VERDI: an Automated Tool for Web Sites Verification*

M. Alpuente[1], D. Ballis[2], and M. Falaschi[2]

[1] DSIC, Universidad Politécnica de Valencia, Camino de Vera s/n, Apdo. 22012, 46071 Valencia, Spain. Email: alpuente@dsic.upv.es.
[2] Dip. Matematica e Informatica, Via delle Scienze 206, 33100 Udine, Italy. Email: {demis,falaschi}@dimi.uniud.it.

Abstract. VERDI is a system for the automated verification of Web sites which can be used to specify integrity conditions for a given Web site, and then automatically check whether these conditions are actually fulfilled. It provides a rule-based, formal specification language which allows us to define syntactic/semantic properties of the Web site as well as a verification facility which computes the requirements not fulfilled by the Web site, and helps to repair the errors by finding out incomplete/missing Web pages.

1 Introduction

The increasing complexity of Web sites calls for tools which are able to aid Web designers in the construction and maintenance of Web sites. Systematic, formal approaches can bring many benefits to quality Web sites development, giving support for automated Web site verification [2]. In [3] a formal, declarative verification algorithm is proposed, which checks a particular class of integrity constraints concerning the Web site's structure (syntactic properties), but not the contents (semantic properties) of a given instance of the site. The framework XLINKIT [2] allows one to check the consistency of distributed, heterogeneous documents as well as to fix the (possibly) inconsistent information. Its specification language is a restricted form of first order logic combined with Xpath expressions [8]. This paper presents the prototype VERDI (Verification an Rewriting for Debugging Internet sites) which provides a rule-based language for the specification and the verification of syntactic as well as semantic properties on collections of XML/HTML documents. The prototype is based on the theoretical framework we proposed in [1] and enjoys the effectiveness of rule-based computation.

We use rewriting-based technology both to specify the required properties and to formalize a verification technique, which is able to check them.

* This work has been partially supported by MCYT under grants TIC2001-2705-C03-01, HU2003-0003, by Generalitat Valenciana under grant GR03/025 and by ICT for EU-India Cross Cultural Dissemination Project under grant ALA/95/23/2003/077-054.

```
<members>
   <member status="professor">
        <name> mario </name>
        <surname> rossi </surname>
   </member>
   <member status="technician">
       <name> franca </name>
       <surname> bianchi </surname>
   </member>
</members>
```

```
members(
  member(status(professor),
         name(mario),
         surname(rossi)
  ),
  member(status(technician),
         name(franca),
         surname(bianchi)
  )
)
```

Fig. 1. An XML document and its corresponding encoding as a ground term.

2 Denotation of Web Sites

In our framework, a *Web page* is either an XML[7] or an HTML[6] document. Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of a given term algebra as shown in Figure 1. Note that XML/HTML tag attributes can be considered as common tagged elements, and hence translated in the same way. Therefore, *Web sites* can be represented as finite sets of (ground) terms.

3 Web specification language

Web specifications formalize conditions to be fulfilled by a given Web site. Roughly speaking, a Web specification is a finite set of rules of the form $\mathtt{l} \rightharpoonup \mathtt{r}$, where $\mathtt{l}$ and $\mathtt{r}$ are terms. Some symbols in the right-hand sides of the rules may be marked by means of the symbol $\sharp$. Marking information of a given rule r is used to select the subset of the Web site in which we want to check the condition formalized by r. Intuitively, the interpretation of a rule $\mathtt{l} \rightharpoonup \mathtt{r}$ w.r.t. a Web site $\mathtt{W}$ is as follows: if (an instance of) $\mathtt{l}$ is recognized in $\mathtt{W}$, also (an instance of) $\mathtt{r}$ must be recognized in $\mathtt{W}$.

In the following we present an example of Web specification.

Example 1. Consider the following Web specification, which models some required properties of a research group Web site containing information about group members affiliation, scientific publications and personal data.

$$\mathtt{hpage}(\mathtt{status}(\mathtt{professor})) \rightharpoonup \sharp\mathtt{hpage}(\sharp\mathtt{status}(\sharp\mathtt{professor}), \mathtt{teaching})$$
$$\mathtt{member}(\mathtt{name}(\mathtt{X}), \mathtt{surname}(\mathtt{Y})) \rightharpoonup \sharp\mathtt{hpage}(\mathtt{name}(\mathtt{X}), \sharp\mathtt{surname}(\sharp\mathtt{Y}), \mathtt{status})$$
$$\mathtt{pubs}(\mathtt{pub}(\mathtt{name}(\mathtt{X}), \mathtt{surname}(\mathtt{Y}))) \rightharpoonup \sharp\mathtt{member}(\mathtt{name}(\mathtt{X}), \sharp\mathtt{surname}(\sharp\mathtt{Y}))$$

First rule states that, whenever a home page of a professor is recognized, then that page must also include some teaching information. Here, for instance, marks are used to check the condition only on professor home pages. Second rule formalizes the following property: if there is a Web page containing a member list,

then for each member, a home page exists containing (at least) the name, the surname and the status of this member. Finally, the third rule specifies that, whenever there exists a Web page containing information about scientific publications, each author of a publication should be a member of the research group.

4 The verification technique

Our verification technique allows us to verify a Web site w.r.t. a given Web specification in order to detect incomplete and/or missing Web pages. Moreover, by analyzing the requirements not fulfilled by the Web site, we are also able to find out the missing information which is needed to repair the Web site. Since reasoning on the Web calls for formal methods specifically fitting the Web context, we developed a novel, rewriting-based technique called *partial rewriting* [1], in which the traditional pattern matching mechanism is replaced by tree *simulation* [4] in order to provide a much more suitable mechanism for recognizing patterns inside semistructured documents, which is (i) independent of the order of the tagged elements contained in a Web page and (ii) able to efficiently extract the partial information of a Web page we need to check.

Basically our verification technique works in two steps. Given a Web site `W` and a Web specification `I`, we first compute a set of requirements (that is, information that should be present in the Web site) by generating the set of all possible Web pages that can be derived from `W` via `I` by partial rewriting. Then, we check whether the computed requirements are satisfied by `W` using simulation and marking information. Requirements which are not fulfilled allow us to detect missing or incomplete Web pages and provide the information which is necessary to fix the Web site.

Example 2. Consider the Web specification `I` of Example 1 and the following Web site `W`:

```
W = {(1) members(status(professor),member(name(mario),surname(rossi)),
             member(status(technician),name(franca),surname(bianchi)),
             member(status(student),name(giulio),surname(verdi))),
     (2) hpage(name(mario),surname(rossi),phone(333),status(professor),
              hobbies(hobby(reading),hobby(gardening))),
     (3) hpage(name(franca),surname(bianchi),status(technician),phone(555)),
     (4) hpage(name(anna),surname(blu),status(professor),phone(444),
              teaching(course(algebra))),
     (5) pubs(pub(name(mario),surname(rossi),title(blahblah1),year(2003)),
              pub(name(anna),surname(blu),title(blahblah2),year(2002)))}
```

Then, by running the VERDI system on Web site `W` and Web specification `I`, we compute the following set of requirements

{(a)♯hpage(name(mario), ♯surname(♯rossi), status),
(b)♯hpage(name(franca), ♯surname(♯bianchi), status),
(c)♯hpage(name(giulio), ♯surname(♯verdi), status),
(d)♯hpage(♯status(♯professor), teaching),
(e)♯member(name(mario), ♯surname(♯rossi)), (f)♯member(name(anna), ♯surname(♯blu)),
(g)♯hpage(name(anna), ♯surname(♯blu), status)}

During the analysis of the set of requirements, our system tries to recognize the structure and the contents of each requirement inside `W`, yielding the following outcomes: (i) requirement `(c)` is missing in Web site `W`, (ii) Web page `(1)` is incomplete w.r.t requirement `(f)`, and (ii) Web page `(2)` is incomplete w.r.t. requirement `(d)`.

Informally, the outputs tell us that (i) the home page of `Giulio Verdi` is missing in `W`, (ii) `Anna blu` should be a member of the research group and (iii) professor `Mario Rossi` should add some teaching information to his personal home page.

5 Implementation

The basic methodology presented so far has been implemented in the prototype VERDI (VErification and Rewriting for Debugging Internet sites), which is written in DrScheme v205 [5] and is publicly available together with a set of tests at `http://www.dimi.uniud.it/~demis/#software`.

The implementation consists of about 80 function definitions (approximately 1000 lines of source code). VERDI includes a parser for semistructured expressions (i.e. XML/HTML documents) and Web specifications, and several modules implementing the user interface, the partial rewriting mechanism and the verification technique. The system allows the user to load a Web site consisting of a finite set of semistructured expressions together with a Web specification. Additionally, he/she can inspect the loaded data and finally check the Web pages w.r.t the Web site specification. The user interface is guided by textual menus, which are (hopefully) self-explaining. We tested the system on several Web site examples which can be found at the URL address mentioned above. In each considered test case, we were able to detect the errors (i.e. missing and incomplete Web pages) efficiently. For instance, the verification of the Web site of Example 2 w.r.t the Web specification of Example 1 is performed almost instantaneously on a standard desktop computer.

References

1. M. Alpuente, D. Ballis, and M. Falaschi. A Rewriting-based Framework for Web Sites Verification. In *Proc. of RULE'04*. ENTCS, Elsevier, 2004. To appear.
2. L. Capra, W. Emmerich, A. Finkelstein, and C. Nentwich. xlinkit: a Consistency Checking and Smart Link Generation Service. *ACM Transactions on Internet Technology*, 2(2):151–185, 2002.
3. M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Verifying Integrity Constraints on Web Sites. In *Proc. of IJCAI'99*, vol. 2, pp. 614–619. Morgan Kaufmann, 1999.
4. M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of IEEE FOCS'95*, pp. 453–462, 1995.
5. PLT. DrScheme web site. Available at: `http://www.drscheme.org`.
6. World Wide Web Consortium (W3C). HyperText Markup Language (HTML) 4.01, 1997. Available at: `http://www.w3.org`.
7. World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0, second edition, 1999. Available at: `http://www.w3.org`.
8. World Wide Web Consortium (W3C). XML Path Language (XPath), 1999. Available at: `http://www.w3.org`.