Exploring the Browsing Semantics of Information on the Web

Cheung, Shing-chi and Lam, Chi-yung

Technical Report HKUST-CS98-14
October, 1998

Abstract
Finding desirable information on the Web can be time consuming and ineffective. The complexity of the Web makes it difficult for users to understand the browsing semantics; the problem is aggravated with the incorporation of dynamic behaviour into information structure using facilities supported by JavaBeans and CORBA. This motivates the need to model HTML documents for rigorous analysis. In this paper, we propose a framework in the Calculus of Communicating Systems (CCS) to unify the modelling of HTML documents based on structure, content and dynamic behaviour. Interesting queries regarding the document, like accessibility, ordering, reachability and mirror site verification, can be answered. The process is automatable via the use of a tool called the Concurrency Workbench (CWB); after the model is inputted to CWB, properties are expressed in the tool environment as modal $\mu$-calculus formulae. CWB can then check automatically whether the model satisfies those properties. Equivalence of two models in terms of behaviour can also be verified. An extended example is used throughout this paper as an illustration.
1 Introduction

The hypertext is the combination of the fragments of the original flat document together with the connections between those fragments, where each fragment is managed in a node and each connection between fragments through a link [1]. With the staggering increase in the popularity for the WWW (World Wide Web), computer users interact with hypertexts, in the form of HTML documents, much more frequently than before. As both the content and the structure of HTML documents become more sophisticated, users and authors alike face the same challenges:

- Starting to navigate from a particular page in the HTML document, can it be made sure that no page with particular content will be accessible?

- Whichever browsing path is chosen, will a certain page be always visited before another page?

- Is a particular page reachable from any page in the HTML document? Is a particular page reachable from the current page via some navigation path?

- Are two HTML documents having the same structure and browsing semantics, i.e. the manner in which information is to be visited and presented to the user through navigation [8]?

As shown from the following respective examples, these challenges are realistic:

- Parents would like to ensure that their children will not be able to get access to pages with pornographic materials starting from their homepage.

- A courseware author needs to make sure that revision materials should precede the quiz in any browsing path.

- A medical professor wants to know whether the conclusion on recent advancement of cancer cure can be reached from any page in his HTML document. Also, it is helpful for a user to know if a page on cancer cure can be visited during web surfing from the page the user is reading.

- While it is not unusual to change the content of the document frequently, an author would like to make sure that the browsing semantics would be kept intact, thereby alleviating the problem of disorientation (situation in which a user is "lost" in the cyberspace during navigation and does not know in what way to make use of the navigation features [1]) when the new document is encountered.

In addition, the incorporation of such new components as Java applets, CGI (Common Gateway Interface) scripts, and RMI (Remote Method Invocation) mechanism, adds dynamic behaviour to such documents. This further complicates the challenges because the structure, semantics and dynamic behaviour of an HTML document are now interrelated. For instance, some interactive HTML documents will lead users to different pages depending on the answer submitted. This will affect the reachability of some pages. Another example is the equivalence of two HTML documents, whose definition is extended to having the same dynamic behaviour as well. This means that both documents behave the same on the same input as far as the user is concerned.

The use of a rigorous model based on which analysis is carried out is helpful in resolving these challenges. HTML documents — in future discussion, HTML documents include dynamic components as well, if any — are first mapped to a model. Then, either properties that we would like the documents to satisfy are expressed in a mathematical form, which
can further be analysed rigorously with the model, or the model itself can be utilised to guide users in the browsing process (say, solving the disorientation problem). From figure 1, we argue that the mapping captures the essential properties of the HTML documents. With the support of mathematical tools, whose correctness are well-established, we claim that the satisfaction or violation of a property in the model infers respectively the possession or lack of the corresponding property in the corresponding HTML document.

Different modelling schemes on hypertext have been proposed. The applicability of such approaches, however, are restricted to hypertext without dynamic components. Our proposed model, on the other hand, is to adopt the formalism of CCS (Calculus of Communicating Systems) put forward by Milner [6]. The most distinguished advantage is that it provides a framework that unifies the modelling of HTML documents, as an information system, based on their structure and semantics, and dynamic behaviour.

Currently, the need for modelling the web-based information system receives little attention. It is our firm belief that people will start realising its importance, when the WWW is gaining its weight in everyday’s life and reliability of such information system is becoming a concern. We understand the worries that the inherited complexity of model construction and property specification may hinder our approach from being widely used. This “barrier” can actually be easily overcome: on the one hand, model construction is easily automatable once clear rules on the mapping from HTML documents to CCS expressions are defined; on the other hand, templates can be made for common categories of properties so that users would only need to supply, for instance, the page that should not be viewed in the accessibility query. As long as we keep the front-end of the system user friendly, users are free from the complexity resided in the back-end, just as in the case of the usage of commercial compilers without knowing the details of the underlying compiler theories.

The remaining of this paper is organised as follows. A brief discussion on some related work is explored in the next section. Then, a case study is presented alongside with the theory of the model in Section 3. Discussion on the analysis of the model will be given in Section 4. Conclusion and future work are mentioned in the last section.

2 Related Work

In [9], hyperdocuments are modelled as an automaton (termed links-automaton) so as to incorporate browsing semantics. A branching temporal logic notation called HTL* (Hyper-text Temporal Logic) is used for specifying properties that should exhibit during browsing. While this model can help verify browsing properties by model checking, the authors ignore the issue of the dynamic behaviour embedded in the hypertext itself: those transition links in the automaton represent hyper-links only. Our model, on the other hand, supports a unifying notation under the umbrella of CCS. Besides, the authors do not provide a way for abstracting the model by means of composition and information hiding, which is very important for web-based information system as the scale is extremely large. CCS is superior in this sense through its operators like composition and restriction for model abstraction.

In [10], the hyperdocuments are modelled
based on *statecharts*, where both the structure and browsing semantics of hypertexts are specified. With this model, synchronisation of simultaneous display can be formulated, and problems like access control, tailored versions and node reachability are solved. The emphasis of this paper, nevertheless, is on navigation: they just consider how actions can help in browsing process (say, programmed learning due to execution of actions associated with transitions). No general modelling of dynamic behaviour is mentioned. Our model, on the other hand, tries to be a general one.

In the field of information retrieval, researchers advocating the confluence of hypertext and information retrieval often see the need for making use of both the structure and content of hypertext. For instance, [3] proposes using *Conceptual Graphs* to represent both structure and content to take advantage from the combination of hypertext and information retrieval approaches. [11] justifies the combined approach that involves content, context, structure and attributes of hypertext for queries and retrieval. Many of these approaches, however, make use of graph models to represent their ideas. Dynamic behaviours may not be easily addressed.

3 Model Construction

To have a successful modelling, rules must be defined to map HTML documents of a web-based information system into CCS notation. In this section, we will illustrate the process through an extended example.

Consider an HTML document written for a Hong Kong Travel Agency. After receiving a greeting message (figure 2), users can choose to learn more about Hong Kong, or to have an on-line quiz (figure 3). If the former link is chosen, a variety of topics can be further pursued (figure 4). Of these topics, there is a page on Hong Kong lifestyle that contains a link to information on horse-racing (figure 5). There is also another page on places of fun in Hong Kong providing links to entertainment activities in Hong Kong. Some link may contain materials not suitable for youngsters, like "Night Life" (figure 6). More details on the document will be unveiled as the discussion goes on.

A brief introduction to CCS, or *process calculus*, is in place. For an in-depth explanation, please refer to [6] and [2]. CCS is a formalism to specify the basic concepts of a reactive and distributed system which consists of *agents* communicating with one another to
achieve system unity. Each agent has a number of states connected by transitions. These transitions, constituting the behaviour of the agent, are defined by actions and operators.

It is, then, straightforward to map fragments of an HTML document as states, and hyperlinks as actions. For each link anchor (i.e. a physical element in the source whose activation will lead the user to the destination of a link), we assign one or more actions to reflect the content information of link destination. This distinguishes our approach from [10] in that the authors do not place emphasis on the annotation of links, while we believe that only by annotating links with semantics will queries on document content itself be carried out. Note that, as exemplified later, actions in our model are not confined to describing content information.

While the alphabet of actions (i.e. the set of all actions) is determined by authors at their own discretion, it is suggested that actions be as orthogonal as possible to facilitate future analysis. Moreover, the use of multiple consecutive actions for an hypertext link is encouraged if such usage retains more information about the document.

With reference to figure 2, the user will reach a page as shown in figure 3 if the anchor Enjoy! is triggered. The HTML document can be modelled as follows, where the system will have a change of state from $A$ (the opening page) to $B$ (the destination page) on action overview (triggering of Enjoy!):

$$A \overset{\text{def}}{=} \text{overview} \cdot B$$

This expression introduces the first operator: prefix operator ($\cdot$). The same expression can be represented in an alternative form called Labelled Transition System (LTS):

$$A \xrightarrow{\text{overview}} B$$

If the author wants to further annotate the destination as hongkong, the corresponding modification is:

$$A \xrightarrow{\text{hongkong \, overview}} B$$

One technical point is mentioned before we proceed. A special state $V$ is introduced to the model. It can be thought as having a “virtual” start page which contains an anchor to the real homepage of the system. In this way, the content of the real homepage can be described. Returning to our example, the following expression can be added based on the discussion so far:

$$V \overset{\text{def}}{=} \text{travelAgent} \cdot A$$

If there is more than one link to choose from a particular page, summation operator ($+$) can be used for modelling. Referring to figure 3, a user can either choose anchor Learn more about HK! or Take a Quiz!. We can express it as follows:

$$B \overset{\text{def}}{=} \text{background} \cdot C + \text{quiz} \cdot D$$

Intuitively, once quiz is triggered due to the choice of anchor Take a Quiz! (say), the
system will behave in accordance with the definition of $D$ (the quiz page; see figure 7).

![Quiz](image)

Figure 7: Online quiz page in our example

The real power of the calculus is revealed when dynamic behaviour is modelled. Suppose we have to submit the answer of a true-or-false question to the web server in the quiz page (figure 7). Based on the correctness of the answer checked by the server, different pages will be presented (see figures 8 and 9).

![Correct Answer](image)

Figure 8: Page shown when the answer is correct

![Wrong](image)

Figure 9: Page shown when the answer is wrong

The behaviour on the client side can be modelled as in figure 10, while one high-level model (with answer-checking mechanism abstracted away) of the server is shown in figure 11. Note that an action can have parameters, as in $\text{submit}(\text{ans})$, where $\text{ans}$ is the answer for the true-or-false question. Also, actions like $\text{correct}$ or $\text{wrong}$ does not refer to any anchor in the document; rather, they are used to represent the dynamic behaviour involved in the interaction between the client and server during the answer-checking process (actions like $\text{correct}$ and $\text{correct}$ are called complementary actions, which serve for the purpose of synchronisation).

![Model for the quiz page](image)

Figure 10: Model for the quiz page

![Model for the server](image)

Figure 11: Model for the server

The incorporation of synchronisation is accomplished by the use of composition operator ($\downarrow$) in several ways. If the server does not store any information for later use by the client, we can simply regard the interaction as a session: the session ends when the interaction is over. With respect to our example, the server is modelled as shown in figure 12. The formulation will then be $D \downarrow D_1|S_2$. Note that a
new instance of $S_2$ is created every time when the quiz page is visited.

If the server stores information for future use (say, keeping the number of questions answered correctly by the user), we would instead formulate as $V \overset{def}{=} travelAgent.A|S_1$. The server is instantiated at the beginning and will keep running. Only one server exists throughout the whole scenario.

Whatever the case, it can be thought of having both the client and server (or server session) running in parallel. After the client receives the action $userInput(ans)$, it emits the action $submit(ans)$. The server (or server session), on receiving this action (as indicated by the complementary action, $submit(ans)$), checks for the correctness and then emits either $correct$ or $wrong$. When the server (or server session) is doing its job, the client cannot proceed but only stop at the dummy state, until it receives either $correct$ or $wrong$.

One important concept is that such synchronised actions will then be hidden after applying the composition operator to become internal action $(\tau)$, which is not observable. This helps abstract away unimportant details of the system: when viewed externally, a user will not notice the presence of such actions as $submit(ans)$ or $wrong$.

If actions are to be only involved in synchronisation, it is a good idea to restrict their occurrences so that they cannot act independently. This is the function of restriction operator $(\setminus)$.

According to [5], multiple concurrent and synchronised browsing streams are desirable in many situations. This feature can actually be modelled using composition and restriction operators, if frames are regarded as the different browsing streams (which is not yet supported in the current HTML specification [4]).

Suppose the author replaces figure 3 with
4 Analysis on the Model

Once the model is formed, we can formulate the desired properties and carry out model checking for verification. In the following subsections, we will use an automated tool called the Concurrency Workbench (CWB) [7] to answer the challenges discussed at the beginning of the article. CWB is a tool that allows users to specify agent behaviour in CCS and formulate property propositions in modal $\mu$-calculus, a particular temporal logic.

Based on the information, CWB can then verify automatically whether the system satisfies the properties by constructing a data structure which represents the system’s state space and then exploring it to check if it satisfies those formulae [2]. It can also check if the systems are equivalent in terms of behaviour.

4.1 Modelling in Action

With reference to the discussion in Section 3, we obtain a list of agent expressions. Since CWB only accepts basic CCS, i.e. expressions whose actions and agents are free from value variables, value expressions and conditions, we utilise one tool mentioned in [2] to first transform value-passing calculus expressions into basic CCS ones, which are then inputted to the CWB environment.

Figure 17 shows a typical session in CWB. Note that command agent specifies an agent expression, and the complementary actions are represented as correct and $\text{'}$correct$\text{'}$.

With the use of daVinci, a graph viewer, a transition diagram of the agent $V$ can be generated automatically and is shown in the Appendix.

4.2 Safety and Liveness Properties Checking

Many properties fall into the following two classes: safety and liveness. A safety property asserts that the system never enters an undesirable state, whereas a liveness property asserts that the system eventually enters a desirable state.

The task now is therefore to transform the queries to modal-$\mu$ logic formulae (refer to figure 17 for the sample interaction session):

- The challenge of whether some page is inaccessible can be modelled as a safety property. So long as no corresponding action appears in the navigation path, the property is preserved. As an illustration, suppose a user wants to know if any indecent materials will be encountered when our extended example is navigated through. Assuming that the action is named $\text{nightlife}$, the formula expressing this property is:

$$\nu X.[\text{nightlife}] \text{fff} \land \lnot X$$

The query should return “false”, which indicates that information on night life
Figure 17: A sample session of CWB for our model

Figure 18: A simulation run to illustrate accessibility of pages on nightlife
is provided via one particular link. We can actually make use of CWB to work out that path (see figure 18).

- Page-ordering constraint problems can similarly be modelled as a safety property. If a user wants to make a query on whether the page on Hong Kong lifestyle always precedes the page on horse-racing (note that we require such an ordering whatever the navigation path), a corresponding formula is shown below, where \( a \) and \( b \) represent actions for links to Hong Kong lifestyle page and horse-racing page respectively:

\[
\nu X. [a] tt \land [b] ff \land [-a, b] X
\]

The query will return "true" based on the model checking. On the other hand, if we modify the document such that access to the page on horse-racing is provided when a user answers the quiz question correctly, the answer to the query should be "false". This shows how the inclusion of a dynamic component affects the result.

- Let us investigate the reachability problem. According to different needs, we have different reachability properties. In the following discussion, reachability of the quiz page is explored.

Case 1: If we just need to make sure that the quiz page is reachable via some navigation path from the starting page, we can make use of the safety property for inaccessibility by making a negation:

\[
\nu X. [a] tt \land [b] ff \land [-a, b] X
\]

violating the inaccessibility property implies that the target is reachable, and vice versa. The property is formulated as \( \Phi \), where

\[
\Phi = -\nu X. [\text{quiz}] ff \land [-] X
\]

Evidently, the result of the query should return "true".

Figure 19: A simulation run to illustrate reachability of quiz page (part 1)

Case 2: Suppose we need to ensure that the quiz page is reachable (via some navigation path) regardless of where the current location is. The property can be expressed as:

\[
\nu Y. \Phi \land [-] Y
\]

where \( \Phi \) is the expression defined in the last case. Note that the query returns "true". As shown in figures 19 and 20, the quiz page can still be reachable via the action \text{history} even when other anchors like Hong Kong Lifestyle was selected earlier, because there is always an anchor that leads back to the page entitled “Explore Hong Kong”.

\[\text{It is not required that } a \text{ or } b \text{ must eventually occur; liveness property is involved if we add the requirement that } b \text{ should eventually occur once } a \text{ occurs. The corresponding logic formula is:}
\]

\[
\nu X. [a]\text{even}(b) \land [b] ff \land [-a, b] X
\]

where \( \text{even}(b) \) is defined as:

\[
\mu Y. (-) tt \land [-b] Y
\]
Case 3: If we have to make sure that the quiz page is reachable (via any navigation path) from the starting page, liveness property comes to play. The corresponding formula is $\Phi'$, where

$$\Phi' = \mu X.(-)tt \land [-quiz]X$$

The query returns “false” since a user can repeat navigating on pages related to background of Hong Kong without the need to choose the anchor to quiz page.

Case 4: This is the strictest of all cases. Suppose the quiz page has to be reachable (via any navigation path) no matter where the current location is, the formula should be:

$$\nu Y.\Phi' \land [-]Y$$

where $\Phi'$ is the expression defined in the last case. Obviously, if the query for the previous case fails, this query should return “false” also.

Before we end this subsection, a note on user manipulation of such formulae is in place. As observed from the examples, these formulae share a similar structure. In other words, they can be categorised from which a common pattern can be derived. One possible way of easing the burden of users in manipulating these expressions is to parameterise the formulae such that users only need to select from a list of formulae and then supply the necessary information.

### 4.3 Mirror Site Checking

One of the important features of CCS is that of observation equivalence, which means that two agents are indistinguishable as far as observable behaviours, i.e. behaviours perceivable by an observer performing experiments on the agents, are concerned. This is in contrast with strong equivalence, a much more
flexible notion in which two agents are indistinguishable even when unobservable behaviours (in other words, τ actions) are taken into the account.

This feature assists in checking the equivalence of the two HTML documents: as long as the structure and the semantics that specify the theme of the content do not change, it does not matter if the content details are changed. In addition, as long as the dynamic components in their respective documents are sharing the same interface and interaction sequences, the two documents are regarded as equivalent.

As an illustration, suppose that an old version of our extended example HTML document used Javascript to implement the quiz section. The author would like to know if the model is changed after replacing the Javascript with CGI scripts involving client/server interactions. This can be done by checking for observation equivalence between the new and old models. In CWB, the verification can be carried by command \texttt{eq(Agent1,Agent2)}.

5 Future Work and Conclusion

We are planning to realise the model through Extensible Markup Language (XML): by extending the document type of HTML, documents on the WWW can be readily analysed as mentioned in the previous section.

Some other interesting issues to be investigated are mentioned below:

- **Information Retrieval**: Current successful commercial search engines largely rely on keyword search solely, say, boolean retrieval, without utilisation of the structure of hypertext. The use of such features will improve the relevance of retrieved documents. What stands out more is that our model can even handle queries on dynamic behaviour of hypertext (a sample query is: find all documents that are shown due to failure in a question).

- **Electronic Commerce**: Various protocols and interactions are involved in the transaction. Once the behaviour is modelled in CCS, it can be plugged into the model and a lot of interesting queries on the transaction (rather than the protocol itself) can be made. For instance, an on-line publishing company implements a service that a user can only read a certain number of new pages for each transaction payment. It is helpful if the user can check whether the particular information s/he wants to read can be accessed under that constraint. Equally useful is the ability to calculate how much a user has to pay in order to access the materials s/he wants.

- **User Map**: One of the main problems in hypertext navigation is user disorientation. As this model is hierarchical, we can provide users with some kind of high-level road map to aid the browsing process.

- **Incorporation of Non-determinism**: Stochastic process algebra can be applied to our current model. One possible scenario is to model the actual load of a web server and the network traffic. Further interesting queries, like “find the fastest route towards a particular node”, can then be made.

We believe that our approach is promising as it provides one single framework for further development. On the one hand, plain hypertext, in its general sense, can be modelled using our approach without the loss of power when compared to other approaches discussed in Section 2. On the other hand, dynamic behaviours in HTML documents, a special type of hypertext, can be modelled seamlessly. With the model, lots of interesting issues concerning the document can be explored.
References


Appendix

The labelled transition graph for our example is shown as follows.