

# Using Maude for the Formal Verification of Websites\*

Sonia Flores Salvador Lucas Alicia Villanueva

sflores@dsic.upv.es slucas@dsic.upv.es villanue@dsic.upv.es

DSIC, UPV

DSIC, UPV

DSIC, UPV

## Abstract

In this paper we address the problem of formal verification of websites by using declarative languages. In particular, we first define a model for websites which can intuitively be specified by using Maude. The model is defined to be well-suited for the formal verification of dynamic as well as static properties of the system. A website is defined as a collection of web pages which are semantically connected in some way. External web pages (which are related pages not belonging to the website) are treated as the *environment* of the system. We also present the temporal logic which is used to specify properties of websites, and illustrate the kinds of properties that can be specified and verified by using the Maude model-checking tool. Finally, we have implemented a prototype by using the specification language Maude.

## 1 Introduction

Internet is an essential component of the modern Information Society. It provides easy and flexible access to information and resources distributed all around the world. There is a big conceptual distance from its origins in the sixties (as a governmental, restricted way to exchange sensible data and information) to its current use as a kind of public service, open to everybody, open to everything. The development of the *Hypertext Markup Language* (HTML) [17] and the *Hypertext Transfer Pro-*

*ocol* (HTTP) in the 90s led to a happy marriage between wires, waves, and software components (*the Web*) which can be thought of as the main developments driving this change.

The development of websites (which can be often understood as sets of HTML documents) is an important task in modern software engineering. As expected in any software project, the web developers or designers must guarantee that the system (i.e., the website) satisfies some particular requirements. For instance, they might want to ensure that some private resources are only available to a given set of registered users. Or they could want to guarantee that some information (held as HTML documents in the site) is reachable by any user from any point in the site.

To this end, developers can apply some validation and verification techniques and tools. Formal methods have proved their usefulness to guarantee (beyond any doubt) that the behavior of a software system corresponds to what has been specified. Formal methods try to mathematically establish provable connections between the formal description of the system (i.e., its semantics) and the properties of interest. Regarding the web, the underlying programming language is (some variant of) HTML and the first problem is to devise a suitable semantics which fits the desired abstraction level and also enables the use of some logic and verification algorithms to express and check the properties. Viewed as software systems, websites can be understood as *reactive* or *interactive* systems [18]. Model checking [6], which is a well-known technique for verifying *reactive* and *interactive* systems provides an appropriate framework for dealing

---

\*This work has been partially supported by the EU (FEDER) and the Spanish MEC under grants TIN2004-7943-C04-02, HA2006-0007, and Valencian Government under Grant GV06/285

with such kind of systems.

In this paper, we propose a model (in terms of a directed graph) which is suitable for the verification of many interesting properties of web sites. We implement it in Maude [7] since Maude is a specification language able to represent in a very intuitive way graphs and transitions between nodes. Moreover, a model-checking tool able to verify LTL formulas is available for Maude specifications [8]. We show how to encode our website model in Maude. Being an essential aspect of the Web, the proposed model focuses on the information in the HTML sources concerning *connectivity*. Following the *semantic web* approach, however, we also consider *annotations* [10] which can also complement the information associated to each node of the website. Such annotations can be thought of as *abstractions* of the (contents of the) HTML document. We show how to verify properties which are specified by using a Linear Temporal Logic (LTL [15]). We also show how to combine all of these notions to verify very interesting properties on the Web. If the output from the model checker is *yes*, then we are sure that the system satisfies the property. Otherwise, a counterexample is provided showing a trace (or path in our case) in which the property is *not* satisfied. This counterexample can be very useful for detecting the error.

The paper is organized as follows. Section 2 is a brief introduction to some notions related to the Web that are used along the paper. In Section 3, the formalism used to model websites is presented and motivated. Thereafter, in Section 4 the formalism for the specification of properties is presented. We describe the verification approach that we have implemented to verify such properties in Section 4.2. In Section 5 we discuss the future extensions that can be integrated in our framework. Related work is discussed in Section 6, and Section 7 presents our conclusions.

## 2 Introduction to Websites

In this work, we model web pages and their connections as a directed graph whose nodes

represent web pages and whose edges represent links among web pages. We restrict ourselves to the analysis of a website (which plays the role of the *system* in classical approaches to formal verification). Here, a website is a collection of web pages which are semantically related in some way and hosted at a single machine.<sup>1</sup> We can easily extend the notion of system by considering, for instance, a set of websites instead of a single one. In that case, a preprocess of the source documents is mandatory in order to adapt the notions, but the contents of this paper would remain valid. It is important to note that our focus is not on analyzing the entire Web, but rather on a specific part. Therefore, we represent the *environment* (i.e., other parts of the Web) as a single special node. This fact does not limit the applications of the framework since, due to the flexibility of the notion of system, it is possible to analyze huge parts of the Web.

Usually, each web page is defined by means of a document, typically specified in HTML, XHTML or XML markup languages. The examples shown along this paper assume that the specification language is XHTML, but the definitions can be parameterized w.r.t. any specification language with a similar expressiveness. Each web page could contain links to other web pages. *The source of the link* is the source document where a link is specified, whereas the document where a link points to is called *the destination of the link*.

In order to be able to define a link to a specific point within a source document, a label (or anchor) to such specific point must be defined in the source document. This label will later be used in a link to specify the destination point in a web page. Links and labels are the only information about the structure of a web page that our model will take into consideration. For example, the relative position of a specific content or link is abstracted from.

Being the basis for the definition of the kind of links our model will deal with, let us recall the definition of URI [17]. Intuitively, an URI is the description of where the source docu-

---

<sup>1</sup>This assumption does not restrict the applicability of the approach as we will show later.

ment of a web page is hosted. It consists of three components:

1. the mechanism used to access the resource: usually `http`, `https`, or `mailto`,
2. the name of the machine hosting the site,
3. the name of the resource itself, described as a path in the hosting machine.

We represent the URI of a document as a structure of the form `uri(mech,host,resource)` where  $\text{mech} \in \{\text{http}, \text{https}, \text{mailto}\}$ <sup>2</sup>, `host` is a string defining a domain, and `resource` is a string representing the path at the host where the source document can be found. When the string `resource` contains the character `#` followed by a label, then the URI does not refer to the top of a web page, but to a point within the source document. Let us show an example:

**Example 1** *The string*

`http://www.w3.org/2002/ws/sawsdl/#Publications`

*refers to a document accessible by using the http protocol which is hosted at `www.w3.org`. The navigator accesses the source document by following the path `2002/ws/sawsdl/` at the host, and it shows the section identified by the anchor `#Publications`.*

The example above shows how an absolute URI can be defined. It is also possible to define relative URIs where the only described information is the path where the destination document is hosted. Note that we can specify the access to a web page in the website both by using a relative and an absolute URI. In this paper we assume that, whenever possible, URIs are specified in its relative form. This assumption can be overcome by defining a classification method for links depending on the considered system.

The way in which links are specified depends on the specification language. Typically, links and anchors are defined by means of the `LINK` and `A` elements. In order to specify the different characteristics of the link or anchor, some attributes can be associated to these elements. Here we focus on the `name` and `href` attributes.

<sup>2</sup>For the purposes of this work, only `http` and `https` values are considered.

`name` is associated to the `A` element in order to define a label (anchor) to a specific point of the document that can later be used in a URI. `href` can be associated to both `A` and `LINK` elements, and contains the URI where the destination document is located.

Finally, our model considers the information regarding how the resource is being loaded: in the same navigator window, in a different one, in the same frame, etc. This information is specified by using the argument `target`. In this work we consider the following possible values for that argument: `Target = {_blank, _self, _parent, _top}`.

### 3 The Website Model

In any formal verification process, one of the first tasks to be performed is the definition of the model that the verification algorithm will handle. In this section, we present the model for websites, and its representation in Maude. Recall that a website as described above is a collection of web pages. Our model is defined in terms of a directed graph: each node represents a web page (a source document), whereas each edge represents a link specified in the source node and pointing to the destination node of the edge. In other words, a link connects two ends (web pages) and has one direction [17].

In the following we define the different components of our model. First of all we describe how nodes of the graph are formed. Later we define each component of the node.

**Definition 1 (Node)** *Let  $u$  be the URI associated to the web page  $s$ . The node representing  $s$  is defined as the tuple  $\text{PageSkeleton}(s) = \langle u, \text{lab}, \text{loc}, \text{site}, \text{ext}, \text{Sem} \rangle$  where  $\text{lab}$  is the set of labels (anchors) defined in  $s$ ,  $\text{loc}$  is the set of local links defined in  $s$ ,  $\text{site}$  is the set of site links defined in  $s$ ,  $\text{ext}$  is the set of external links defined in  $s$ , and  $\text{Sem}$  is the set of annotations representing the semantic web information of the web page  $s$ .*

As one can observe, our model can represent three kinds of links: local links, site links and external links. A local link relates a web page

with itself by using a label (anchor) defined in the same document. A site link connects web pages from the same website, i.e., web pages that share the hosting machine and that semantically are related. Finally, an external link connects a specific web page to any other resource not belonging to the website. We encode this definition in Maude as follows:

```
op <_,_,_,_,_> :Uri LabSet LocLnkSet SiteLnkSet
  ExtLnkSet PSSemSet -> WebPage .
```

The set of labels (anchors) that are specified along a source document is defined as follows:

**Definition 2 (lab)** *Let  $s$  be a source document, we denote as  $lab(s)$  the set of values of attributes `name` and `id` specified for elements `A` or `LINK` in  $s$ .*<sup>3</sup>

The corresponding Maude encoding is:

```
op _ _ : Lbls Lbls -> Lbls [assoc comm id:empty] .
op empty: -> Lbls .
op {}: Lbls -> LabSet .
```

To model a local link we need to specify the label where the link points to, and the target:

**Definition 3 (Local link)** *Let  $u$  be the URI of the web page  $s$  and `Target` the set defined above. The pair  $\langle l, t \rangle$  denotes a local link specified in  $s$  whose destination resource is  $s$  itself,  $l \in lab(s)$  and  $t \in Target$ .*

We do not need to specify the mechanism, host and resource path of the document since they coincide with the ones of  $s$ . We represent this set in Maude as follows:

```
sort LocLnk LocLnkSet .
subsort LocLnk < LocLnkSet .
op empty:LocLnk .
op _ _ :LocLnk LocLnk -> LocLnk [assoc comm id:empty]
op |_|:LocLnk -> LocLnkSet .
op <_,_,_>:Lbls Target -> LocLnk .
```

Assume that we have the following code at some point along a document  $s$

```
<a name="Participants">
  The list of the Working Group's participants </a>
```

<sup>3</sup>This definition establishes a direct relation between the model and the specification language. If a different specification language were used, the definition should be parameterized w.r.t. the new language.

defining an anchor to that point of the web page ( $Participants \in lab(s)$ ). The following local link could appear at any other point of  $s$

```
<a href="#Participants"> Participation</a>
```

For a site link, the tuple has an additional component: the path from which the document can be retrieved. The auxiliary function `doc`, given a URI `uri(mech,host,res)`, retrieves the document associated to it. For example, if we had the URI `uri(m,h,r)`, then `doc(m,h,r) = d` denotes that  $d$  is the source document associated to such URI.

**Definition 4 (Site link)** *Let  $uri(m,h,u)$  be the URI of document  $s$  ( $doc(m,h,u) = s$ ). The tuple  $\langle p, l, t \rangle$  denotes a site link specified in  $s$  where  $doc(m,h,p) = d$ ,  $l \in lab(d) \cup \{''\}$  and  $t \in Target$ .*

Let us show the encoding in Maude:

```
sort SiteLnk SiteLnkSet Path .
subsort SiteLnk < SiteLnkSet .
op {}:SiteLnkSet -> SiteLnkSet .
op empty:SiteLnkSet .
op _ _ :SiteLnk SiteLnk -> SiteLnk [assoc comm id:empty]
op <_,_,_>: Path Lbls Target -> SiteLnk .
```

The specification of the anchor is non compulsory. We represent this case by using the value  $l = ''$ .

Finally, we assume that external links point to resources hosted at any machine.

**Definition 5 (External Link)** *Let  $uri(m,h,u)$  be the URI of the web page  $s$ . Then the tuple  $\langle m', h', p, l, t \rangle$  denotes an external link specified in  $s$  where  $doc(m',h',p) = d$ ,  $l \in lab(d) \cup \{''\}$ , and  $t \in Target$ .*

The corresponding Maude encoding is

```
sorts ExtLnk ExtLnkSet Mech Host Path .
subsort ExtLnk < ExtLnkSet .
op empty:ExtLnkSet .
op _ _ :ExtLnk ExtLnk->ExtLnk [assoc comm id:empty]
op <_,_,_,_,_>:Mech Host Path Lbls Target->ExtLnk .
op {}:ExtLnkSet -> ExtLnkSet .
```

An example of external link and its corresponding representation is shown below:

```
<link href= "http://www.dsic.upv.es/users/elp/elp.html"
  target=_blank>
```

```
 $\langle m', h', p, l, t \rangle = \langle "http", "www.dsic.upv.es",
  "users/elp/elp.html", "", _blank \rangle$ 
```

Note that the extension of the framework to deal with a set of hosts is straightforward. We simply should need a preprocess to adapt the notions of local and external link.

The last component of nodes in the graph models web semantics, i.e., the semantics associated to the web page, and that ideally are defined in the source document. In our model, we represent the semantics associated to a web page by using a set of pairs. Pairs consist of two components: the name of the attribute (type) and its assigned value (cont) where the assigned value typically is a list of keywords.

**Definition 6 (Page Semantics)** *The semantics of a webpage  $s$  is defined as the set  $Sem$  of pairs  $\langle type, \{cont\} \rangle$ .*

Let us show the Maude encoding:

```
sorts TypeAtt PSem PSet Sem SemSet .
subsort PSem < PSet .
subsort Sem < SemSet .
op empty:->PSet .
op _ _:PSem PSet -> PSet [assoc comm id:empty] .
op {_:}PSet PSet -> PSet .
op <_,{_:}>:TypeAtt SemSet ->PSem .
```

Next we show an example of meta data interpreted as part of the semantics of the web.

```
<meta name= "keywords"
  content= "Web Services,DL 2,
  WSDL 2.0,WS-CDL 1.0,WS-Addressing,
  Web Services Addressing,annotation"/>
```

This information will be included in the  $Sem$  set of the node as the pair:

```
<(type,{cont}) = <"keywords",{ "Web Services,WSDL 2.0,
  WS-CDL 1.0,WS-Addressing,
  Web Services Addressing,annotation"}>
```

Up to this point, we have defined how nodes of the graph are formed. Next we define edges linking nodes. In order to show how edges are defined, we need an auxiliary function which, given a link, retrieves the node corresponding to the destination document of such link.

**Definition 7 (Auxiliary function node)**

*We call  $N$  to the set of nodes in the graph and  $Web$  a special node representing all the nodes outside the system. Given a mechanism  $m$ , a host  $h$ , and a path  $p$ ,  $node(m,h,p)$  returns*

- $n \in N$  s.t.  $n = \langle u, lab, loc, site, ext, Sem \rangle$  and  $u = (m, h, p)$ ,

- or the node  $Web$  in case there exists no node in  $N$  with URI  $(m,h,p)$ .

The case in which node returns value  $Web$  corresponds to the case when the link is an external link.

Now we can define the set of edges:

**Definition 8** *For each  $n, n' \in N$  of the form  $n = \langle u, lab, loc, site, ext, Sem \rangle$  and  $n' = \langle u', lab', loc', site', ext', Sem' \rangle$ , we define the set of edges  $E$  for the graph as follows:*

1.  $(n, n, l, t) \in E$  for each  $\langle l, t \rangle \in loc$  if  $l \in lab$ ,
2. For each  $\langle p, l, t \rangle \in site$ ,  $u = (m, h, p)$ , if  $node(m,h,p) = n'$  and  $l \in lab'$ , then  $(n, n', l, t) \in E$
3. For each  $\langle m, h, p, l, t \rangle \in ext$ , if  $node(m,h,p) = Web$ , then  $(n, Web, l, t) \in E$

Finally, we formally define the graph structure representing the system:

**Definition 9 (Website Model)** *Given a set of documents  $P$  forming the website  $S$ , we define the model of  $S$  as  $SiteModel(S) = \langle N, E \rangle$  where  $N$  is the set of  $PageSkeletom(p)$  for each  $p \in P$  and  $E$  is the set of edges as defined in Def. 8 on  $N$ .*

```
subsort WebPage < WebPageSet .
op empty:->WebPageSet .
op _ _:WebPageSet WebPageSet->WebPageSet
  [assoc comm id:empty] .
op {_:}WebPageSet -> WebSite .
```

The set  $E$  is implicitly specified in the transition relation defined in Maude. Such transition implements the different links among edges in order to travel the web pages.

### 3.1 An Illustrative Example

In Fig. 1 we can see the webpage associated to the URI [www.w3.org/2002/ws/](http://www.w3.org/2002/ws/).

The model obtained for the website associated to such page is partially shown in Fig. 2.

Next we show the components of the node representing the above webpage which in the graph is depicted at the middle of the figure:

```
<(uri(http,"www.w3.org", "/2002/ws/"),
 {"News","Events","Wiki and tools","Groups",
  "Documents","Technical","drafts","discussion","related"}
 {"News",_self}, {"Events",_self}, {"Wiki and tools",_self},
 {"Technical",_self}, {"Documents",_self}, {"Groups",_self})>
```

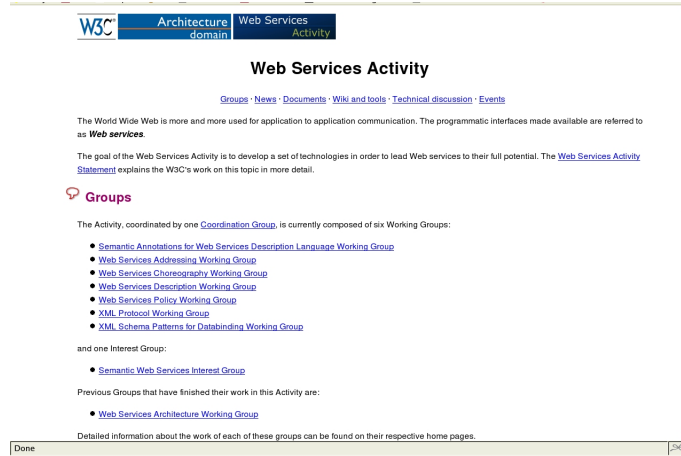


Figure 1: Web Page of W3C Web site

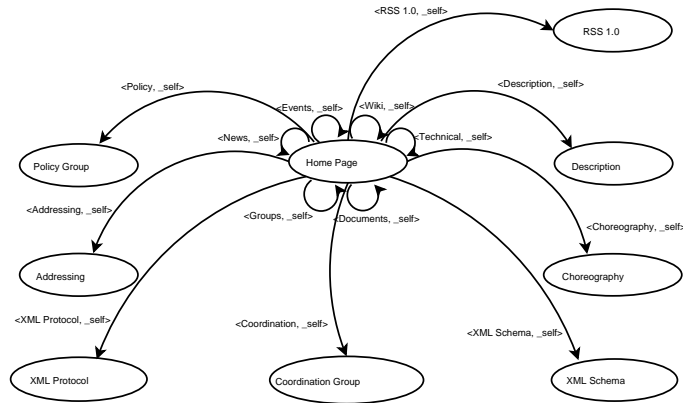


Figure 2: Model of `www.w3.org/2002/ws/`

```

{"../../../../sawSDL",_self}, {"../../../../cg",_self},
 {"../../../../chor",_self}, {"../../../../addr",_self},
 {"../../../../desc",_self}, {"../../../../policy",_self},
 {"../../../../databinding",_self}},
 {"http://search.w3.org/2000/xp/group",_self}},
 {"keywords",{"Web Services","WS-Addressing",
 "WS-Addressing 1.0","Semantic","annotation"}},
 {"revision",{"$Id: Overview.html",
 "v 1.230 2007/01/29 18:08:38 ylafonExp$"}}
}

```

#### 4 Verification of Properties

Let us now show the formal framework for the specification of properties. We use the well-known Linear Temporal Logic (LTL) [15] for the specification of properties. Next we recall

the syntax of LTL formulas:

$$\varphi ::= p \mid (\neg\varphi) \mid \varphi \wedge \varphi \mid \varphi \mathcal{U} \varphi \mid \mathcal{G}\varphi \mid \mathcal{X}\varphi$$

Where  $p$  is an atomic proposition,  $\mathcal{U}$  is the *until* operator,  $\mathcal{G}$  is the *always* operator, and  $\mathcal{X}$  is the *next* operator. Other classical connectives can be defined in terms of the above ones as usual, in particular  $\varphi \rightarrow \psi \equiv (\neg\varphi) \vee \psi$ .

The semantics of the logic are given in terms of a Kripke Structure which can also be seen as a directed graph whose nodes are labeled with the atomic propositions true in such nodes. Formally, given a set of atomic propositions  $AP$ , we denote as  $M = (S, S_0, T, L)$  a Kripke

Structure where  $S$  is the set of states,  $T \subseteq S \times S$  is the transition relation between states,  $S_0 \subseteq S$  is the set of initial states, and  $L$  is a labeling function from  $S$  to the  $\wp(AP)$ . A sequence of states  $\pi = s_0 \cdot s_1 \cdot \dots$  is called a path of  $M$  whenever  $(s_i, s_{i+1}) \in T$  for all  $i \geq 0$ . When  $s_0 \in S_0$ , then we say that  $\pi$  is an initial path of the system.  $\pi^i = s_i \cdot s_{i+1} \cdot \dots$  denotes the suffix of the sequence  $\pi$  starting at the position  $i$ th. Note that  $\pi^0 = \pi$ . The semantics of LTL formulas is defined as follows:

$$\begin{aligned} \pi \models p &\Leftrightarrow p \in L(s_0) \\ \pi \models \neg\varphi &\Leftrightarrow \pi \not\models \varphi \\ \pi \models \varphi \wedge \phi &\Leftrightarrow \pi \models \varphi \text{ and } \pi \models \phi \\ \pi \models \mathcal{X}\varphi &\Leftrightarrow \pi^1 \models \varphi \\ \pi \models \mathcal{G}\varphi &\Leftrightarrow \text{for all } i \geq 0, \pi^i \models \varphi \\ \pi \models \varphi \mathcal{U} \phi &\Leftrightarrow \exists j \geq 0 \text{ and } \forall 0 \leq i < j, \\ &\quad \pi^i \models \varphi, \pi^j \models \phi \end{aligned}$$

A Kripke Structure  $M$  satisfies a given formula  $\varphi$  ( $M \models \varphi$ ) if and only if for each initial path  $\pi$  of  $M$ ,  $\pi \models \varphi$ . In the following we show the correspondence between the model defined in the previous section and a Kripke Structure.

The relation between the two formalisms is straightforward except for the labeling function, i.e., the function that determines which propositions are satisfied in each node. Nodes and edges of both formalisms coincide,<sup>4</sup> this, the temporal operators of the logic can be interpreted on the edges of our model. Regarding the labeling function  $L$ , we define it (and also the set of atomic propositions  $AP$ ) in terms of the information stored in each node of our model, i.e., the semantics of the web, the URIs, and the links.

We think that one of the most interesting things of our model is the fact that it allows reasoning about the web semantics. For this reason, next we formalize how this information is included in the Kripke Structure. Given  $n \in N$  a node of the graph, we call  $n'$  to the corresponding node in the Kripke Structure. Let  $Sem$  be the indexed set (with index  $I$ ) in the last component of the tuple  $n$  (i.e., the representation of the semantics of the webpage).  $L(n') = \bigcup_{t \in I} Sem_t|_2$  being  $Sem_t$  the  $t$ th element

of  $Sem$ .  $|_2$  denotes the projection on the second component of the pair  $Sem_t$ .

#### 4.1 Specification of properties

Temporal logic allows specification of properties such as safety properties (ensuring that something bad never happens) and liveness properties (ensuring that something good eventually happens). These properties are related to the infinite behavior of a system. We check properties related to paths among webpages. For example, it is important to guarantee that private webpages are available only to a given set of registered users. The next example models the property:

**Example 2 (Registered User)** *Let  $S$  be a website and  $SiteModel(S) = \langle N, E \rangle$  its corresponding model. Assume that the semantics of nodes contains information about the privacy of webpages. In particular, there exists a type scope whose values can be any of `{public,private,access}` meaning respectively that the page can be accessed by any user, only by registered users, and a public page used to register users.*

*The property that establishes that a private page can only be viewed by registered users is defined in LTL as follows:*

$$\neg(\text{public } \mathcal{U} \text{ private})$$

*Note that we can use the values of the semantics in the formula thanks to the definition of the labeling function in the Kripke model.*

Other properties that can be checked are whether a webpage has any link to itself, or whether there exists a webpage reachable by a direct link from any other webpage of the website. This last property is specially interesting since we could infer some information regarding the structure of the website from it. The node reachable from any other node by a direct edge could be representing the *HomePage* of the website. Let  $\phi$  be a characterization of a given webpage. The formula

$$\mathcal{G}((\neg\text{web}) \rightarrow (\mathcal{X}\phi))$$

<sup>4</sup>Labels on the edges of the graph can be integrated into the labeling of Kripke Structures by means of well-known transformation techniques.

is satisfied whenever all the nodes except the one representing the environment have a direct link to the webpage  $\phi$ .

Finally, a relevant property that is commonly checked in the literature (for example [11]) is whether every node in the website is reachable from another one. We can statically check whether the URI of each node is used in, at least, one node different from itself, but this would not ensure that every node is reachable from any other. To solve this problem we should verify for each node of the model whether it is reachable from an initial state. In particular we should verify that the formula

$$\neg(\text{init } \mathcal{U} \phi)$$

is **not** satisfied<sup>5</sup> assuming that for every initial state  $s$ ,  $\text{init} \in L(s)$  and that  $\phi$  identifies the state we are analyzing.

#### 4.2 The prototype

We have encoded the graph representing the model of the system in Maude [7]. The current version of our prototype parses HTML webpages to models (as defined along this paper) encoded in Maude. As we are dealing with LTL properties, we can use the Maude model checker [8] to verify the above mentioned properties on the website model. Maude is a specification language based on rewriting logic with some features that make it suitable for the encoding of this kind of systems. We have modeled the graph and defined the transition rules that move from one webpage to another.

As the experiments with our prototype have provided very interesting results, we think it is worth extending the system in several ways. First of all, we plan to develop parsers for other markup languages (not only HTML).

The features regarding the flexibility of the notion of system can be integrated in the prototype. We plan to implement the pre-process which considers the webpages' URIs for determining the kind of each link, and whether pages belong or not to the site. Since the source document can be written in any

<sup>5</sup>In LTL, to check if there exist a path satisfying the property  $\text{init } \mathcal{U} \phi$ , we have to use the equivalent strategy that checks whether the negation is not satisfied (for all path).

markup language, the transformation method must be defined for each language commonly used by programmers and designers.

## 5 Extensions of the framework

The guide [16] establishes some principles for the easy and intuitive navigation inside a website. The rules that the guide proposes regard the style of navigation elements in terms of the size and the architecture of sites. The guide enumerates some properties that cannot be specified by using the LTL logic, in particular those for which quantification is needed. For example properties related to the cost (number of steps) to reach a given node. In order to be able to check such kind of properties, a real time temporal logic is needed, i.e., not a qualitative but a quantitative temporal logic where counting the number of time instants is possible. Several real temporal logic have been previously defined in the literature [4, 5, 9], but in order to use them, the model checking algorithm of Maude must be adapted. Therefore, an interesting extension of our framework is to consider such kind of properties and adapt a model-checking algorithm to deal with the new logic and with our model. Note that, in any case, the model presented in this paper needs not to be modified.

We plan also to consider, not only quantifying on the number of steps, but also on the number of paths. Properties regarding the number of links pointing to one page, or the number of links defined in a document can be very useful for a web designer, for example to restrict the number of links defined in (to) a webpage. To be able to check such kind of properties, we need to quantify on the number of possible paths thus a branch temporal logic able to count branches should be needed.

Finally, we think that one of the most interesting extensions that can be done is to enrich the behavior of the method by defining heuristics for inferring information from the structure of the website (see the HomePage property described in the previous section). The inferred information could be added to the semantics of the model such that more sophisticated properties could be checked.

## 6 Related work

Defining formal models for the Web is not a new topic. In [1], a model for a website able to capture information about links and frames of webpages was defined. Then the model-checking technique was applied. In fact, [1] is the first example of how model-checking techniques can be adapted to the verification of the Web. In particular, special attention is paid on how to avoid the infinite component intrinsic to the Web nature.

Our proposal is different from [1] in the sense that, first of all, we define a different, more precise model for the formal verification, and second, we use the traditional temporal logic for model checking. More in particular, we restrict the search space by restricting the analysis to a finite system (a website). In [1], this abstraction is made at the logic level. Another example on how to restrict the search space is found in [11], where LTL properties are defined to be checked w.r.t. a subset of states modeling an excerpt of the Web. A second important difference is the kind of properties we are able to verify, implied by the fact that our model captures different information from the source documents of the site.

In [14], a different formalism is used in order to check the web. Term rewriting techniques are used in order to model the dynamic behavior of the Web. Then, properties regarding reachability can be verified under some restrictions. These restrictions are imposed due to the decidability results of reachability for the different term rewriting theories. Our approach improves the one in [14] since we are able to verify more expressive properties by using an effective algorithm. Finally, note that [14] uses Maude as a specification model for rewriting theories whereas in this work, we use Maude to specify the defined graph model.

The rule-based approach has also been used in [2, 3] where static properties regarding both syntax and semantics of the web are verified. A new technique inspired in declarative debugging algorithms has been developed in order to check errors regarding correctness as well as completeness of websites w.r.t. a given specification. The main difference between this ap-

proach and the framework presented in this paper is that we are interested in dynamic properties that can be specified by using temporal logics and that can be checked by using a model-checking algorithm.

Finally, other models for the web have been defined along the years such as the different versions of the random graph models [12, 13]. The principal aim of these works is not to apply formal methods to the analysis or verification of websites. Random graph models are useful in order to *measure* the web in the sense that they try to model the web and perform a number of search algorithms that counts, for example, the number of links that contains a webpage, or the number of links that points to a given webpage. The numeric results are analyzed in order to identify clusters regarding a topic, *hub* webpages, etc. However, our main purpose is the verification of properties by applying formal methods, in particular model-checking algorithms, thus the model must differ. Note that it is possible to quantify and *measure* the web by using quantitative logics such as the real time logics as mentioned in Section 5. In conclusion, we are able to analyze different properties by using a unified formalism: temporal logics, whereas in [12, 13] different algorithms must be run in order to study different aspects of the web. Finally, we note that some of the search algorithms described in these works abstract the environment of the system similarly as we do.

## 7 Conclusions

In this paper we have defined a new model for websites which is more precise and flexible than other approaches. In our framework, the modeled and analyzed system can be a single website intended as a collection of webpages semantically related, but also a set of websites. We have demonstrated that the model captures enough information to apply formal methods such as model checking or inference of properties. We have established a concrete relation between our model and Kripke Structures, what makes possible to apply the model-checking technique to the Web. The most important issue in such rela-

tion is the definition of the labeling function of the Kripke Structure. In particular, we have shown how the labeling function must be defined to handle web semantic information of websites allowing us to check properties related to the semantic information webpages.

We have shown the kind of properties can be specified and verified by using the LTL logic, and which logics can be considered to specify more sophisticated properties. We have also described the possible extensions of our framework, that we plan to abord as future work.

## References

- [1] L. de Alfaro. Model Checking The World Wide Web. In *Proc. of 13th Int'l Conference on Computer Aided Verification*, vol. 2102 of *LNCS*, pp. 337–349, 2001.
- [2] M. Alpuente, D. Ballis, and M. Falaschi. Rule-based Verification of Web Sites. *Software Tools for Technology Transfer*, 8(6):565–585, 2006.
- [3] M. Alpuente, D. Ballis, M. Falaschi, P. Ojeda, and D. Romero. A Fast Algebraic Web Verification Service. In *First Int'l Conf. on Web Reasoning and Rule Systems*, to appear in *LNCS*, 2007.
- [4] M. Alpuente, M.M. Gallardo, E. Pimentel, and A. Villanueva. A Real-Time Logic for tcp verification. *J. of Universal Comp. Science*, 12(11):1551–1573, 2006.
- [5] R. Alur and T.A. Henzinger. A really temporal logic. *Journal of ACM*, 41(1):181–204, 1994.
- [6] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, Cambridge, MA, 1999.
- [7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The maude 2.0 system. In *Proc. RTA'03*, vol. 2706 *LNCS*, pp. 76–87, 2003.
- [8] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL Model Checker. In *Proc. of 4th WRLA*, vol. 71 of *ENTCS*. Elsevier Science, 2002.
- [9] E. A. Emerson and R. Trefler. Parametric Quantitative Temporal Reasoning. In Giuseppe Longo, editor, *Proc. of 14th Annual IEEE Symp. on Logic in Computer Science*. IEEE Press, 1999.
- [10] S. Handschuh and S. Staab. *Frontiers in Artificial Intelligence and Applications - Annotation for the Semantic Web*. IOS Press, 2003.
- [11] M. Haydar, S. Boroday, A. Petrenko, and H. Sahraoui. Properties and Scopes in Web Model Checking. In *Proc. of 20th IEEE/ACM Int'l Conf. on Automated Software Engineering*, pp. 400–404, 2005. ACM Press.
- [12] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: Measurements, models and methods. *LNCS*, 1627:1–17, 1999.
- [13] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Ufal. The Web as a graph. In *Proc. 19th ACM SIGACT-SIGMOD-AIGART Symp. Principles of Database Systems*, pp. 1–10. ACM Press, 2000.
- [14] S. Lucas. Rewriting-based navigation of web sites. In *Proc. of 1st Int'l Workshop on Automated Specification and Verification of Web Sites*, vol. 157(2) of *ENTCS*, 2005.
- [15] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems - Specification*. Springer-Verlag, New York, 1992.
- [16] F. Millerand and O. Martial. Guide Pratique de Conception et d'évaluation Ergonomique de sites Web. Tech. Rep. 25, Centre de Recherche Informatique du Montreal, 2001.
- [17] D. Raggett, A. Le Hors, and Jacobs I. *HTML 4.01 Specification*. W3C, MIT, Inria, Keio, 1999.
- [18] Klaus Schneider. *Verification of Reactive Systems. Formal Methods and Algorithms*. Springer-Verlag, Berlin, 2004.