placed into "answer" and back into "joke". Consequently the joke's text remains visible, but the "answer" link is no longer selectable, as shown in Figure 4, right. At this point, the reader can decide to re-visit the current joke (restoring the initial display at the left of the figure) or can see the next joke in the sequence. If the reader selects the "next" link, a token is deposited into place "exit" and the other tokens removed from the subnet. Because "exit" is mapped in the top-level net, a token also appears in place "joke2" at this point, causing the invocation of the next instance of the subnet.

Note that each instance in this example uses the *same* definition of structure while mapping distinct *content* elements. Note further that in a separate application, it could be the content that is reused, with an element appearing in multiple structural contexts. This strong separation of structure and content permits the separate adjustment of either.

3. CAUSATIVE FACTORS AND APPLICATION EXAMPLES

Factors that make it desirable to change the reading environment of a hypertext dynamically include those involved with a reader's characteristics (e.g., the reader's expertise, experience, job role, etc.) or actions (e.g., what portions of the hypertext have already been visited and other characteristics of the browsing history), other readers' characteristics or actions (e.g., interaction with other readers, presence or absence of classes of readers such as desk assistants, etc.), features of the hardware/networking environment (e.g., screen size and resolution, network speed and cost), features of the real-world environment (e.g., time, location, speed, proximity to other objects, etc.), and policy decisions made by the hypertext's managers (e.g., access limitations, etc.). See [10] for more detail. A response to changes in these factors is a modification of the hypertext's browsing behavior—a perceived change to the hypertext's browsing semantics.

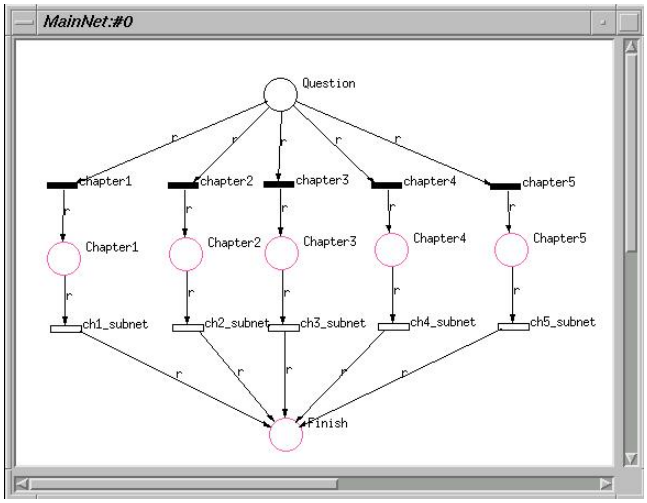

Figure 5: The top-level net.

In reaction to these factors, the browsing behavior of a responsive hypertext may be modified. Modifications may be actual—a change to a value binding or to the structure—or they may be virtual—modifications to the apparent behavior of the hypertext, perhaps as the result of evaluation of conditional expressions that include settable attribute values or perhaps through substitution of a parallel segment of the specification. Modifications may involve the hypertext's content, its link structure, and/or the hypertext's presentation.

Perceived changes to the content may achieve several effects. Most directly, variants of the same content may be presented—for example, information in different languages, versions specialized for different experience backgrounds, or versions prepared for presentation on different media. In most cases of this sort the selection, once made, will not change during the browsing session. More dynamically, content may be modified during the browsing session. For example, the information presented on the first version might be more detailed than the version encountered subsequently (e.g., a tutorial description that is replaced by a summary). Alternately, the information can become more detailed on subsequent visits or additional information might be disclosed. Perhaps in such cases the substitution reflects the time between visits—e.g., the original version is shown when greater amounts of time elapse between visits. An additional possibility may be that the content varies in less obviously predictable fashion between visits—for example, a different quote might be displayed on each visit to a portion of a hypertext.

Perceived link structural modifications may be modest in scale, achieving results similar to content changes, or may result in large-scale change to the hypertext's browsing behavior. A straightforward appearing modification is the addition or removal of portions of the structure (i.e., enabling or disabling links). For example, additional help links could be added on repeated visits to a portion of the hypertext. In administering a quiz, portions of the quiz could become unavailable after a time period elapses or after the student moves on to a subsequent section. Substitution of "isomorphic" structures (e.g., structures covering essentially the same content) may be an appropriate response to meet the goals of readers with different interests, different training, or different job functions. As with content modifications, seemingly arbitrary modifications of structure may be appropriate for some applications with seemingly different organizations encountered on subsequent visits.

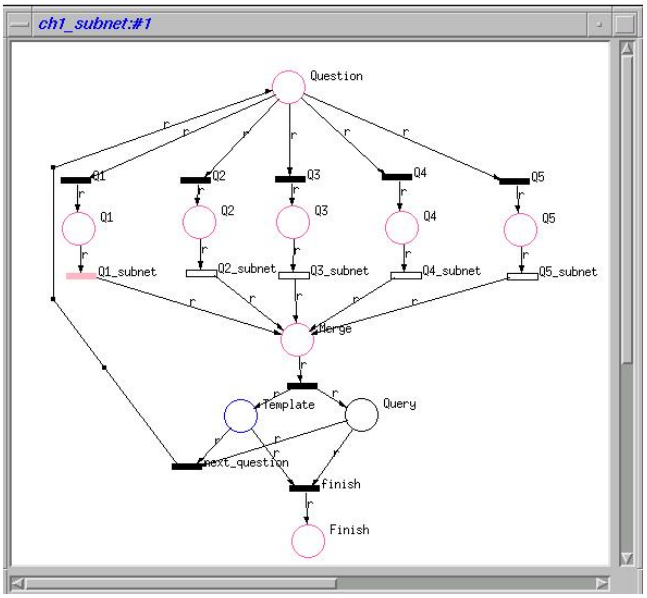

Figure 6: Chapter one's subnet.

Modification of the presentation of information displayed to a reader during browsing presents a third category of potential responses to the causative factors. Content may be rendered for different devices (e.g., shown as printed text on displays or converted into synthesized speech for audio devices) and surface features of the formatting may be modified (e.g., fonts used in display). If the displayed node is actually represented in the specification as a composite without temporal ordering dependencies among the elements (e.g., a picture and its caption; a procedure call signature and the free-text description of its function). In this case the relative ordering of the different information elements can be altered if desired. (This may be perceived by the reader to be a content change, not a presentational change.) Link appearance may also be modified—for example, links may be shown associated with anchors in the text, or as separate menu items either within the content display or separated from the display. Choice among certain link behaviors may also be considered to be presentational in nature. For example, the three link types defined in Guide[1] [3] represent distinct presentational responses to link following.

3.1 Example one: quiz application

Consider a multiple-choice quiz application intended for invocation by other nets, such as those presenting a tutorial. In the example, we assume that the tutorial covers five topics and that for each topic five questions are to be asked. Figure 5 shows the top-level net; place "Question" is the entry point. Reader access to a set of questions is permitted only if the corresponding topic has been read. This is kept track of in the value of a variable carried by the token and is enforced through conditional statements associated with the top row of transitions.

Assume that the reader has chosen to answer questions for chapter 1. Figure 6 shows the subnet corresponding to the chapter's questions (since all chapters have the same number of questions, the same subnet specification can be used for all). Here the reader can select any of the previously unasked questions to answer. The

[1]The three types are replacement (essentially Nelson's Stretchtext), note/glossary (a pop-up display), and reference (the conventional behavior in which the destination replaces the source).

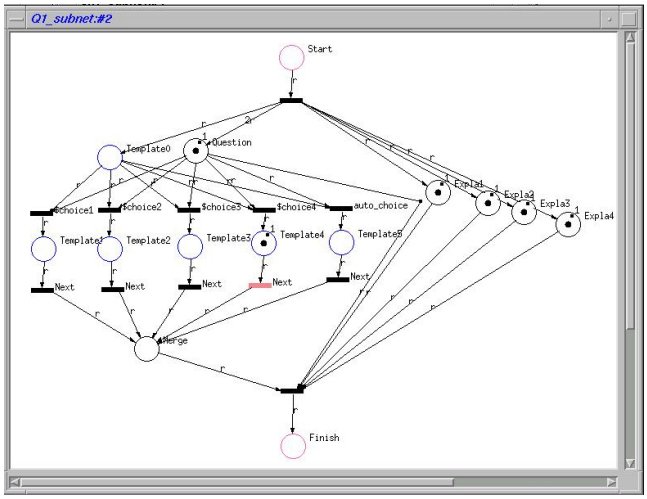

Figure 7: Question one in chapter one.

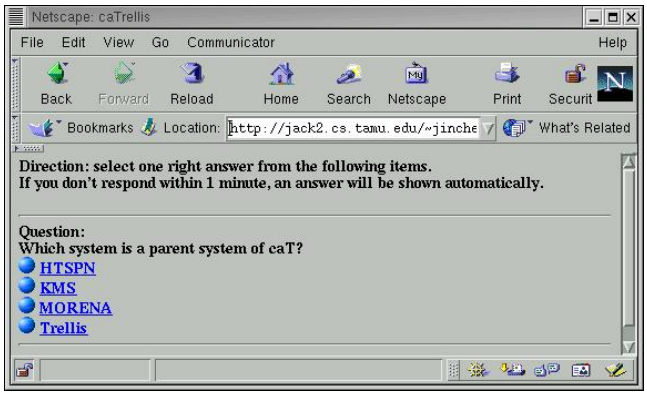

Figure 8: Asking the question.

structure is similar to that of Figure 5, using a token variable to successively restrict access. While questions remain to be answered, the reader is presented with the "next_question" link. When all questions have been answered, the "finish" link is fired automatically to take the reader back to the top-level net.

Figure 7 shows the subnet associated with a question. Again, there are four possible answers. Figure 8 shows the display when the question is first posed; this is specified by the template associated with place "Template0". In the design, the correct answer is found as the fourth choice—this does not necessarily correspond to the fourth link because the links are alphabetized by the system since the *link_sort* attribute's value is specified to be "yes" in the associated `<Place>` construct. The labels displayed for the links are set by reference to a variable, hence the specification of $choice1 through $choice4 on the transitions in the net. The link labeled "auto_choice" is invoked automatically if the reader does not respond within one minute.

By chance in the example, the fourth link does correspond to this correct choice, and the reader has selected it, resulting in the Web display of Figure 9. If the reader had selected an incorrect answer, a display like Figure 10 would be generated instead. Note that these displays are created by selecting an appropriate template—the question and the explanation of the four choices (represented by the four places on the right hand side of Figure 7) remain marked while the subnet is active. The different templates specify whether

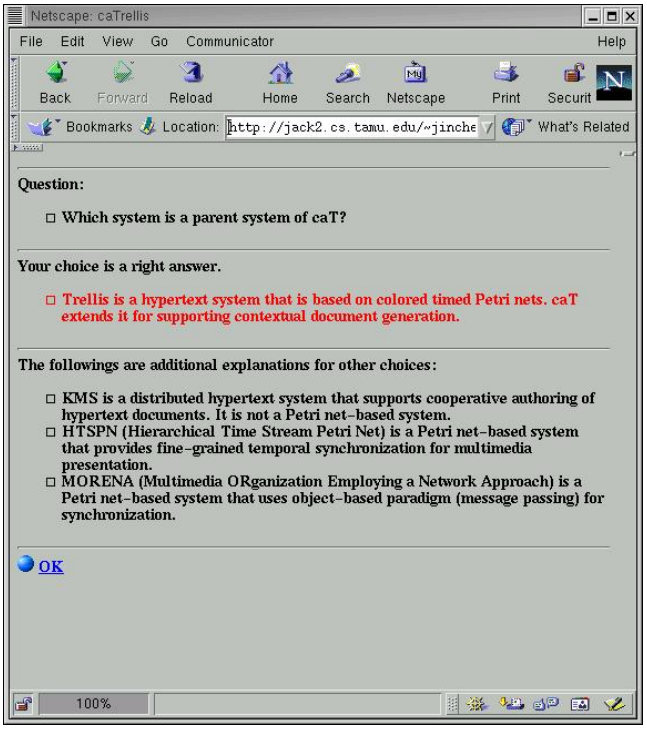

Figure 9: Correct answer.

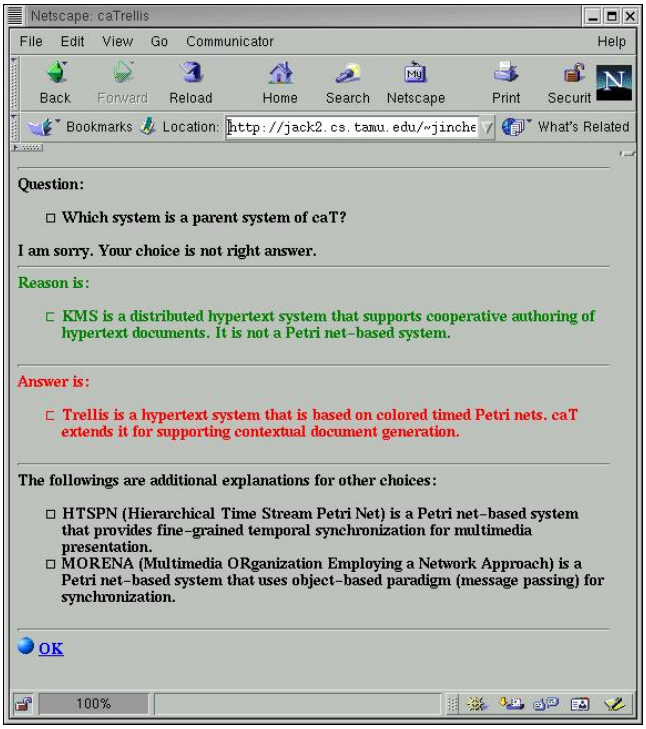

Figure 10: Incorrect answer!

the answer is correct or incorrect and present the explanations in an appropriate order and with necessary connective text.

3.2 Example two: access policies

A characteristic of caT's programmable browsing semantics is that policy decisions concerning issues such as access can be encoded into the net's specification rather than into the system's implementation. Figure 11 provides a brief illustration of this. The figure sketches the specification for four different access policies as an illustration of the generality that can be achieved—certainly others are realizable as well. Example (a) is straightforward—no limits are placed on access. In example (b), no access is permitted. Example (c) allows exactly two visits and blocks off access after that point. Finally, example (d) prohibits access on the first three visits but permits free access after that point. This is because the transition with the $(0,0)$ timing will fire immediately when it is enabled, effectively preventing the firing of the transition leading to the "Resource" place.

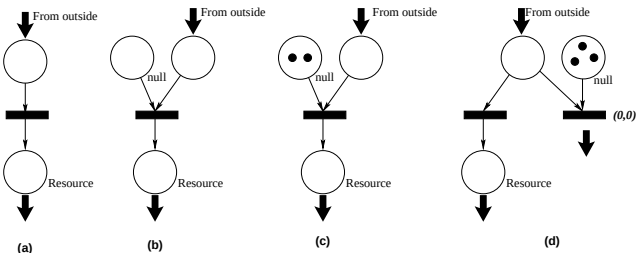

Figure 11: Access protocol specification examples.

3.3 Example three: dynamically-interacting readers

A third application illustrates the interaction of multiple readers. We have reimplemented and extended the Trellis meeting protocol described earlier in [11]. The protocol defined the interactions of a group of meeting participants—permitting only one to speak at a time but allowing a moderator to preempt a participant immediately, representing arrival and departure of meeting participants, and allowing the moderator to exchange roles with a participant. The implementation demonstrated how to separate the roles of readers in the different classes (i.e., the moderator and the meeting participants in this example); allowing multiple listeners simultaneous access to the net, while restricting speaking and moderating roles to a single individual. The new implementation illustrates caT's abilities to provide different Web-based interfaces for each class reflecting the operations available to the reader and to refine further the interactions that occur between the meeting participants. Extensions include the addition of a voting function to the protocol and the ability to reflect the results of votes in the state of the net. In addition, "aging" is added to the protocol—for example, a participant who has just finished speaking can be given lower priority for gaining the floor again immediately, to allow other participants a better opportunity to obtain permission to speak.

3.4 Notes and comments

The examples have sketched ways in which caT hypertexts can be specified to respond to user, environmental, and use factors. We conclude the section by touching on some additional points.

caT has inherited Trellis' client/server implementation architecture. In this architecture, the clients make the determination of how to best render content elements for presentation to the reader. For example, the χTed Petri net editor presents a net view, while the standard χTrellis browser `xtb` presents a multi-window textual view. Specialized clients may decide to present only a single media type, for example graphics or audio.

Multiple clients may be active simultaneously, communicating with a server. If several clients are associated with the same display, this means that each will present its interpretation of the content to the reader simultaneously—perhaps providing a debugging view in addition to the browsing view, or perhaps providing simultaneous textual and synthesized speech renderings of the content. A reader who does not like the Web-based views presented in this paper can continue to use `xtb`, or indeed can use both interfaces at the same time.

The implementation architecture provides a natural mechanism by which separate readers can interact through a hypertext—each reader has an independent client that communicates with a common server. It also provides a means by which multiple displays/computers can be used by an individual reader—separate clients for each of the displays/computers. Since clients determine the content rendering, specializing that rendering for the specific output device characteristics (e.g., large-sized and small-sized displays) is a natural extension. One of our current research projects is to specify a mechanism by which clients can negotiate, when requested by the reader, to avoid redundant rendering of content (e.g., multiple essentially identical views of the same content element). In the future, we expect that such negotiations will continue to increase in complexity to provide more effective information displays; see, for example, issues such as those identified in [14].

As a second comment, we note that the syntax of the template file continues to evolve. For example, it is easy to see how to specify Guide's [3] replacement and reference link semantics (mentioned earlier) using a combination of net structure and template file semantics. However, with the current system, Guide's note/glossary link cannot be specified. Both note/glossary and replacement links would share the same net state in caT, as both source and destination are visible simultaneously. Within the caT context, the distinction between the two link types is presentational, not structural. Consequently, it will be up to the template file specification to allow the hypertext's author to distinguish between the two link types' presentations; this is an extension that we are discussing currently.

Finally we close the section by emphasizing, once again, that the changes to the net's browsing semantics that we have described may be implemented either as an actual modification to the specification or as virtual modifications. In the caT context, a virtual modification is achieved through the mechanisms of net structures implementing access control—access to sections of the net specification is permitted or denied based on the state of the net (i.e., the distribution of tokens in the net).

4. RELATED WORK

In the years following Trellis' introduction, several other teams also adopted automaton-based hypertext models. MORENA [2] and MHPN [27] provide finer-grained synchronization than Trellis in order to describe multimedia documents. HMBS [26, 7] and XHMBS [20] use the Statechart formalism as their basis.

Recently, Muchaluat-Saade and Soares [18] have presented a graphical notation for describing hypertexts that is similar in appearance to Petri nets but without the automaton formalism. A component of their notation that may be an interesting extension to caT's is the inclusion of graphical attachment points, corresponding to anchors in the content. Also of relevance are Ellis, et al.'s long-standing investigations into representing workflows with automaton-based models [8]. A topic of particular applicability to caT in the context of the current paper is their work in identifying feasible transformations to such specifications [25].

Storyspace Guard fields [17] are one example of an early mechanism by which hypertext structures could respond to events that occurred during hypertext browsing. The more recent SMIL standard [23] has provided a mechanism that permits flexible specification of multimedia presentations that can be defined to respond to a user's characteristics.

The design of the template file, which directs the mapping from caT net space to Web display, is straightforward and shares concepts with earlier research efforts in hypertext and similar fields. Within the Hypertext subject area, we note the relationship to earlier work involving composites (e.g., [15]). Also of relevance are investigations of information reuse (e.g., [12]) which contain the concept of presentational transformation of hypertext content. The sources for this concept's incorporation into caT include the investigations carried out in the context of document preparation systems concerning the flexible presentation of structured document specifications [4, 13].

Our caT-related work can be viewed as an alternate view of issues raised in the context of Adaptive Hypermedia; De Bra, Houben, et al.'s nicely-developed framework [6, 5] describes hypertextual adaptations incorporated into such systems, which are similar to those identified in this paper. In addition similar issues have been encountered in other contexts; see for example the earlier investigations into Active Documents (e.g., [1] and [9]). We believe that caT's contribution is complementary to those of these other projects. The perspective of programmable browsing semantics presents the hypertext as a dynamic entity, and one in which it is possible to include considerations such as the coordinated activities of multiple simultaneous readers as well as the effects of independent external events.

5. COMMENTS AND CONCLUSIONS

Our goal in writing this paper was to illustrate the mechanisms in caT that enable the specification of hypertexts that respond to the wide range of factors that occur during their use. In addition to characteristics such as a reader's role (e.g., student, teacher, administrator, or parent) and the reader's browsing history, we also wish to include factors that may not have been incorporated as directly before, such as measures of the external environment and the attributes/actions of other simultaneous readers.

caT's incorporation of the Petri-net-based model introduced in Trellis seems to us to focus attention on the structural characteristics of the hypertext and on the temporal characteristics of its browsing. This tends to encourage the development of nets through definition and reuse of regularly-structured specification fragments rather than through the more organic and ad-hoc structure that results when the hypertext definition focuses primarily on its content.

The hierarchical network extensions supplied by caT makes it realistic to specify hypertexts that are both larger than those in Trellis but also more consistently structured. Increasing size is possible because the specification can be broken down into manageable pieces through the use of subnets. The reuse of those same subnets provides a consistency of specification that was hard to maintain with a single level representation. In general, the incorporation of a hierarchical net structure allows centralization of specifications implementing policies, while at the same time helping to localize the modifications needed if the policies are changed.

We have not described in this paper all of caT's features. In particular, we have only touched briefly on its inclusion of net analysis and debugging tools. caT incorporates standard Petri net analysis techniques that can be used to verify properties of the net through inspections of the states that can be generated. In addition to the analysis of the net's states, caT examines net properties related to its variables through simulation. In addition caT allows capture and replay of browsing sessions, allowing the hypertext's author to

trace through the paths that readers have taken. Such tools provide additional opportunities for hypertext authors to verify the integrity of their hypertext specifications.

Finally, it is interesting to note the wide ranging characteristics of the participants that can take part simultaneously in a browsing session. The participants can be either human or computational in form; interactions may be human–human, human–computer, or perhaps even computer–computer. The wide range of participants and their interests, as well as the breadth of possible environmental factors, may lead to hypertext responses that may not be those preferred by a specific individual. For example, an administrative decision (such as that given in the examples) that limits access to information when outside of normal working hours may not be appreciated by a weekend visitor. caT's mechanisms provide us with the flexibility to decide whether one participant or another should be provided with higher priority in their actions. It will be interesting to continue this exploration of hypertext browsing in response to a wide variety of participants' requests rather than hypertext as responsive primarily to a single individual.

6. REFERENCES

[1] E. A. Bier and A. Goodisman. Documents as user interfaces. In R. Furuta, editor, *EP90*, pages 249–262. Cambridge University Press, Sept. 1990.

[2] R. Botafogo and D. Mossé. The MORENA model for hypermedia authoring and browsing. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 42–49. IEEE Computer Society Press, 1995.

[3] P. J. Brown. Turning ideas into products: The Guide system. In *Proceedings of Hypertext '87*, pages 33–40, Nov. 1987. Published by the Association for Computing Machinery, 1989.

[4] P. Chen and M. A. Harrison. Multiple representation document development. *Computer*, 21(1):15–31, Jan. 1988.

[5] P. De Bra, P. Brusilovsky, and G.-J. Houben. Adaptive hypermedia: from systems to framework. *ACM Comput. Surv.*, 31(4es), Dec. 1999.

[6] P. De Bra, G.-J. Houben, and H. Wu. AHAM: A Dexter-based reference model for adaptive hypermedia. In *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia*, pages 147–156. ACM Press, 1999.

[7] M. C. F. de Oliveira, M. A. S. Turine, and P. C. Masiero. A statechart-based model for hypermedia applications. *ACM Transactions on Information Systems*, 19(1):28–52, Jan. 2001.

[8] C. Ellis. Team automata for groupware systems. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work: The Integration Challenge*, pages 415–424, 1997.

[9] P. M. English, E. S. Jacobson, R. A. Morris, K. B. Mundy, S. D. Pelletier, T. A. Polucci, and H. D. Scarbro. An extensible, object-oriented system for active documents. In R. Furuta, editor, *EP90*, pages 263–276. Cambridge University Press, Sept. 1990.

[10] R. Furuta and J.-C. Na. Applying programmable browsing semantics within the context of the World-Wide Web. In *Proceedings of ACM Hypertext 2002*, 2002. To appear.

[11] R. Furuta and P. D. Stotts. Interpreted collaboration protocols and their use in groupware prototyping. In *Proceedings, ACM 1994 Conference on Computer Supported Cooperative Work*, pages 121–132. Association for Computing Machinery, Oct. 1994.

[12] F. Garzotto, L. Mainetti, and P. Paolini. Information reuse in hypermedia applications. In *Proceedings of the seventh ACM conference on hypertext*, pages 93–104. ACM Press, 1996.

[13] S. L. Graham, M. A. Harrison, and E. V. Munson. The Proteus presentation system. In *Proceedings of the ACM SIGSOFT Fifth Symposium on Software Development Environments*, pages 130–138, Tyson's Corner, VA, Dec. 1992. ACM Press.

[14] J. Grudin. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 458–465. ACM, 2001.

[15] F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Commun. ACM*, 37(2):30–39, Feb. 1994. Edited by Kaj Grønbæk and Randall H. Trigg.

[16] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, volume 1 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1992.

[17] M. Joyce. Storyspace as a hypertext system for writers and readers of varying ability. In *Proceedings of Hypertext '91*, pages 381–387. ACM, Dec. 1991.

[18] D. C. Muchaluat-Saade and L. F. G. Soares. Towards the convergence between hypermedia authoring languages and architecture description languages. In *Proceedings of the ACM Symposium on Document Engineering (DocEng '01)*, pages 48–57. ACM, 2001.

[19] J.-C. Na and R. Furuta. Dynamic documents: Authoring, browsing, and analysis using a high-level Petri net-based hypermedia system. In *Proceedings of the ACM Symposium on Document Engineering (DocEng '01)*, pages 38–47. ACM, 2001.

[20] F. B. Paulo, M. A. S. Turine, M. C. F. de Oliveira, and P. C. Masiero. XHMBS: A formal model to support hypermedia specification. In *Hypertext 98: The Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia*, pages 161–170. ACM, 1998.

[21] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Inc., 1981.

[22] W. Reisig. *Petri Nets: An Introduction*. Springer-Verlag, 1985.

[23] SMIL. Synchronized Multimedia Integration Language (SMIL 2.0) specification, W3C proposed recommendation (05 June 2001), 2001. http://www.w3.org/TR/smil20/.

[24] P. D. Stotts and R. Furuta. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7(1):3–29, Jan. 1989.

[25] C. Sun and C. S. Ellis. Operational transformation in real-time group editors: Issues, algorithms, and achievements. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, pages 59–68, 1998.

[26] M. A. S. Turine, M. C. F. de Oliveira, and P. C. Masiero. A navigation-oriented hypertext model based on statecharts. In *Hypertext 97: The Eighth ACM Conference on Hypertext*, pages 102–111. ACM, 1997.

[27] H. K. Wang and J.-L. C. Wu. Interactive hypermedia applications: A model and its implementation. *Software—Practice and Experience*, 25(9):1045–1063, Sept. 1995.