MCWEB: A Model-Checking Tool for Web Site Debugging*

Luca de Alfaro Thomas A. Henzinger Freddy Y.C. Mang
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, Berkeley, CA 94720-1770, USA
{dealfaro,tah,fmang}@eecs.berkeley.edu

ABSTRACT

We show how model checking techniques can be applied to the analysis of connectivity and cost-of-traversal properties of Web sites.

1. INTRODUCTION

The design of a Web site is an inherently error-prone process. A Web site must be correctly designed both at a *local,* and at a *global* level. Good design at the local level implies that the pages contain well-formed HTML code, have the intended visual appearance, and have no broken links. In addition to these properties, that are local to individual Web pages, the structure of a Web site must satisfy global properties concerning its connectivity and cost of traversal. Another class of global properties of Web sites concerns the use of frames. Since each link loads only a portion of a frame-based page, the content of a frame-based page depends on the path followed by the browser in the site. Indeed, in a frame-based site there are many pages that can be reached only by following a specific sequence of links: we call such pages *secondary pages,* to distinguish them from the *primary pages* that can be reached simply by typing an URL into the browser window.

Current tools for Web site debugging focus mainly on local properties, and in particular, on the detection of broken links and errors in the HTML code [8, 9, 6, 10, 4, 7, 3, 12, 13, 5]. The only global analysis possible with these tools consists in displaying a map of the Web site; such maps are however of limited utility for large sites. Moreover, current tools explore only primary pages in frame-based sites, while our experience indicates that the greatest number of errors occurs in secondary pages — most likely because they are more difficult to check exhaustively without automated assistance. The difficulty of debugging all secondary pages has hindered the use of frames in high-reliability applications, such as commercial Web sites. To help in the analysis of global properties of Web sites, we have implemented a checker called MCWEB (*Model Checking the Web*), that relies on *model checking techniques* to verify connectivity, cost-of-traversal, and frame-dependent properties of Web sites.

2. MODEL CHECKING THE WEB

Model checking is the name of a set of techniques that check the correctness of a system design by exploring the graph corresponding to the state space and transition structure of the system [2]; these techniques have been effectively applied to the debugging of hardware designs. To cope with the complexity of such systems, model checking has developed a set of techniques suited for the exploration of very large graphs, that are not given explicitly, but must be gradually explored. Moreover, model checking has developed a rich theory for the specification of graph properties [2]. MCWEB applies model checking techniques to the analysis of the graph formed by Web pages and their links. The properties that can be checked using MCWEB include the following:

- **Connectivity structure.** Given a collection of Web pages, MCWEB can check that all the pages of the collection are reachable from a specified start page. Similarly, MCWEB can check that none of a specified set A of "private" pages is reachable from a set B of "public" pages. The above reachability and unreachability properties are just two examples of a general class of connectivity properties that can be checked by MCWEB. Another example consists in checking that every path from pages of a set A (the "entry" pages) to pages of a set B (the "members" pages) must contain a page in a set C (the "authorization" pages). MCWEB provides a rich query language for the specification of connectivity properties.

- **Frame structure.** MCWEB also checks for errors in the frame structure of Web pages. A common error occurs when more than one sub-frame is labeled by the same name, making the use of the `target` link attribute ambiguous. A related error is for the same URL to be loaded into two hierarchically nested sub-frames. MCWEB also checks for links that specify a `target` that does not correspond to any frame name. These errors generally occur in secondary pages, since link traversal can cause unforeseen combinations of pages to be loaded into the sub-frames of a page; an example of such errors is given in Figure 1.

- **Longest paths.** MCWEB can compute the longest and shortest paths between sets of Web pages, where the "length" of a path can be measured either as the number of links in the path, or as the number of bytes that a browser must download while following the path. In particular, the all-pair longest-path between pages of a site can provide useful information about bottlenecks in the navigation of the site.

3. THE QUERY LANGUAGE

Model-checking is based on the analysis of a graph structure. We model the Web as a graph with

Poster Proceedings of the 10th World-wide Web Conference, 2001.

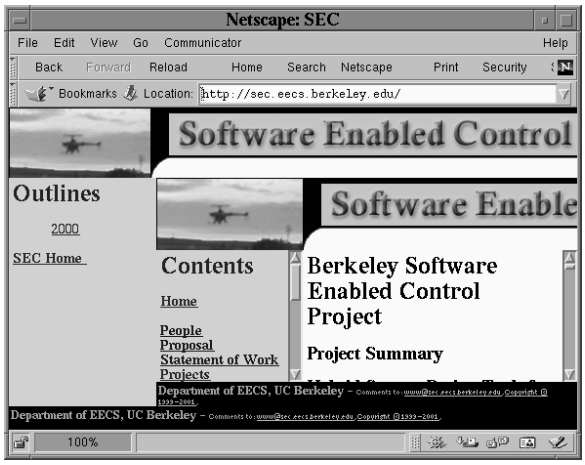

Figure 1: A secondary Web page with repeated frame names, and nested sub-frames. This anomalous configuration was detected by MCWEB.

webnodes for vertices. A webnode is a hierarchical frame structure, generated by the grammar $webnode ::= URLpage\,(name\ webnode)^*$, where an *URLpage* is the result of fetching a given URL from the Web with a GET method, and each pair (*name webnode*) consists of the name of a subframe, and of the subframe content. The edges of the graph correspond to links between Web pages; the destination webnode is obtained by updating the frame structure as specified by the HTML standard [11]. Taking webnodes, rather than URLpages, as vertices of the graph enables an accurate representation of the frame structure of pages. Moreover, since webnodes correspond to pages as displayed by a browser, they lead to a natural connectivity analysis of the Web.

The model checking algorithms are phrased in terms of operations applied to sets of webnodes; this is similar to the approach of *symbolic model checking* [2]. Operating on sets of nodes is efficient, since all the links from a set of webnodes can be followed in parallel by a multi-threaded implementation. If a is an URL, and S, T are two known sets of webnodes, we can perform the following operations:

- $GetWeb(a)$ returns the singleton set consisting of the webnode corresponding to URL a.
- $\mathrm{Post}(S)$ returns the set of webnodes reachable by following one edge from webnodes in S, and $\mathrm{Pre}(T, S)$ returns the subset of webnodes in T that have an edge leading to S.
- For any webnode property P, we can compute the subset $S \mid P = \{s \in S \mid s \models P\}$ of webnodes that satisfy P. The webnode properties considered by MCWEB include text inclusion, sub-frame structure well-formedness, and errors in fetching URLpages.
- We can compute $S \cap T$, $S \cup T$, and $S \setminus T$.

MCWEB enables us to write queries that combine the above operators in a fixpoint calculus called *μ-calculus* [1]. For example, consider the property that all paths from a set A of public pages to a set B of private pages must contain a page in a set C of authorization pages. Assume that the pages in A, B, C can be distinguished by predicates P_A, P_B, and P_C (based e.g. on text inclusion). Let h be the URL of the home page of the Web site, and let P_D be a predicate that is satisfied only by pages of the site being debugged. Then, the following query with output z computes the webnodes that violate the above property:

$$\mu x = GetWeb(h) \cup (\mathrm{Post}(x) \mid P_D)$$
$$\mu y = (x \mid P_A) \cup (\mathrm{Post}(y) \mid (P_D \wedge \neg P_C))$$
$$\mu z = y \mid P_B$$

Here, x, y, and z are variables corresponding to sets of webnodes. Each line computes the smallest set of webnodes that satisfies the given equality. Thus, the first line computes the set x of webnodes that are reachable in the site from the home page. This set is computed iteratively by setting $x_0 = \emptyset$ and $x_{k+1} = GetWeb(h) \cup (\mathrm{Post}(x_k) \mid P_D)$ for $k \geq 0$. The computation terminates when we reach $n \geq 0$ such that $x_{n+1} = x_n$, at which point we let $x = x_n$. Similarly, the second line computes the set y of webnodes reachable in the site from $x \cap A$ without visiting C. Finally, z consists of $y \cap B$, and hence, of the webnodes in B that can be reached in D from A without visiting C — which is the desired answer. MCWEB is written in Python, and makes heavy use of multi-threading. Furthermore, MCWEB employs model checking techniques to optimize query evaluation.

4. REFERENCES

[1] G. Bhat and R. Cleaveland. Efficient model checking via the equational μ-calculus. In *Proc. 11th IEEE Symp. Logic in Comp. Sci.*, pages 304–312, 1996.

[2] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.

[3] Electronic Software Publishing Co. Linkscan. http://www.elsop.com/linkscan/.

[4] Watchfire Co. Linkbot. http://www.watchfire.com/products/linkbot.htm.

[5] Voget Selbach Enterprises GmbH. Link tester. http://vse-online.com/link-tester/.

[6] Tilman Hausherr. Link sleuth. http://home.snafu.de/tilman/xenulink.html.

[7] Biggbyte Software Inc. Infolink. http://www.biggbyte.com/infolink/index.html.

[8] Link Alarm Inc. Link alarm. http://www.linkalarm.com/.

[9] NetMechanic Inc. Html toolbox. http://www.netmechanic.com/.

[10] InContext. Web analyzer 2.0. http://www.incontext.com/WAinfo.html.

[11] D. Raggett, A. Le Hors, and I. Jacobs. HTML 4.01 specification, 1999. W3C Recommendation 24 December 1999.

[12] Internet Software Services. Theseus. http://www.matterform.com/theseus/.

[13] DACPro Computer Solutions. Webtester. http://awsd.com/scripts/webtester/.