Int J Softw Tools Technol Transfer (2006) 8:565–585
DOI 10.1007/s10009-006-0009-7

SPECIAL SECTION ON LEVERAGING APPLICATIONS OF FORMAL METHODS

Rule-based verification of Web sites

M. Alpuente · D. Ballis · M. Falaschi

Published online: 26 July 2006

Abstract In this paper, we develop a framework 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 fulfilled. First, we provide a rewriting-based, formal specification language which allows us to define syntactic as well as semantic properties of the Web site. Then, we formalize a verification technique which detects both incorrect/forbidden patterns as well as lack of information, that is, incomplete/missing Web pages inside the Web site. Useful information is gathered during the verification process which can be used to repair the Web site. Our methodology is based on a novel rewriting-based technique, called *partial rewriting*, in which the traditional pattern matching mechanism is replaced by tree *simulation*, a suitable technique for recognizing patterns inside semistructured documents. The framework has been implemented in the prototype GVerdi, which is publicly available.

M. Alpuente (✉)
DSIC, Universidad Politécnica de Valencia,
Camino de Vera s/n, Apdo. 22012, 46071 Valencia, Spain
e-mail: alpuente@dsic.upv.es

D. Ballis
Dip. Matematica e Informatica,
Via delle Scienze 206, 33100 Udine, Italy
e-mail: demis@dimi.uniud.it

M. Falaschi
Dip. di Scienze Matematiche e Informatiche,
Pian dei Mantellini 44, 53100 Siena, Italy
e-mail: moreno.falaschi@unisi.it

1 Introduction

The increasing complexity of Web sites has turned their design and construction into a challenging problem. Systematic, formal approaches can bring many benefits to Web site construction, giving support for automated Web site verification. This paper presents an approach to Web site specification and verification based on rewriting-like machinery. We use rewriting-based technology both to specify the integrity conditions and to formalize a verification technique which obtains the requirements not fulfilled by the Web site, and then is able to diagnose errors by finding out incorrect/forbidden patterns and missing/incomplete pages.

Although the management of Web sites has received significant attention in recent years [10,16,21], few works address the semantic verification of Web sites. In [21], a declarative verification algorithm is developed which checks a particular class of integrity constraints concerning the Web site's structure, but not the contents of a given instance of the site. In [16], a methodology to verify some semantic constraints concerning the Web site contents is proposed, which consists of using inference rules and axioms of natural semantics. The framework XLINKIT [19,32] allows one to check the consistency of distributed, heterogeneous documents as well as to fix the (possibly) inconsistent information. The specification language is a restricted form of first order logic combined with Xpath expressions [37] where no functions are allowed. With respect to the correctness of Web applications, a symbolic model-checking approach is formalized in [17], which constructs a finite states model of the system in the model checker input language, and then checks the considered properties which are expressed in CTL logic. For a comprehensive

survey about the general problem of checking constraints between multiple documents, we refer to [20,18,13].

Our idea in this paper is that term rewriting techniques can support in a natural way not only intuitive, high level Web site specification, but also efficient Web site verification techniques. As far as we know, rewriting-based techniques have not been explored in the context of Web site verification to date. We only know of two rewriting-based approaches for Web site processing, but they focus on transformation rather than verification issues: a term rewriting implementation is provided in [29] for (a fragment of) XSLT, the rule-based language designed by W3C for the transformation of XML documents, whereas rewrite rules are used in [7] to perform HTML transformations with the aim of improving Web applications by cleaning up syntax, reorganizing frames, or updating to new standards. Our rule specification language does offer the expressiveness and computational power of functions and is simpler than formalizations of XML schemata based on tree automata often used in the literature such as, e.g., the regular expression types [26].

Our contribution

We first provide a rewriting-based, formal specification language which allows us to define conditions on both the structure and the contents of Web sites in a simple and concise way. For instance, it allows us to recognize erroneous information inside the Web site and, additionally, to enforce that some information is available at a given Web page, including the adequate links between pages or even the existence of the Web pages themselves. In our formalism, web pages (HTML/XML documents) are modeled as Herbrand terms, and, consequently, Web sites are finite sets of terms. Then, we formalize a verification technique in which a Web site is checked w.r.t. a given Web specification in order to detect incorrect data and incomplete and/or missing Web pages. Moreover, by analyzing the error symptoms gathered during the verification process, we are also able to (1) exactly locate the incorrect/forbidden information and (2) to find out the missing information which the user should provide to repair the Web site. Since reasoning on the Web calls for formal methods specifically fitting the Web context, we combine a standard regular expressions methodology with a novel, rewriting-based technique called *partial rewriting*, in which the traditional pattern matching mechanism is replaced by tree *simulation* [24] in order to provide a suitable mechanism for recognizing patterns inside semistructured documents. The notion of simulation has been already used for dealing with semistructured data in a number of query and transformation languages [10,12,22,14]. The reason is twofold: on the one hand, it provides a powerful method to extract information from semistructured data; on the other hand, there exist efficient algorithms for computing simulations [24]. To assess the feasibility and efficiency of our approach, we have implemented the prototype system GVerdi (Graphical VErification and Rewriting for Debugging Internet sites), which is based on the verification methodology that we propose and is publicly available online. Following the "tolerant" approach of xlinkit [19,32], we do not force the immediate repairing of the Web site, but simply provide the diagnosis information that enables document owners to decide on further actions. Some of the results in this work have been included in Ballis' doctoral thesis [5].

Plan of the paper

The rest of the paper is organized as follows. Section 2 summarizes some preliminary definitions and notations. In Sect. 3, we formulate a simple method for translating XHTML/XML documents into Herbrand terms. Section 4 is devoted to formalize the specification language, whereas Sect. 5 formalizes the *partial rewriting* mechanism, which is based on tree simulation. Section 6 introduces our verification technique for detecting incorrect as well as missing/incomplete Web pages. A description of the system GVerdi is provided in Sect. 7. Section 8 concludes. Proofs of the technical results can be found in the appendix.

2 Preliminaries

We call a finite set of symbols *as alphabet*. Given the alphabet A, A^* denotes the set of all finite sequences of elements over A. Syntactic equality between objects is represented by $\equiv$.

By $\mathcal{V}$ we denote a countably infinite set of variables and Σ denotes a set of function symbols, or *signature*. We consider varyadic signatures as in [15] (i.e., signatures in which symbols have an unbounded arity, that is, they may be followed by an arbitrary number of arguments). $\tau(\Sigma, \mathcal{V})$ and $\tau(\Sigma)$ denote the *non-ground term algebra* and the *term algebra* built on $\Sigma \cup \mathcal{V}$ and Σ, respectively.

Terms are viewed as labeled trees in the following (non-standard) way: a term in $\tau(\Sigma, \mathcal{V})$ is a tree $(V, E, r,$ label), where V is a set of vertices, E is a set of edges (i.e., pairs of vertices), $r \in V$ is the *root* vertex and label is a *labeling* function such that label$(v) \in (\Sigma \cup \mathcal{V})$, for each $v \in V$. Let us see a small example.

Example 1 Consider the term $t \equiv f(h(a), X)$ in $\tau(\{f, h, a\}, \{X\})$, which is illustrated in Fig. 1. Term t can be

represented by the structure (V, E, r, label), where $V \equiv \{v_0, v_1, v_2, v_3\}, E \equiv \{(v_0, v_1), (v_0, v_2), (v_1, v_3)\}, r \equiv v_0$, and function label is defined as follows: label $(v_0) = f$, label $(v_1) = h$, label $(v_2) = X$, label $(v_3) = a$.

Given two vertices $v, v' \in V$ of a term $t \equiv (V, E, r, \text{label})$, by $v \geq v'$ we mean that v is a *descendant* of v' in t. By $t|_v$ we mean the subterm rooted at vertex v of t. $t[r]_v$ is the term t with the subterm rooted at vertex v replaced by term r.

We denote the *depth* of a vertex v in a term t, that is the number of edges between r and v in t, as depth (t, v). A *substitution* $\sigma \equiv \{X_1/t_1, X_2/t_2, \ldots, X_n/t_n\}$ is a mapping from the set of variables $\mathcal{V}$ into the set of terms $\tau(\Sigma, \mathcal{V})$ satisfying the following conditions: (1) $X_i \neq X_j$, whenever $i \neq j$, (2) $X_i\sigma = t_i$, $i = 1, \ldots, n$, and (3) $X\sigma = X$, for any $X \in \mathcal{V} \setminus \{X_1, \ldots, X_n\}$. By $\text{Var}(s)$ we denote the set of variables occurring in the syntactic object s.

In the following, we consider marked terms. Given Σ and $\mathcal{V}$, we denote the *marked* version of Σ ($\mathcal{V}$, respectively) as $\underline{\Sigma}$ ($\underline{\mathcal{V}}$, respectively). A syntactic object $\underline{o} \in \underline{\Sigma} \cup \underline{\mathcal{V}}$ is called the *marked version* of $o \in \Sigma \cup \mathcal{V}$. Given a term $t \equiv (V, E, r, \text{label}) \in \tau(\Sigma, \mathcal{V})$, a *marking* for t is a (boolean) function $\mu : V \to \{\text{yes}, \text{no}\}$. The *empty* marking ε for t is a marking for t, such that $\varepsilon(v) = \text{no}$, for each $v \in V$. We define the *marked part* of a term t as

$$\text{mark}(t, \mu) \equiv (\{v \in V \mid \mu(v) = \text{yes}\}, \{(v_1, v_2) \in E \mid \mu(v_1) = \mu(v_2) = \text{yes}\}, r, \text{label}).$$

A *valid* marking μ for a term $t \equiv (V, E, r, \text{label})$ is the empty marking for t or a marking for t such that the two following conditions hold:

1. $\mu(r) = \text{yes}$.
2. mark (t, μ) is a term in $\tau(\Sigma, \mathcal{V})$.

Given a term $t \equiv (V, E, r, \text{label})$ and a valid marking μ for t, by slightly abusing notation, a *marked* term $\mu(t)$ is a term in $\tau(\Sigma \cup \underline{\Sigma}, \mathcal{V} \cup \underline{\mathcal{V}})$ such that, for each vertex $v \in V$, the label associated with v in t is replaced by its marked version in $\mu(t)$, whenever $\mu(v) = \text{yes}$.

When no confusion can arise, we simply denote the marked term $\varepsilon(t)$ by t.

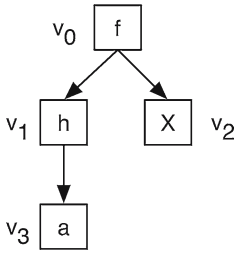

Fig. 1 Term representation of $f(h(a), X)$

Example 2 Consider again the term $t \equiv (f(h(a), X))$ of Example 1. Let μ_1 be a marking for t defined as $\mu_1(v_0) = \mu_1(v_2) = \mu_1(v_3) = \text{yes}$, $\mu_1(v_1) = \text{no}$. Additionally, let μ_2 be a marking for t such that $\mu_2(v_0) = \mu_2(v_1) = \text{yes}$, $\mu_2(v_2) = \mu_2(v_3) = \text{no}$. Note that μ_1 is not a valid marking for t as the marked part of t is not a term in $\tau(\{f, h, a\}, \{X\})$ [see Fig. 2a], whereas μ_2 is valid for t and $\mu_2(t) = \underline{f}(\underline{h}(a), X)$ is a marked term [see Fig. 2b].

2.1 Term rewriting systems

Term rewriting systems provide an adequate computational model for functional languages. In the sequel, we follow the standard framework of term rewriting (see [4,30]). A *term rewriting system* (TRS for short) is a pair (Σ, R), where Σ is a signature and R is a finite set of reduction (or rewrite) rules of the form $\lambda \to \rho$, $\lambda, \rho \in \tau(\Sigma, \mathcal{V})$, $\lambda \notin \mathcal{V}$, and $\text{Var}(\rho) \subseteq \text{Var}(\lambda)$. We will often write just R instead of (Σ, R). Sometimes, we denote the signature of a TRS (Σ, R) by Σ_R.

We formalize the rewriting relation for (V, E, r, label) terms as follows. A rewrite step is the application of a rewrite rule to an expression. A term $s \equiv (V, E, r, \text{label})$ *rewrites* to a term t via $r \in R$, $s \to_r t$ (or $s \to_R t$), if there exist $v \in V$, $r \equiv \lambda \to \rho$, and substitution σ such that $s|_v \equiv \lambda\sigma$ and $t \equiv s[\rho\sigma]_v$. When no confusion can arise, we will omit any subscript (i.e., $s \to t$). By $\to_R^*$, we denote the reflexive, and transitive closure of $\to_R$. A term s is a *irreducible form* (or *normal form*) w.r.t. R, if there is no term t s.t. $s \to_R t$. t is the irreducible form of s w.r.t R (in symbols $s \to_R^! t$) if $s \to_R^* t$ and t is irreducible.

We say that a TRS R is *terminating*, if there exists no infinite rewrite sequence $t_1 \to_R t_2 \to_R \cdots$ A TRS R is *confluent* if, for all terms s, t_1, t_2, $s \to_R^* t_1$ and $s \to_R^* t_2$ imply there exists term t s.t. $t_1 \to_R^* t$ and $t_2 \to_R^* t$. When R is terminating and confluent it is called *canonical*. In canonical TRSs, each input term t can be univocally reduced to an *irreducible form*.

Let $s = t$ be an equation, we say that the equation $s = t$ *holds* in a canonical TRS R, if there exists $z \in \tau(\Sigma, \mathcal{V})$ such that $s \to_R^! z$ and $t \to_R^! z$.

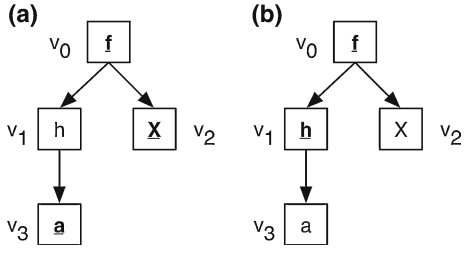

Fig. 2 Examples of valid and non-valid markings for the term $f(h(a), X)$

Example 3 Let R be the following canonical TRS

$$\begin{aligned}
\mathtt{sum}(\mathtt{X},\mathtt{0}) &\to \mathtt{X}\\
\mathtt{sum}(\mathtt{X},\mathtt{s}(\mathtt{Y})) &\to \mathtt{s}(\mathtt{sum}(\mathtt{X},\mathtt{Y}))\\
\\
\mathtt{append}([\,],\mathtt{L}_1) &\to \mathtt{L}_1\\
\mathtt{append}([\mathtt{X}|\mathtt{L}_1],\mathtt{L}_2) &\to [\mathtt{X}|\mathtt{append}(\mathtt{L}_1,\mathtt{L}_2)]\\
\\
\leq(\mathtt{0},\mathtt{X}) &\to \mathtt{true}\\
\leq(\mathtt{s}(\mathtt{X}),\mathtt{0}) &\to \mathtt{false}\\
\leq(\mathtt{s}(\mathtt{X}),\mathtt{s}(\mathtt{Y})) &\to \leq(\mathtt{X},\mathtt{Y})
\end{aligned}$$

In the TRS R we define three functions. Function *sum* computes the sum of two naturals, function *append* concatenates two lists, and, function $\leq$ which defines the "less or equal to" relation between natural numbers. Natural numbers are represented in Peano's notation by means of the constant `0` and the unary operator `s`. By abuse we will write $n \in \mathbb{N}$ as a shorthand for $\mathtt{s}^n(\mathtt{0})$. We use the standard notation for lists with [ ] being the empty list. Strings are viewed as lists of characters as usual.

3 Denotation of Web sites

In our framework, a *Web page* is either an XML [36] or an XHTML [38] document, which we assumed to be well-formed. There are already programs and online services such as Tidy [33], which are able to validate and correct the XHTML/XML syntax, and Doctor HTML [28], which also performs link-checking.

Let us consider two alphabets T and $\mathcal{T}$ag. We denote the set T^* by $\mathcal{T}$ext. An object $\mathtt{t} \in \mathcal{T}$ag is called *tag* element, while an element $\mathtt{w} \in \mathcal{T}$ext is called *text* element. Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of a given term algebra $\tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag})$ as shown in Fig. 3. Note that XML/XHTML tag attributes can be considered as common tagged elements, and hence translated in the same way.

A *marked* Web page is defined as $\mu(\mathtt{p})$, where $\mathtt{p} \in \tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag})$ and μ is a valid marking for $\mathtt{p}$. A *Web site* is a finite collection of marked Web pages $\{\varepsilon(\mathtt{p}_1) \ldots \varepsilon(\mathtt{p}_n)\}$. In the following, we will also consider terms of the non-ground term algebra $\tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag}, \mathcal{V})$, which may contain variables. An element $\mathtt{s} \in \tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag}, \mathcal{V})$ is called *Web page template*. $\mu(\mathtt{s})$ is a *marked* Web page template, when $\mathtt{s} \in \tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag}, \mathcal{V})$ and μ is a valid marking for $\mathtt{s}$. In our methodology, (marked) Web page templates are used for specifying properties on Web sites as described in the following section.

Example 4 In the following, we present a Web site `W` of a research group, which contains information about group members affiliation, scientific publications, research projects, teaching and personal data.

```
{(1) members(member(name(mario),
                    surname(rossi),
                    status(professor)),
             member(name(franca),
                    surname(bianchi),
                    status(technician)),
             member(name(giulio),
                    surname(verdi),
                    status(student)),
             member(name(mario),
                    surname(rossi),
                    status(professor))
           ),
 (2) hpage(fullname(mariorossi),phone(3333),
       status(professor),hobbies(
                          hobby(reading),
                          hobby(gardening))),
 (3) hpage(fullname(francabianchi),
       status(technician),phone(5555),
         links(link(url(www.google.com),
               urlname(google)),
             link(url(www.sexycalculus.com),
               urlname(FormalMethods))),
 (4) hpage(fullname(annagialli),
         status(professor),
         blink(phone(4444)),
         teaching(courselink(
              url(http://www.algebra.math),
              urlname(Algebra)))),
 (5) pubs(pub(name(mario),surname(rossi),
              title(blah1),year(2003)),
          pub(name(anna),surname(gialli),
              title(blah2),year(2002))),
 (6) projects(project(pname(A1),grant1(1000),
       grant2(200), total(1200), coordinator(
                        fullname(mariorossi))),
       project(pname(B1),grant1(800),
                         grant2(300),
               projectleader(surname(gialli),
                             name(anna)),
                             total(1000)))}
```

4 Web specification language

In the following, we present a term rewriting specification language, which is helpful to express properties about the content and the structure of a given Web site. Roughly speaking, a Web specification is a pair of finite set of rules. The first set of rules describes constraints for detecting erroneous Web pages (*correctNess rules*) as well as discovering incomplete/missing Web pages (*coMpleteness rules*). Diagnoses are carried out

Fig. 3 An XML document and its corresponding encoding as a ground term p

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

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

by running Web specifications on Web sites. The operational mechanism, formalized in Sect. 5, is based on a novel rewriting-based technique, which is able to extract partial structure from a term, and then rewrite it. The second set of rules R contains the definition of some auxiliary functions which the user would like to provide, such as string processing, arithmetic, boolean operators, etc. It is formalized as a canonical term rewriting system which is handled by standard rewriting [30].

Correctness rules allow us to recognize incorrect and forbidden patterns in the Web site and to locate the wrong Web pages containing these errors. We address these problem by considering the specific nature of semistructured documents, that is, we combine a structured pattern search technique with a more standard textual search, which is based on regular expressions detection. For the sake of simplicity, we consider an intuitive Unix-like regular expression syntax [34]. In contrast to other Web verification methods, we are able to deal with correctness rules containing interpreted functions, which allow us to check whether the values that may appear in the semistructured documents are correctly computed.

The verification process is carried out by first extracting a partial structure of a (possibly incorrect) Web page, and then checking whether the required constraints are fulfilled. These constraints are expressed by means of equations and regular expressions. Correctness rules are formalized as follows.

Definition 1 *(Correctness rule) Let* (Σ, R) *be a canonical TRS. A* correctness rule *has the following form*

$$\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{C}$$

where

1. $\mathtt{l} \in \tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$ *is a Web page template and* $\mathtt{error} \notin (\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag} \cup \Sigma)$ *is a new fresh constant.*
2. $\mathtt{C}$ *is a (possibly empty) sequence*

   $$X_1 \ \mathtt{in}\ \mathtt{rexp}_1, \ldots, X_n \ \mathtt{in}\ \mathtt{rexp}_n, \Gamma$$

   with Var $(\mathtt{C}) \subseteq \mathrm{Var}(\mathtt{l})$*,* $\mathtt{rexp}_i$ *a regular expression over* $(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag})$*,* $i = 1, \ldots, n$*, and* Γ *a sequence of equations over* $\tau(\Sigma, \mathcal{V})$*.*

When $\mathtt{C}$ *is empty, we simply write* $\mathtt{l} \rightharpoonup \mathtt{error}$*.*

Given a correctness rule $r \equiv (\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{C})$, we call $\mathtt{l} \rightharpoonup \mathtt{error}$ the *unconditional part of r* and we denote it by r_u. For the sake of expressiveness, we also allow to write inequalities of the form $\mathtt{s} \neq \mathtt{t}$ in the conditional part of the correctness rules. Such inequalities are just syntactic sugar for $(\mathtt{s} = \mathtt{t}) = \mathtt{false}$. Informally, the meaning of a correctness rule $\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{C}$ is the following. Whenever an instance $\mathtt{l}\sigma$ of $\mathtt{l}$ is recognized in some Web page $\mathtt{p}$, and

(1) Each structured text $X_i\sigma$, $i = 1, \ldots, n$, is contained in the language of the corresponding regular expression $\mathtt{rexp}_i$.
(2) Each instantiated equality $(\mathtt{s} = \mathtt{t})\sigma$ in Γ holds in the canonical TRS R.

then, Web page $\mathtt{p}$ is signaled as an incorrect page.

Completeness rules of a Web specification formalize the requirement that some information must be included in all or some pages of the Web site. We use attributes $\langle \mathtt{A} \rangle$ and $\langle \mathtt{E} \rangle$ to distinguish "universal" from "existential" rules. Right-hand sides of completeness rules may contain functions, which are defined by the user via a canonical TRS.

Definition 2 *(Completeness rule) Let* (Σ, R) *be a canonical TRS. A* completeness rule *is either a* universal *rule of the form* $\mathtt{l} \rightharpoonup \mu(\mathtt{r})\ \langle \mathtt{A} \rangle$ *or an* existential *rule of the form* $\mathtt{l} \rightharpoonup \mu(\mathtt{r})\ \langle \mathtt{E} \rangle$*, where* $\mathtt{l} \in \tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$*,*

$r \in \tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag} \cup \Sigma, \mathcal{V})$, $\text{Var}(r) \subseteq \text{Var}(l)$ *and* μ *is a valid marking for* r.

Intuitively, the interpretation of a universal rule $l \rightharpoonup \mu(r)\,\langle A\rangle$ [respectively, an existential rule $l \rightharpoonup \mu(r)\,\langle E\rangle$] w.r.t. a Web site W is as follows: if (an instance of) l is recognized in W, also (an instance of) the irreducible form of r must be recognized in *all* (respectively, *some*) of the Web pages which embed (an instance of) the marked part of r. Informally, marking information provides the "scope" of the universal/existential quantifiers of the rule, and allows us to compute the part of the Web site which is affected by the quantification.

Example 5 Consider the Web site

$$W = \{f(h(g(a),t)), h(t,s(m,n),g(p)), h(n,p(r,q)), w(s(m,n))\}$$

and the rule $r \equiv (w(X) \rightharpoonup \underline{h}(\underline{t},X)\langle q\rangle)$, $q \in \{E,A\}$. Then, the marked part of the right-hand side of r is $h(t)$.

Thus, the property, which is modeled by r, will be checked only on the set $\{f(h(g(a),t)), h(t,s(m,n),g(p))\}$, which contains all and only the Web pages of W embedding the term $h(t)$.

Formally, Web specifications are as follows.

Definition 3 (Web specification) *A Web specification is a pair* (I,R)*, where* $I \equiv I_N \uplus I_M$ *is a finite set of rules such that* I_N *(respectively,* I_M*) is a set of correctness (respectively, completeness) rules, and R is a canonical TRS.*

In order to provide an effective, and terminating verification method, we require that the arguments of the function calls in the right-hand sides of the completeness rules do not contain symbols in

$$\{f \mid f(t_1,\ldots,t_n) \rightharpoonup \mu(r) \in I_m\}$$

(for the technical reasons, see Lemma 3 in the appendix).

Given a set of completeness rules I_M, we denote the set of all left-hand sides (right-hand sides without marks, respectively) of rules in I_M by Lhs_M (Rhs_M, respectively). In symbols, $\text{Lhs}_M = \{l \mid l \rightharpoonup \mu(r) \in I_M\}$ and $\text{Rhs}_M = \{r \mid l \rightharpoonup \mu(r) \in I_M\}$.

Note that, by using the traditional encoding of boolean operations by means of rewrite rules [31,23], and by introducing non-deterministic rewriting [27], it would also be possible to extend our framework to a richer specification language providing non-confluent functions such as those that express disjunctive conditions (so that the presence of information of kind A requires information of kind B or C to be also represented). For the sake of simplicity, we do not deal with non-confluent TRSs in this work.

The following example illustrates the definition of a Web specification. Marks are introduced by the user to select those Web pages for which she wants to check the specified integrity conditions.

Example 6 Consider the Web specification which consists of the canonical TRS R of Example 3 together with the built-in definition of function `Nat(X)`, which translates a string `X` to a natural number, and the completeness and correctness rules of Fig. 4.

This Web specification models some required properties for the Web site of Example 4.

The first four rules are completeness rules, while the remaining ones are correctness rules. First rule formalizes the following property: if there is a Web page containing a member list, then for each member, a home page should exist which contains (at least) the full name and the status of this member. The full name is computed by appending the name and the surname strings by means of the standard `append` function whose definition is given in TRS R. The marking information establishes that the property must be checked only on home pages (i.e., pages containing the tag "hpage"). Second rule states that, whenever a home page of a professor is recognized, that page must also include some teaching information. The rule is universal, since it must hold for each professor home page. Such home pages are selected by exploiting the mark given on the tag "status". 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. In this case, we must check the property only in the Web page containing the group member list. The fourth rule formalizes that, for each link to a course, a page describing that course must exist. The verification process is carried out only on Web pages containing course information as described by marks.

The fifth rule forbids sexual contents from being published in the home pages of the group members. This is enforced by requiring that the word `sex` does not occur in any home page by using the regular expression `[:TextTag:]* sex [:TextTag:]*`, which identifies the regular language of all the strings built over $(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag})$ containing that word. The sixth rule is provided with the aim of improving accessibility for people with disabilities. It simply states that blinking text is forbidden in the whole Web site. The seventh rule states that, for each research project, the total project budget must be equal to the sum of the funds, which has been granted for the first and the second research period. The eighth rule formalizes the condition that only recent publications are referred to in the Web site (after the

Fig. 4 Correctness and completeness rules of Example 6

$$\texttt{member(name(X), surname(Y))} \rightharpoonup \underline{\texttt{hpage}}\texttt{(fullname(append(X, Y)),status)}\ \langle \texttt{E} \rangle$$
$$\texttt{hpage(status(professor))} \rightharpoonup \underline{\texttt{hpage}}\texttt{(}\underline{\texttt{status}}\texttt{(}\underline{\texttt{professor}}\texttt{), teaching))}\ \langle \texttt{A} \rangle$$
$$\texttt{pubs(pub(name(X), surname(Y)))} \rightharpoonup \underline{\texttt{member}}\texttt{(name(X),surname(Y))}\ \langle \texttt{E} \rangle$$
$$\texttt{courselink(url(X), urlname(Y))} \rightharpoonup \underline{\texttt{cpage}}\texttt{(title(Y))}\ \langle \texttt{E} \rangle$$
$$\texttt{hpage(X)} \rightharpoonup \texttt{error} \mid \texttt{X in [:TextTag:]* sex [:TextTag:]*}$$
$$\texttt{blink(X)} \rightharpoonup \texttt{error}$$
$$\texttt{project(grant1(X), grant2(Y), total(Z))} \rightharpoonup \texttt{error} \mid \texttt{sum(Nat(X), Nat(Y))} \neq \texttt{Nat(Z)}$$
$$\texttt{pub(year(X))} \rightharpoonup \texttt{error} \mid \texttt{X in [0-9]*}, \leq \texttt{(Nat(X), 1999)} = \texttt{true}$$
$$\texttt{members(member(name(X), surname(Y)), member(name(X), surname(Y)))} \rightharpoonup \texttt{error}$$

year 1999). Finally, the last rule forbids repetitions of the same member entry in the group member list.

The verification of both kinds of rules is mechanized by means of partial rewriting.

5 Partial rewriting

In order to mechanize the intended semantics of Web specification rules, we first devise a mechanism which is able to recognize the structure and the labeling of a given Web page template inside a particular page of the Web site. This is provided by page simulation. Without loss of generality, in the following, we disregard the conditional part of correctness rules and the existential/universal quantification of completeness rules. In other words, a rule is just a pair of (possibly) marked terms $\texttt{l} \rightharpoonup \mu(\texttt{r})$.

5.1 Page simulations

The notion of *page simulation* for Web pages allows us to analyze and extract the partial structure of the Web site which is subject to verification.

Roughly speaking, a Web page $\texttt{p}_1$ is simulated by a Web page $\texttt{p}_2$, if the tree-structure of $\texttt{p}_1$ is "embedded" into the tree-structure of $\texttt{p}_2$. In other words, a simulation of a Web page (i.e., a labeled tree) $\texttt{p}_1$ in a Web page $\texttt{p}_2$ can be seen as a relation among the nodes of $\texttt{p}_1$ and the nodes of $\texttt{p}_2$ which preserves the edges and the labelings. Before formalizing the idea, we illustrate it by means of a rather intuitive example.

Example 7 Consider the following Web pages (called $\texttt{p}_1$ and $\texttt{p}_2$, respectively):

```
hpage(name,surname,status(professor),teaching)
```

```
hpage(name(mario),surname(rossi),
      status(professor),
      teaching(course(logic1),
               course(logic2)),
      hobbies(hobby(reading),hobby(gardening)))
```

Looking at Fig. 5, we observe that the structure of $\texttt{p}_1$ can be recognized inside the structure of $\texttt{p}_2$ by considering the relation among nodes of $\texttt{p}_1$ and $\texttt{p}_2$, which is described by the dashed arrows in the figure. This relation essentially provides the so-called *simulation of* $\texttt{p}_1$ *in* $\texttt{p}_2$. Note that vice-versa does not hold: no relations can be found among nodes of $\texttt{p}_2$ and $\texttt{p}_1$, which "embed" the structure of $\texttt{p}_2$ into $\texttt{p}_1$. In other words, there does not exist a simulation of $\texttt{p}_2$ in $\texttt{p}_1$.

Simulations have been used in a number of works dealing with querying and transformation of semistructured data. For instance, [1,22] propose some techniques based on simulation for analyzing semistructured data w.r.t. a given schema. The language Xcerpt [11,10] is a (logic) query language for XML and semistructured documents, which implements a sort of unification by exploiting the notion of graph simulation. Other approaches involving simulation, or closely related notions, have been employed to measure similarity among semistructured documents [8]. To keep our framework simple, we do not consider a semantic change/load for labels; this would require to introduce ontologies, which are outside the scope of the work.

In the following, we provide our notion of simulation which is a slight adaptation of the one given in [10] to consider Web page templates: we generalize the usual label relation to cope with the case when variables are used as labels, in the following definition.

Definition 4 *Let* $\texttt{s}_1 \equiv (V_1, E_1, r_1, \text{label}_1)$, $\texttt{s}_2 \equiv (V_2, E_2, r_2, \text{label}_2)$ *be two Web page templates in* $\tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag}, \mathcal{V})$. *The* label *relation* $\sim \subseteq V_1 \times V_2$ *is defined as follows:*

$$v_1 \sim v_2 \quad \textit{iff} \quad \text{label}_1(v_1) = \text{label}_2(v_2) \textit{ or } \text{label}_1(v_1) \in \mathcal{V}.$$

Definition 5 *Let* $\texttt{s}_1 \equiv (V_1, E_1, r_1, \text{label}_1)$, $\texttt{s}_2 \equiv (V_2, E_2, r_2, \text{label}_2)$ *be two Web page templates in* $\tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag}, \mathcal{V})$ *and* $\sim \subseteq V_1 \times V_2$ *be the corresponding label relation. A* (page) simulation *of* $\texttt{s}_1$ *in* $\texttt{s}_2$ *w.r.t.* $\sim$ *is a relation* $\mathbf{S} \subseteq V_1 \times V_2$ *such that, for each* $v_1 \in V_1, v_2 \in V_2$

1. $r_1\ \mathbf{S}\ r_2$.
2. $v_1\ \mathbf{S}\ v_2 \Rightarrow v_1 \sim v_2$.
3. $v_1\ \mathbf{S}\ v_2 \wedge (v_1, v_1') \in E_1 \Rightarrow \exists\, v_2' \in V_2,\ v_1'\ \mathbf{S}\ v_2' \wedge (v_2, v_2') \in E_2$.

We define the *projection* of a simulation $\mathbf{S}$ of $\texttt{s}_1$ in $\texttt{s}_2$ w.r.t $\sim$ as $\pi(\mathbf{S}) = \{v_2 \mid (v_1, v_2) \in \mathbf{S}\}$.

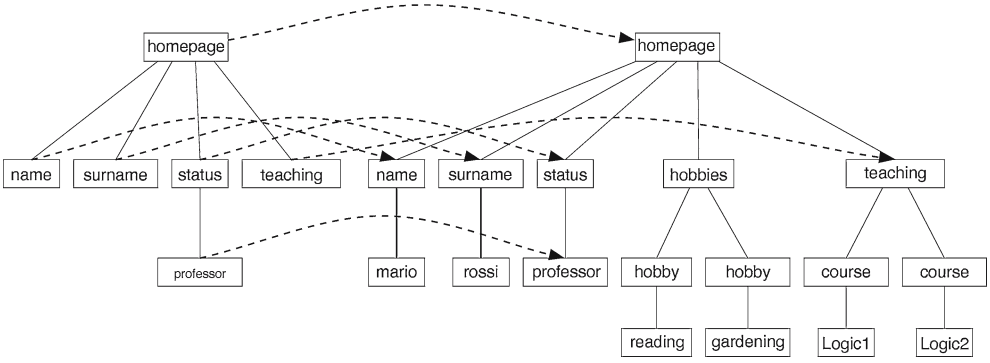

Fig. 5 Page simulation between $\mathtt{p}_1$ and $\mathtt{p}_2$

Roughly speaking, Definition 5 ensures two degrees of similarity between Web page templates, not only w.r.t. the labelings but also w.r.t. the structures of the templates. On the one hand, Condition (2) of Definition 5 formalizes the similarity w.r.t. labelings, that is, any pair of nodes (v, v') in a page simulation $\mathbf{S}$ of $\mathtt{s}_1$ in $\mathtt{s}_2$ have the same label, otherwise node v must be labeled by a variable, which somehow means that the label of v can be seen as a generalization of any concrete label of v'. Finally, Conditions (1) and (3) provide a relation between the tree structure of $\mathtt{s}_1$ and $\mathtt{s}_2$.

Note that simulations are just relations among nodes of two given Web page templates. For our purposes, we are interested in simulations which are injective mappings from nodes of a given Web page template to nodes of another Web page template. As it will be apparent later, those simulations allow us to project the structure of a Web page template into another one, thus performing a sort of "partial" pattern matching between templates, which will be exploited to formulate our verification technique. So, we first define a subclass of simulations called minimal simulations.

Definition 6 *Let* $\mathtt{s}_1 \equiv (V_1, E_1, r_1, \mathrm{label}_1)$, $\mathtt{s}_2 \equiv (V_2, E_2, r_2, \mathrm{label}_2)$ *be two Web page templates in* $\tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$. *A page simulation* $\mathbf{S}$ *of* $\mathtt{s}_1$ *in* $\mathtt{s}_2$ *w.r.t.* $\sim$ *is* minimal *if there are no page simulations* $\mathbf{S}'$ *of* $\mathtt{s}_1$ *in* $\mathtt{s}_2$ *w.r.t.* $\sim$ *such that* $\mathbf{S}' \subseteq \mathbf{S}$.

It is straightforward to prove that minimal simulations are mappings.

Proposition 1 *Let* $\mathtt{s}_1 \equiv (V_1, E_1, r_1, \mathrm{label}_1)$, $\mathtt{s}_2 \equiv (V_2, E_2, r_2, \mathrm{label}_2)$ *be two Web page templates in* $\tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$. *A minimal page simulation* $\mathbf{S}$ *of* $\mathtt{s}_1$ *in* $\mathtt{s}_2$ *w.r.t.* $\sim$ *is a mapping* $\mathbf{S} : V_1 \to V_2$.

Let us see an example which illustrates the notion of minimal simulation.

Example 8 Let us consider the following Web page templates $\mathtt{s}_1$ and $\mathtt{s}_2$:

```
hobbies(hobby(X))
hobbies(hobby(reading),hobby(gardening)).
```

In Fig. 6a, the dashed arrows represent a non-minimal simulation of $\mathtt{s}_1$ in $\mathtt{s}_2$, while in Fig. 6b and c two minimal simulations of $\mathtt{s}_1$ in $\mathtt{s}_2$ are depicted. Note that the last two simulations are mappings.

However, minimal simulations do not guarantee that the tree structure of a given Web page template is recognized inside another template. Consider, for instance, the page simulation of $f(X, Y)$ in $f(a)$ depicted in Fig. 7: it is minimal, but the tree structures of $f(X, Y)$ and $f(a)$ are distinct.

For this purpose, we need a one-to-one correspondence between edges of considered Web page templates. Therefore, we only consider minimal and *injective* simulations. Given two Web page templates $\mathtt{s}$ and $\mathtt{t}$, we denote by $\mathtt{s} \cong \mathtt{t}$, the fact that there exists a minimal, injective simulation of $\mathtt{s}$ in $\mathtt{t}$ w.r.t. $\sim$.

It is not difficult to prove that minimal injective simulations are particular instances of Kruskal's *embeddings* [9] w.r.t. the relation $\sim$. In other words, a minimal injective page simulation of $\mathtt{s}_1$ in $\mathtt{s}_2$ w.r.t. $\sim$ exists iff $\mathtt{s}_1$ is embedded into $\mathtt{s}_2$ w.r.t. $\sim$, i.e., we are able to find out the structure and the labeling of $\mathtt{s}_1$ inside $\mathtt{s}_2$.

5.2 Rewriting Web page templates

Definition 7 *Let* $\mathtt{s}_1 \equiv (V_1, E_1, r_1, \mathrm{label}_1)$, $\mathtt{s}_2 \equiv (V_2, E_2, r_2, \mathrm{label}_2) \in \tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$. *We say that* $\mathtt{s}_2$ partially matches $\mathtt{s}_1$ *via substitution* σ *iff*

1. *There exists a minimal, injective page simulation* $\mathbf{S}$ *of* $\mathtt{s}_1$ *in* $\mathtt{s}_2$ *w.r.t.* $\sim$.
2. *For each* $(v, v') \in \mathbf{S}$ *such that* $\mathrm{label}(v) = X \in \mathcal{V}$, $\sigma(X) = (\mathtt{s}_{2|v'})$.

In Definition 7, we consider only minimal, injective simulations between Web page templates $\mathtt{s}_1$ and $\mathtt{s}_2$ to easily compute a substitution σ such that there exists a

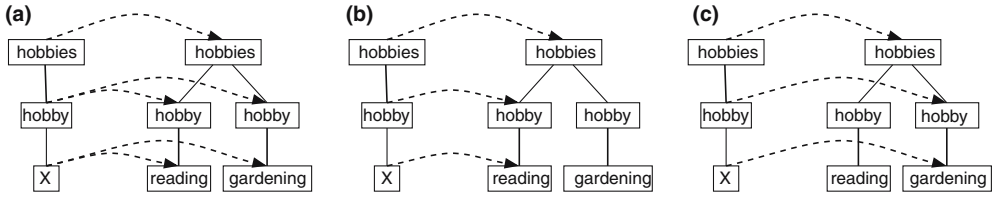

Fig. 6 Non-minimal and minimal simulations

simulation of $\mathtt{s}_1\sigma$ in $\mathtt{s}_2$ w.r.t. $\sim$; in other words, $\mathtt{s}_1\sigma$ is embedded into $\mathtt{s}_2$. It is worth noting that non-minimal simulations not always ensure the existence of such a substitution. This is the reason why the minimal simulations are also required to be injective. Let us see an example.

Example 9 Consider again Example 8. We have that $\mathtt{s}_2$ partially matches $\mathtt{s}_1$ via $\{\mathtt{X/reading}\}$ (see Fig. 6b) and $\mathtt{s}_2$ partially matches $\mathtt{s}_1$ via $\{\mathtt{X/gardening}\}$ (see Fig. 6c). Note that performing partial matching by the non-minimal simulation of Fig. 6 a would produce $\sigma \equiv \{\mathtt{X/reading}, \mathtt{X/gardening}\}$, which is not a substitution.

Now we are ready to define a partial rewrite relation between marked Web page templates, which includes a simplification stage using the user functions of a TRS R.

Definition 8 *Let R be a canonical TRS.*
Let $\mathtt{s} \equiv (V, E, r_{\mathtt{s}}, \text{label})$, $\mathtt{t} \in \tau(\mathcal{T}\text{ext} \cup \mathcal{T}\text{ag}, \mathcal{V})$. Let μ_1 and μ_2 be two valid markings for $\mathtt{s}$ and $\mathtt{t}$, respectively. Then, $\mu_1(\mathtt{s})$ partially rewrites *to $\mu_2(\mathtt{t})$ via rule $rl \equiv \mathtt{l} \rightharpoonup \mu(\mathtt{r})$ and substitution σ (in symbols, $\mu_1(\mathtt{s}) \rightharpoonup_{rl}^{\sigma} \mu_2(\mathtt{t})$) iff there exists $v \in V$ such that*

1. $\mathtt{s}_{|v}$ partially matches $\mathtt{l}$ *via* σ.
2. *Let* $\mathtt{r} \equiv (V_{\mathtt{r}}, E_{\mathtt{r}}, r, \text{label}_{\mathtt{r}})$ *and* $\mathtt{r}\sigma \equiv (V_{\mathtt{r}\sigma}, E_{\mathtt{r}\sigma}, r, \text{label}_{\mathtt{r}\sigma})$. *For each* $v \in V_{\mathtt{r}\sigma}$,

$$\mu_2(v) = \begin{cases} \mu(v) \text{ if } v \in (V_{\mathtt{r}} \cap V_{\mathtt{r}\sigma}) \\ \mu(v') \text{if } v \in (V_{\mathtt{r}\sigma} \setminus V_{\mathtt{r}}) \wedge (\exists\, v' \in V_{\mathtt{r}}, v \geq v', \\ \qquad \text{label}_{\mathtt{r}}(v') \in \text{Var}(\mathtt{r})) \end{cases}$$

3. $\mathtt{t} = \text{Reduce}(\mathtt{r}\sigma, R)$, *where function* $\text{Reduce}(x, R)$ *computes, by standard term rewriting, the irreducible form of x in R ignoring the eventual marks for the functions in R.*

When rule rl and substitution σ are understood, we simply write $\mu_1(\mathtt{s}) \rightharpoonup \mu_2(\mathtt{t})$.

It is worth noting that we provide a notion of partial rewriting in which the context of the selected reducible expression $\mathtt{s}_{|v}$ of the Web page template which is rewritten is disregarded after the rewrite step [see point (3) of Definition 8]. Roughly speaking, given a Web specification rule $\mathtt{l} \rightharpoonup \mu(\mathtt{r})$, partial rewriting allows us to extract a subpart of a given Web page (template) $\mathtt{s}$, which partially matches $\mathtt{l}$, and to replace $\mathtt{s}$ by a reduced instance of $\mathtt{r}$; namely, $\text{Reduce}(\mathtt{r}\sigma, R)$ [see points (1) and (3) of Definition 8]. Point (2) of Definition 8 establishes that rewritten templates inherit marks from the right-hand sides of the applied rules. More precisely,

- Each vertex of $\mathtt{r}\sigma$, which is not affected by substitution σ, maintains the same marking of $\mathtt{r}$.
- Each vertex, which belongs to a subterm of $\mathtt{r}\sigma$ replacing a variable $\underline{X}$ of $\mathtt{r}$, is marked *yes*.
- Each vertex, which belongs to a subterm of $\mathtt{r}\sigma$ replacing a variable X of $\mathtt{r}$, is marked no.

A *partial* rewrite sequence is of the form $\mu_0(\mathtt{s}_0) \rightharpoonup_{r_0}^{\sigma_0} \mu_1(\mathtt{s}_1) \rightharpoonup_{r_1}^{\sigma_1} \ldots$ Moreover, we denote the transitive closure (resp., the transitive and reflexive closure) of $\rightharpoonup$ by $\rightharpoonup^+$ (resp., $\rightharpoonup^*$). By notation $\mu_0(\mathtt{s}_0) \rightharpoonup^n \mu_1(\mathtt{s}_1)$ we denote a partial rewrite sequence of length n (i.e., a partial rewrite sequence which is made up of n partial rewrite steps).

Example 10 Consider the Web page $\mathtt{p}$ of Fig. 3 and the first rule $\mathtt{r}_1$ of the Web specification of Example 6. Let us suppose that TRS R defines the standard function `append` for concatenating strings. Then, Web page template $\varepsilon(\mathtt{p})$ partially rewrites to the following Web pages by applying $\mathtt{r}_1$.

$$\varepsilon(\mathtt{p}) \rightharpoonup_{\mathtt{r}_1} \textit{Reduce}(\underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{append}(\mathtt{mario}, \mathtt{rossi})), \mathtt{status}), R) = \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{mariorossi}), \mathtt{status})$$

$$\varepsilon(\mathtt{p}) \rightharpoonup_{\mathtt{r}_1} \textit{Reduce}(\underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{append}(\mathtt{franca}, \mathtt{bianchi})), \mathtt{status}), R) = \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{francabianchi}), \mathtt{status})$$

$$\varepsilon(\mathtt{p}) \rightharpoonup_{\mathtt{r}_1} \textit{Reduce}(\underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{append}(\mathtt{giulio}, \mathtt{verdi})), \mathtt{status}), R) = \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{giulioverdi}), \mathtt{status})$$

Roughly speaking, marks in the right-hand sides of the rules allow us to find sets of Web pages, which might

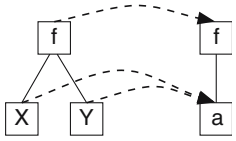

Fig. 7 Minimal, non-injective simulation

be incomplete or missing. Then, real buggy pages are detected inside these sets. We formalize the idea in the following section.

6 The verification framework

In the following, we show how simulation and partial rewriting can be applied to verify a given Web site w.r.t. a Web specification. As we have seen in Sect. 5.1, simulation allows us to identify the structure of a given Web page (possibly, a template) into another one. By taking advantage of this fact, we can develop a methodology, which is able to discover correctness as well as completeness errors in a given Web site w.r.t. a Web specification.

More precisely, our analysis allows us to discover the following kinds of errors:

- Erroneous/forbidden information in the Web site (*correctness errors*).
- Web pages which are missing in a Web site or Web pages which are incomplete w.r.t. a given Web specification (*completeness errors*).

6.1 Detecting correctness errors

In this section, we provide a simple mechanism based on partial rewriting which can detect erroneous or undesirable data included in a Web site. Our methodology allows us to precisely locate which part of a Web page does not fulfill the Web specification. We apply correctness rules to the Web pages of the Web site in order to discover incorrect patterns. More precisely, given a Web page $\mathtt{p}$, we first try to recognize a given Web page template $\mathtt{l}$ into $\mathtt{p}$ by partially rewriting $\mathtt{p}$ via the (unconditional part of a) correctness rule $\mathtt{l} \rightharpoonup \mathtt{error}|\mathtt{C}$. Then, we analyze the values taken by the variables of $\mathtt{C}$, which are obtained as a by-product of the partial rewrite step. If the structured text, which is bound to each variable in $\mathtt{C}$, belongs to the language of the corresponding regular expression, and all the instantiated equations in $\mathtt{C}$ hold, the faulty Web page $\mathtt{p}$ and an incorrectness symptom are supplied to the user.

Definition 9 *Let* $\mathtt{W}$ *be a Web site,* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification. Given* $\mathtt{p} \in \mathtt{W}$*, we say that* $\mathtt{p}$ *is* incorrect *w.r.t.* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$*, if there exists a rule* $r \equiv (\mathtt{l} \rightharpoonup \mathtt{error}|X_1\ \mathtt{in}\ \mathtt{rexp}_1, \ldots, X_n\ \mathtt{in}\ \mathtt{rexp}_n, \Gamma) \in \mathtt{I_N}$ *such that*

1. $\mathtt{p} \rightharpoonup^{\sigma}_{r_u} \mathtt{error}$*, where* r_u *is the unconditional part of* r*.*
2. *For* $i = 1, \ldots, n$*,* $X_i\sigma \in \mathcal{L}(\mathtt{rexp}_i)$*, where* $\mathcal{L}(\mathtt{rexp}_i)$ *is the regular language described by* $\mathtt{rexp}_i$*.*
3. *Each instantiated equation of* Γ*,* $(\mathtt{s} = \mathtt{t})\sigma$*, holds in* R*.*

We say that $\mathtt{l}\sigma$ *is an* incorrectness symptom *for* $\mathtt{p}$*.*

Let us see an example which illustrates the above definition.

Example 11 Let $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ be the Web specification of Example 6 and $\mathtt{W}$ be the Web site of Example 4.

Now, consider the correctness rule

```
r ≡ hpage(X) ⇀ error | X in
                    [:TextTag:] ∗ sex[:TextTag:] ∗ .
```

Note that the only Web page in $\mathtt{W}$ which can yield a correctness error by using r is Web page `(3)`, since `(3)` can be partially rewritten to `error` via r_u by means of the following substitution σ

```
{X/links(link(url(www.google.com),
              urlname(google)),
         link(url(www.sexycalculus.com),
              urlname(FormalMethods)))}
```

and $\mathtt{X}\sigma$ belongs to

$$\mathcal{L}(\mathtt{[:TextTag:]*\ sex\ [:TextTag:]*}).$$

The corresponding incorrectness symptom is

```
hp(links(link(url(www.google.com),
              urlname(google)),
         link(url(www.sexycalculus.com),
              urlname(FormalMethods)))).
```

Web page `(4)` is incorrect w.r.t $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$, as it rewrites to `error` by rule $\mathtt{blink(X)} \rightharpoonup \mathtt{error}$ and gives rise to the incorrectness symptom `blink(phone(4444))`. Moreover, we can also discover that the total budget of project `B1` is wrongly computed by applying rule

```
project(grant1(X), grant2(Y),
    total(Z)) ⇀ error| sum(Nat(X),Nat(Y)) ≠ Nat(Z).
```

Indeed, Web page `(6)` is incorrect, since the equation

$$\mathtt{sum(Nat(800),Nat(300))} \neq \mathtt{Nat(1000)}$$

hold in R. The associated incorrectness symptom is

```
project(grant1(800),grant2(300),total(1000)).
```

Finally, Rule

$r' \equiv \texttt{members}(\ \texttt{member}(\texttt{name}(\texttt{X}), \texttt{surname}(\texttt{Y})),$
$\texttt{member}(\texttt{name}(\texttt{X}), \texttt{surname}(\texttt{Y}))) \rightharpoonup \texttt{error}$

detects that the entry for `Mario Rossi` in Web page `(1)` is repeated twice, since it derives `error` by using r' via substitution $\{\texttt{X/mario}, \texttt{Y/rossi}\}$. The respective incorrectness symptom is:

```
members(member(name(mario),surname(rossi)),
        member(name(mario),surname(rossi))).
```

The example above points out the usefulness of incorrectness symptoms: they allow us to precisely locate which erroneous piece of information must be modified by the user in order to repair the faulty Web site.

Algorithm 1 outlines a procedure for the detection of correctness errors, which takes as input a Web site $\texttt{W}$, a set of correctness rules $\texttt{I}_\texttt{N}$, and a canonical TRS R. Basically, the procedure repeatedly applies the test of Definition 9 for recognizing incorrect and forbidden patterns. More precisely, for every Web page in $\texttt{W}$, we verify whether (1) $\texttt{p}$ reduces to the constant `error` via the unconditional part of some correctness rule r, and (2) the constraints in the condition of r are fulfilled. In the case that an error is found in a Web page $\texttt{p}$ by using a rule $\texttt{l} \rightharpoonup \texttt{r} \mid \texttt{C}$, the pair $(\texttt{p}, \texttt{l}\sigma)$ is returned, which consists of the wrong page together with the corresponding incorrectness symptom.

Algorithm 1 An algorithm for detecting correctness errors in a Web site.

```
1: procedure CORRECTNESS-ERRORS(W, I_N, R)
2:   for all p ∈ W do
3:     for all r ≡ (l ⇀ error |
         X_1 in rexp_1, ..., X_n in rexp_n,
         s_1 = t_1, ..., s_m = t_m) ∈ I_N do
4:       if (p ⇀^σ_{r_u} error) then
5:         If (X_i σ ∈ L(rexp_i), i = 1, ..., n) and
             ((s_j = t_j)σ, j = 1, ..., m, holds in R) then
6:           output("Error: (p,lσ)") end if
7:       end if
8:     end for
9:   end for
10: end procedure
```

Proposition 2 *Let* $\texttt{W}$ *be a Web site, and* $(\texttt{I}_\texttt{N} \uplus \texttt{I}_\texttt{M}, R)$ *be a Web specification. Then, the procedure* CORRECTNESS-ERRORS$(\texttt{W}, \texttt{I}_\texttt{N}, R)$ *terminates, and for each pair* $(\texttt{p}, \texttt{l}\sigma)$*, which is returned,* $\texttt{p}$ *is an incorrect Web page w.r.t.* $(\texttt{I}_\texttt{N} \uplus \texttt{I}_\texttt{M}, R)$ *and* $\texttt{l}\sigma$ *is the corresponding incorrectness symptom.*

6.2 Detecting completeness errors

Essentially, the main idea to diagnose completeness errors is to compute the set of all possible marked expressions that can be derived from $\texttt{W}$ via the completeness rules of a Web specification $(\texttt{I}_\texttt{N} \uplus \texttt{I}_\texttt{M}, R)$ by means of partial rewriting. These marked terms can be thought of as requirements to be fulfilled by $\texttt{W}$. Then, we check whether the computed requirements are satisfied by $\texttt{W}$ using simulation and marking/quantification information.

In summary, the method works in two steps, as described below.

1. Compute the set of requirements $\texttt{Req}_{\texttt{I}_\texttt{M},\texttt{W}}$ for $\texttt{W}$ w.r.t. $\texttt{I}_\texttt{M}$.
2. Check $\texttt{Req}_{\texttt{I}_\texttt{M},\texttt{W}}$ in $\texttt{W}$.

Formally, a *requirement* is a pair $\langle \mu(\texttt{e}), \texttt{q} \rangle$, where $\mu(\texttt{e})$ is a marked term and $\texttt{q} \in \{\texttt{A}, \texttt{E}, \_\}$. A requirement is called *universal* whenever $\texttt{q} = \texttt{A}$, while it is called *existential* when $\texttt{q} = \texttt{E}$. Requirements of the form $\langle \mu(\texttt{e}), \_ \rangle$ are called *non-quantified* requirements.

In order to formalize step 1, we define the following operator.

Definition 10 *Let* $\texttt{T}$ *be a set of marked terms and* $(\texttt{I}_\texttt{N} \uplus \texttt{I}_\texttt{M}, R)$ *be a Web specification. Then, the* immediate completeness requirements *operator*

$$\mathcal{J}_\texttt{M}(\overline{\texttt{T}}) = \overline{\texttt{T}} \cup \{\langle \mu_2(\texttt{s}_2), \texttt{q} \rangle \mid \exists\, \langle \mu_1(\texttt{s}_1), \texttt{q}_1 \rangle \in \overline{\texttt{T}},\ r \equiv (\texttt{l} \rightharpoonup \mu(\texttt{r})\ \langle \texttt{q} \rangle) \in \texttt{I}_\texttt{M}\ \textit{s.t.}\ \mu_1(\texttt{s}_1) \rightharpoonup_r \mu_2(\texttt{s}_2)\}$$

where $\overline{\texttt{T}} = \{\langle \texttt{s}, \_ \rangle \mid \texttt{s} \in \texttt{T}\}$.

Roughly speaking, the operator in Definition 10 computes all the requirements which are obtained by partially rewriting the marked expressions in $\overline{\texttt{T}}$ using the completeness rules of $\texttt{I}_\texttt{M}$, and returns the union of the resulting set and $\overline{\texttt{T}}$. By repeatedly applying this operator, it is possible to compute all marked terms that can be derived from an initial Web site after an arbitrary number of partial rewriting steps. For this purpose, we formalize the *ordinal powers* of the operator $\mathcal{J}_\texttt{M}$ w.r.t. a Web site $\texttt{W}$ as follows: $\mathcal{J}_\texttt{M} \uparrow^\texttt{W} 0 = \overline{\texttt{W}}$, $\mathcal{J}_\texttt{M} \uparrow^\texttt{W} n = \mathcal{J}_\texttt{M}(\mathcal{J}_\texttt{M} \uparrow^\texttt{W} (n-1))$, $n > 0$.

It is immediate to prove that the operator $\mathcal{J}_\texttt{M}$ is continuous on the lattice consisting of the powerset of the requirements ordered by set inclusion. This ensures that a least fixpoint of $\mathcal{J}_\texttt{M}$ exists and can be reached after ω applications of $\mathcal{J}_\texttt{M}$, that is, $\mathcal{J}_\texttt{M} \uparrow^\texttt{W} \omega$ where ω is the first infinite ordinal. The least fixpoint of $\mathcal{J}_\texttt{M}$ contains all the marked expressions derivable from $\texttt{W}$ via $\texttt{I}_\texttt{M}$ along with their quantification information.

Now, recalling the interpretation of the completeness rules of the Web site specification given in Sect. 4, marked terms derived by the application of a completeness rule must be recognized as (part of) some Web page

in the Web site. Therefore, the expressions in the least fixpoint of $\mathcal{J}_{\mathtt{M}}$ provide information that must occur in $\mathtt{W}$. Thus, since the pages in $\mathtt{W}$ trivially occur in the Web site $\mathtt{W}$, we define the *set of requirements* for $\mathtt{W}$ w.r.t. $\mathtt{I}_{\mathtt{M}}$ as

$$\mathtt{Req}_{\mathtt{M},\mathtt{W}} = \mathit{lfp}(\mathcal{J}_{\mathtt{M}}) \setminus \overline{\mathtt{W}}$$

where $\mathit{lfp}(\mathcal{J}_{\mathtt{M}})$ is the least fixpoint of the operator $\mathcal{J}_{\mathtt{M}}$. Observe that the set $\mathtt{Req}_{\mathtt{M},\mathtt{W}}$ does not contain any non-quantified requirement.

Example 12 Consider the Web specification $(\mathtt{I}_{\mathtt{N}} \uplus \mathtt{I}_{\mathtt{M}}, R)$ of Example 6 and the Web site $\mathtt{W}$ of Example 4. Then, the set of computed requirements $\mathtt{Req}_{\mathtt{M},\mathtt{W}}$ is

$$\begin{aligned}\{\ &\langle \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{mariorossi}), \mathtt{status}), \mathtt{E}\rangle,\\ &\langle \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{francabianchi}), \mathtt{status}), \mathtt{E}\rangle,\\ &\langle \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{giulioverdi}), \mathtt{status}), \mathtt{E}\rangle,\\ &\langle \underline{\mathtt{hpage}}(\underline{\mathtt{status}}(\underline{\mathtt{professor}}), \mathtt{teaching}), \mathtt{A}\rangle,\\ &\langle \underline{\mathtt{member}}(\mathtt{name}(\mathtt{mario}), \mathtt{surname}(\mathtt{rossi})), \mathtt{E}\rangle,\\ &\langle \underline{\mathtt{member}}(\mathtt{name}(\mathtt{anna}), \mathtt{surname}(\mathtt{gialli})), \mathtt{E}\rangle,\\ &\langle \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{annagialli}), \mathtt{status}), \mathtt{E}\rangle,\\ &\langle \underline{\mathtt{cpage}}(\mathtt{title}(\mathtt{Algebra})), \mathtt{E}\rangle\ \}\end{aligned}$$

Clearly, the fixpoint of $\mathcal{J}_{\mathtt{M}}$ (and hence $\mathtt{Req}_{\mathtt{M},\mathtt{W}}$) for an arbitrary Web specification might be infinite. Consider for instance the following example.

Example 13 Let $\mathtt{W} \equiv \{\mathtt{h}(\mathtt{g}(0), \mathtt{f}(0))\}$ be a Web site and

$$\mathtt{I}_{\mathtt{M}} \equiv \{\mathtt{h}(\mathtt{g}(\mathtt{X})) \rightharpoonup \mathtt{h}(\mathtt{g}(\mathtt{g}(\mathtt{X})))\langle \mathtt{q}\rangle\}$$

be a set of completeness rules of a Web specification $\mathtt{S}$. Then,

$$\begin{aligned}\mathtt{Req}_{\mathtt{M},\mathtt{W}} = \{&\langle h(g(g(0))), q\rangle, \langle h(g(g(g(0)))), q\rangle,\\ &\langle h(g(g(g(g(0))))), q\rangle, \ldots\}\end{aligned}$$

is an infinite set of requirements.

Fortunately, the computation of the set of requirements is finite for some interesting classes of Web specifications. Trivially, non-recursive specifications allow to reach $\mathit{lfp}(\mathcal{J}_{\mathtt{M}})$ after a finite number of applications of $\mathcal{J}_{\mathtt{M}}$, i.e., $\mathit{lfp}(\mathcal{J}_{\mathtt{M}}) = \mathcal{J}_{\mathtt{M}} \uparrow^{\mathtt{W}} k$, $k \in \mathbb{N}$. However, non-recursive definitions are not expressive enough for verification purposes, since some relevant conditions about Web sites cannot be formalized without resorting to recursion; e.g., some properties stated in Example 6 cannot be formulated by using a non-recursive specification.

In the following, we ascertain two important classes of recursive Web specifications whose set of requirements is finite. Basically, the idea is to consider those specifications in which the computation of the least fixpoint only generates marked expressions whose size is bounded [2]. We believe that such classes are expressive enough, since they can model the most common completeness as well as correctness requirements that one can desire for a given Web site. Moreover, these classes allow us to generate only finite sets of conditions to be verified, which is *crucial* for the effectiveness of the whole method and *reasonable* in this context as Web sites are finite collections of semistructured data with an associated finite semantics.

The following definition formalizes the class of the *bounded* Web specifications.

Definition 11 *A Web specification* $(\mathtt{I}_{\mathtt{N}} \uplus \mathtt{I}_{\mathtt{M}}, R)$ *is* bounded *iff, for each* $\mathtt{l} \equiv (V_1, E_1, r_1, \mathrm{label}_1) \in \mathtt{Lhs}_{\mathtt{M}}$, $\mathtt{r} \equiv (V_2, E_2, r_2, \mathrm{label}_2) \in \mathtt{Rhs}_{\mathtt{M}}$ *and each minimal injective simulation* $\mathbf{S}$ *of* $\mathtt{l}$ *in* $\mathtt{r}_{|v}$ *w.r.t.* $\sim$, $v \in V_2$, *the following properties hold*

1. *if* $v_2 \in \pi(\mathbf{S})$ *and* $\mathrm{label}_2(v_2) \in \mathrm{Var}(\mathtt{r}_{|v})$, *then for all* $v_1 \in V_1$ *s.t.* $\mathrm{label}_1(v_1) \in \mathrm{Var}(\mathtt{l})$, $\mathrm{depth}(\mathtt{r}_{|v}, v_2) = \mathrm{depth}(\mathtt{l}, v_1)$.
2. *For each* $\mathtt{r} \in \mathtt{Rhs}_{\mathtt{M}}$, $\mathtt{r}$ *does not contain any symbol of* Σ_R.

Roughly speaking, Definition 11 states that, whenever the left-hand side $\mathtt{l}$ of a rule is simulated by (a subterm of) the right-hand side $\mathtt{r}$ of a (possibly different) rule, then no variables in the recognized substructure of $\mathtt{r}$ must be located at positions which are deeper than all the positions of the variables in $\mathtt{l}$. Moreover, bounded Web specifications do not allow function calls in the completeness rules. Let us see an example.

Example 14 Consider again the set of completeness rules $\mathtt{I}_{\mathtt{M}}$ of Web specification $\mathtt{S}$ in Example 13. The left-hand side of the rule $\mathtt{h}(\mathtt{g}(\mathtt{X})) \rightharpoonup \mathtt{h}(\mathtt{g}(\mathtt{g}(\mathtt{X})))\langle \mathtt{q}\rangle$ is simulated by its own right-hand side. Moreover, variable $\mathtt{X}$ in the right-hand side is located at depth 3, while the unique variable in the left-hand side is at depth 2. Thus, $\mathtt{S}$ is not bounded.

Now, take into account the Web specification $\mathtt{S}' \equiv (\mathtt{I}'_{\mathtt{N}} \uplus \mathtt{I}'_{\mathtt{M}}, \emptyset)$, whose set of completeness rules is

$$\begin{aligned}\mathtt{I}'_{\mathtt{M}} \equiv \{\ &\mathtt{m}(\mathtt{n}(\mathtt{X})) \rightharpoonup \mathtt{h}(\mathtt{n}(\mathtt{X}), \mathtt{s}(\mathtt{s}(\mathtt{X})))\langle \mathtt{q}'\rangle,\\ &\mathtt{h}(\mathtt{n}(\mathtt{X})) \rightharpoonup \mathtt{m}(\mathtt{n}(\mathtt{X}), \mathtt{t})\langle \mathtt{q}''\rangle\},\quad \mathtt{q}', \mathtt{q}'' \in \{\mathtt{A}, \mathtt{E}\}.\end{aligned}$$

Then, $\mathtt{m}(\mathtt{n}(\mathtt{X}))$ is simulated by $\mathtt{m}(\mathtt{n}(\mathtt{X}), \mathtt{t})$ and $\mathtt{h}(\mathtt{n}(\mathtt{X}))$ is simulated by the term $\mathtt{h}(\mathtt{n}(\mathtt{X}), \mathtt{s}(\mathtt{s}(\mathtt{X})))$. In both cases, variables occurring in the substructures of the right-hand sides which are recognized by simulation and variables of the respective left-hand sides are located at the same depth. Therefore, the Web specification $\mathtt{S}'$ is bounded.

For bounded Web specifications, the least fixpoint of the operator $\mathcal{J}_{\mathtt{M}}$ is finite as stated by the next proposition. This provides an effective method for computing the set of requirements $\mathtt{Req}_{\mathtt{M},\mathtt{W}}$ for this class of specifications.

Proposition 3 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded Web specification and* $\mathtt{W}$ *be a Web site. Then, there exists* $k \in \mathbb{N}$ *such that* $lfp(\mathcal{J}_\mathtt{M}) = \mathcal{J}_\mathtt{M} \uparrow^\mathtt{W} k$.

Let us now introduce a more general class of Web specifications in which defined functions can be invoked in the rhs's of the completeness rules. In order to keep the size of the derived terms bounded, the key idea is to ensure that function calls do not generate infinite data structures.

Given a Web specification $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$, we say that the completeness rule $(\mathtt{l} \rightharpoonup \mu(\mathtt{r})\ \langle \mathtt{q} \rangle) \in \mathtt{I_M}$ is *function-dependent* w.r.t. R iff a function call $\mathtt{f}(\mathtt{t_1}, \ldots, \mathtt{t_n}) \in \tau(\Sigma_R, \mathcal{V})$ occurs in $\mathtt{r}$. Otherwise we say that rule r is *function-independent* w.r.t. R. Given a function-dependent rule r, we obtain the *function-independent version* of r w.r.t. R, and we denote it by r^*, by replacing all the function calls in the right-hand side of r with fresh variables not occurring in the rule.

Example 15 Consider the function-dependent rule of Example 6

$$\mathtt{member(name(X), surname(Y))} \rightharpoonup \underline{\mathtt{hpage}}(\mathtt{fullname(append(X, Y)), status})\ \langle \mathtt{E} \rangle.$$

The function-independent version of the considered rule is

$$\mathtt{member(name(X), surname(Y))} \rightharpoonup \underline{\mathtt{hpage}}(\mathtt{fullname(Z), status})\ \langle \mathtt{E} \rangle.$$

Definition 12 *A Web specification* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *is* bounded* *iff*

1. $(\mathtt{I_N} \uplus \{r \in \mathtt{I_M} \mid r \text{ is function-independent w.r.t. } R\}, R)$ *is bounded.*
2. *For each function-dependent rule* $r \in \mathtt{I_M}$ *w.r.t.* R *and* $\mathtt{l} \in \mathtt{Lhs_M}$
   (a) *There exists no minimal, injective simulation of* $\mathtt{l}$ *in any subterm of the right-hand side of* r^* *w.r.t.* $\sim$.
   (b) *There exists no minimal, injective simulation of any subterm of the right-hand side of* r^* *in* $\mathtt{l}$ *w.r.t.* $\sim$.
3. $\{\mathtt{f} \mid \mathtt{f}(\mathtt{t_1}, \ldots, \mathtt{t_n}) \rightharpoonup \mu(\mathtt{r}) \in \mathtt{I_m}\} \cap \Sigma_R = \emptyset$.

Roughly speaking, by Definition 12, the rhs's of function-dependent completeness rules do not introduce terms which can be partially rewritten in a subsequent step. This suffices to ensure the finiteness of all partial rewriting sequences. Now, it is immediate to prove the finiteness of the least fixpoint of the $\mathcal{J}_\mathtt{M}$ operator for bounded* Web specifications.

The following proposition generalizes Proposition 3 for bounded* Web specifications.

Proposition 4 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded* Web specification and* $\mathtt{W}$ *be a Web site. Then, there exists* $k \in \mathbb{N}$ *such that* $lfp(\mathcal{J}_\mathtt{M}) = \mathcal{J}_\mathtt{M} \uparrow^\mathtt{W} k$.

Let us see an example.

Example 16 Let us consider the Web specification of Example 6. Then, we can easily check that it is bounded*. Moreover, the least fixpoint of $\mathcal{J}_\mathtt{M}$ is finite as witnessed by Example 12.

Example 17 Consider now the following Web specification.

$$\begin{aligned} \mathtt{I_N} &= \emptyset \\ \mathtt{I_M} &= \{\mathtt{h(c(X))} \rightharpoonup \mathtt{h(g(X))}\} \\ R &= \{\mathtt{g(0)} \to \mathtt{c(c(0))}, \mathtt{g(c(X))} \to \mathtt{c(g(X))}\} \end{aligned}$$

The function-independent version of $\mathtt{h(c(X))} \rightharpoonup \mathtt{h(g(X))}$ is $\mathtt{h(c(X))} \rightharpoonup \mathtt{h(Z)}$, and there exists a minimal, injective simulation of $\mathtt{h(Z)}$ in $\mathtt{h(c(X))}$ w.r.t. $\sim$. Therefore, the considered Web specification is not bounded*. Actually, we can generate the following infinite partial rewrite sequence:

$$\mathtt{h(c(0))} \rightharpoonup \mathtt{h(c(c(0)))} \rightharpoonup \mathtt{h(c(c(c(0))))} \ldots$$

Now, we are ready to formalize step 2, that is, checking the computed completeness requirements in a given Web site. To accomplish this task, we first use simulation for checking whether (the marked part of) a requirement is embedded into some Web page of the considered site, and then consider the quantification attributes in order to diagnose completeness errors.

Definition 13 *Let* $\mathtt{W}$ *be a Web site,* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification and* $\mathtt{Req_{M,W}}$ *be the set of requirements for* $\mathtt{W}$ *w.r.t.* $\mathtt{I_M}$. *Let* $\langle \mu(\mathtt{e}), \mathtt{q} \rangle \in \mathtt{Req_{M,W}}$. *The* test set *w.r.t.* $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$ *is defined as*

$$\mathtt{TEST}_{\langle \mu(\mathtt{e}), \mathtt{q} \rangle} = \{\mathtt{p} \equiv (V, E, r, \text{label}) \in \mathtt{W} \mid \exists \text{ a minimal injective simulation of } \text{mark}(\mathtt{e}, \mu) \text{ in } \mathtt{p}_{|v} \text{ w.r.t. } \sim, \text{ with } v \in V\}.$$

Roughly speaking, this definition allows us to compute a subset of the Web site containing all the Web pages which simulate the marked part of a given requirement. These Web pages might be incomplete w.r.t. the Web specification, since they may not contain the considered requirement. Let us see an example.

Example 18 Let us consider the completeness rule r

$$\mathtt{hpage(status(professor))} \rightharpoonup \underline{\mathtt{hpage}}(\underline{\mathtt{status}}(\underline{\mathtt{professor}}), \mathtt{teaching})$$

and the Web site $\mathtt{W}$ of Example 4. Rule r allows us to check whether Web pages of the professors contain

some teaching information. In order to do this, we use the marking information in the rhs of r to select the professor Web pages. Let us consider the requirement $\langle \mu_1(\mathtt{e}_1), \mathtt{A}\rangle \equiv \langle \underline{\mathtt{hpage}}(\underline{\mathtt{status}}(\underline{\mathtt{professor}}, \mathtt{teaching}), \mathtt{A}\rangle$, which can be derived from W by means of r. By applying Definition 13, we get the following test set $\mathtt{TEST}_{\langle \mu_1(\mathtt{e}_1),\mathtt{A}\rangle}$

```
{(2) hpage(fullname(mariorossi),phone(3333),
           status(professor),
           hobbies(hobby(reading),
                   hobby(gardening))),
 (4) hpage(fullname(annagialli),
           status(professor),
           blink(phone(4444)),
           teaching(courselink(
                url(http://www.algebra.org),
                urlname(Algebra))))}
```

which contains the two professor Web pages where the computed completeness requirement must be checked.

In the following, we consider completeness errors which refer to incomplete and/or missing Web pages. We distinguish two cases: the former allows us to discover whether a universal requirement is not fulfilled by a given Web site, while the latter recognizes unsatisfied existential requirements. In both cases, our analysis provides the missing/incomplete Web pages which are associated with those requirements.

Definition 14 *Let* W *be a Web site,* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification and* $\mathtt{Req}_{\mathtt{M,W}}$ *be the set of requirements for* W *w.r.t.* $\mathtt{I_M}$. *Let* $\mathtt{re} \equiv \langle \mu(\mathtt{e}), \mathtt{A}\rangle \in \mathtt{Req}_{\mathtt{M,W}}$ *be a universal requirement. Then,* re *is not* satisfied *in* W *if one of the following conditions hold:*

1. $\mathtt{TEST}_{\langle \mu(\mathtt{e}),\mathtt{A}\rangle} = \emptyset$.
2. *There exists* $\mathtt{p} \equiv (V, E, r, \text{label}) \in \mathtt{TEST}_{\langle \mu(\mathtt{e}),\mathtt{A}\rangle}$ *s.t. no minimal, injective simulation of* e *in* $\mathtt{p}_{|v}$ *w.r.t.* $\sim$, *with* $v \in V$, *exists.*

Vice versa, a universal requirement re is satisfied whenever it is possible to recognize re inside any Web page of the corresponding test set $\mathtt{TEST}_{\mathtt{re}}$. Let us clarify Definition 14 by an example.

Example 19 Consider the Web site W of Example 4 and the universal requirement

$$\langle \mu_1(\mathtt{e}_1), \mathtt{A}\rangle \equiv \langle (\underline{\mathtt{hpage}}(\underline{\mathtt{status}}(\underline{\mathtt{professor}}), \mathtt{teaching})), \mathtt{A}\rangle$$

belonging to the set of requirements $\mathtt{Req}_{\mathtt{M,W}}$ of Example 12. The requirement simply states that *all* professor's home pages must contain teaching information.

The test set associated with $\langle \mu_1(\mathtt{e}_1), \mathtt{A}\rangle, \mathtt{TEST}_{\langle \mu_1(\mathtt{e}_1),\mathtt{A}\rangle}$, was computed in Example 18 and contains all the professor's Web pages of the considered site. Now, by applying Definition 14, we detect that $\langle \mu_1(\mathtt{e}_1), \mathtt{A}\rangle$ is not satisfied by W, since there does not exist any minimal, injective simulation of $\mathtt{e}_1$ in (a subterm of) Web page (2) w.r.t. $\sim$.

In fact, Web page (2) lacks teaching information.

Finally, an existential requirement re is fulfilled, if it is recognized inside (at least) a Web page which belongs to the test set $\mathtt{TEST}_{\mathtt{re}}$.

Definition 15 *Let* W *be a Web site,* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification and* $\mathtt{Req}_{\mathtt{M,W}}$ *be the set of requirements for* W *w.r.t.* $\mathtt{I_M}$. *Let* $\mathtt{re} \equiv \langle \mu(\mathtt{e}), \mathtt{E}\rangle \in \mathtt{Req}_{\mathtt{M,W}}$ *be an existential requirement. Then,* re *is not* satisfied *in* W *if one of the following conditions hold:*

1. $\mathtt{TEST}_{\langle \mu(\mathtt{e}),\mathtt{E}\rangle} = \emptyset$.
2. *For each* $\mathtt{p} \equiv (V, E, r, \text{label}) \in \mathtt{TEST}_{\langle \mu(\mathtt{e}),\mathtt{E}\rangle}$ *no minimal, injective simulation of* e *in* $\mathtt{p}_{|v}$ *w.r.t.* $\sim$, *with* $v \in V$, *exists.*

Example 20 Consider the Web site W of Example 4 and the existential requirement

$$\langle \mu_2(\mathtt{e}_2), \mathtt{E}\rangle \equiv \langle \underline{\mathtt{cpage}}(\mathtt{title}(\mathtt{Algebra})), \mathtt{E}\rangle$$

belonging to the set of requirements $\mathtt{Req}_{\mathtt{M,W}}$ of Example 12. Since $\mathtt{TEST}_{\langle \mu_2(\mathtt{e}_2),\mathtt{E}\rangle}$ is empty, by Definition 15, re is not satisfied in W. This implies that a Web page containing some information about the Algebra course should be provided.

Consider now the following existential requirements in $\mathtt{Req}_{\mathtt{M,W}}$

$$\langle \mu_3(\mathtt{e}_3), \mathtt{E}\rangle \equiv \langle \underline{\mathtt{member}}(\mathtt{name}(\mathtt{anna}), \mathtt{surname}(\mathtt{gialli})), \mathtt{E}\rangle$$
$$\langle \mu_4(\mathtt{e}_4), \mathtt{E}\rangle \equiv \langle \underline{\mathtt{hpage}}(\mathtt{fullname}(\mathtt{giulioverdi}), \mathtt{status}), \mathtt{E}\rangle.$$

We have that $\mathtt{TEST}_{\langle \mu_3(\mathtt{e}_3),\mathtt{E}\rangle}$ just includes Web page (1) of W, while $\mathtt{TEST}_{\langle \mu_4(\mathtt{e}_4),\mathtt{E}\rangle}$ contains Web pages (2), (3) and (4). For both requirements, there is no Web page p in the test sets such that a minimal, injective simulation of $\mathtt{e}_3$ (respectively, $\mathtt{e}_4$) in (a subterm of) p w.r.t $\sim$ exists. Therefore, the Web site W does not meet the given requirements. More precisely, the former unsatisfied requirement states that an entry for *Anna Gialli* must be introduced in the group member list, and the latter detects that the home page of *Giulio Verdi* is missing.

The requirements which are not fulfilled can be considered as *incompleteness symptoms*. This allows us not only to locate bugs and inconsistencies w.r.t. a given specification, but also to easily repair them by comparing incomplete pages to unsatisfied requirements, since the latter ones provide the missing information which is needed to complete the erroneous Web pages.

An algorithm for detecting incomplete information can be defined as follows. First of all, we generate the requirements to be checked by computing the fixpoint of the operator $\mathcal{J}_{\mathtt{M}}$. Next, for each (universal/existential) requirement, the corresponding test set is produced and then checked in order to analyze which requirements are not fulfilled. The analysis is based on Definition 14 (resp., Definition 15) which formalizes the notion of universal (resp., existential) requirement satisfiability. Finally, incompleteness symptoms are returned, which are needed to help the user to locate the missing/incomplete pages, whenever a requirement is not verified.

The basic procedure is sketched in Algorithm 2, which takes as input a Web site $\mathtt{W}$, a set of completeness rules $\mathtt{I}_{\mathtt{M}}$, and a canonical TRS R.

Algorithm 2 An algorithm for detecting completeness errors.

```
1: procedure COMPLETENESS-ERRORS(W, I_M, R)
2:     Req_{M,W} ← lfp(J_M) \ W
3:     for all ⟨μ(e), q⟩ ∈ Req_{M,W} do
4:         TEST_⟨μ(e),q⟩ ← {p ≡ (V, E, r, label) ∈ W |
                mark(e, μ) ≅ p|_v, v ∈ V}
5:         if TEST_⟨μ(e),q⟩ = ∅ then
6:             output("Error: A Web page containing e must occur
                   in Web site W!")
7:         end if
8:         case q
9:             q = A :
10:                if ∃p ≡ (V, E, r, label) ∈ TEST_⟨μ(e),A⟩ s.t. e ≱ p|_v,
                       with v ∈ V then
11:                    output("Error: e must occur in all Web pages
                           of TEST_⟨μ(e),A⟩!")
12:                end if
13:            q = E :
14:                if ∀p ≡ (V, E, r, label) ∈ TEST_⟨μ(e),E⟩, e ≱ p|_v,
                       with v ∈ V then
15:                    output("Error: e must occur in at least one
                           Web page of TEST_⟨μ(e),E⟩!")
16:                end if
17:        end case
18:    end for
19: end procedure
```

Proposition 5 *Let* $\mathtt{W}$ *be a Web site,* $(\mathtt{I}_{\mathtt{N}} \uplus \mathtt{I}_{\mathtt{M}}, R)$ *be a bounded* Web specification, and* $\mathtt{Req}_{\mathtt{M},\mathtt{W}}$ *be the set of requirements for* $\mathtt{W}$ *w.r.t.* $\mathtt{I}_{\mathtt{M}}$*. Then, the procedure* COMPLETENESS-ERRORS$(\mathtt{W}, \mathtt{I}_{\mathtt{M}}, R)$ *terminates. Moreover, for each error message regarding term* $\mathtt{e}$*, which is returned by the procedure, there exists* $\langle \mu(\mathtt{e}), \mathtt{q} \rangle \in \mathtt{Req}_{\mathtt{M},\mathtt{W}}$*, with* $\mathtt{q} \in \{\mathtt{A}, \mathtt{E}\}$*, which is not satisfied in* $\mathtt{W}$*.*

7 The GVERDI system

The basic methodology presented so far has been implemented in the prototype system GVERDI (VErification and Rewriting for Debugging Internet sites) which has been written in Haskell (GHC v6.2.2) and is publicly available together with a set of tests at

`http://www.dsic.upv.es/users/elp/GVerdi/`

A short system description can be found in [3,6], while a thorough user's manual of the tool can be retrieved at the URL mentioned above.

The implementation consists of approximately 1,100 lines of source code. It includes a parser for semistructured expressions (i.e., XML/XHTML documents) and Web specifications, and several modules implementing the partial rewriting mechanism, the verification technique, and the graphical user interface. A snapshot of the running system is shown in Fig. 8.

The system allows the user to load a Web site 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. We have tested the system on several XHTML Web sites and XML data collections which can be found at the URL address mentioned above. For instance, we checked the Web site of the Computational Logic Group of the University of Udine which is available at `http://www.dimi.uniud.it/clg`. It contains about 20 Web pages concerning publications, people, and projects of the group. In each considered test case, we have been able to detect the errors (i.e., missing and incomplete Web pages) efficiently.

8 Conclusions

Conceiving and maintaining Web sites is a difficult task. In this paper, we provide a rewriting-based, formal

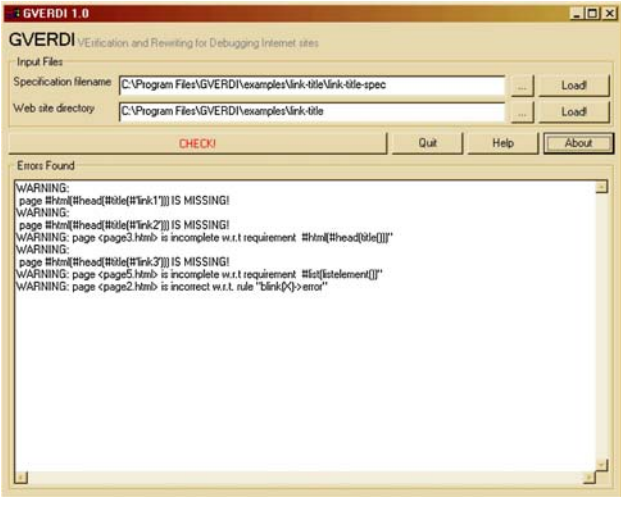

Fig. 8 A screenshot of the system

specification language which can be used to impose properties both on the structure (syntactic properties) and on the contents (semantic properties) of Web sites. Some XML schema languages such as XLINKIT [19] and Schematron [35] can express some of the semantic constraints we consider; however, our rule specification language, which offers the expressiveness and computational power of functions, is richer and can be appreciated much by users who prefer to avoid the encumbrances of DTDs and XML rule languages. The computation mechanism underlying our language is based on a novel rewriting-like technique, called *partial rewriting*, in which the traditional pattern matching mechanism is replaced by tree *simulation* [24]. In our methodology, Web sites are automatically checked w.r.t. a given Web specification in order to detect incorrect, incomplete, and missing Web pages. Moreover, by analyzing the error symptoms, we are also able to find out the missing information needed to repair the Web site and the wrong information we have to remove or to fix. We have also discussed some implementation details of the system GVERDI, a prototype implementation of the verification framework which can help Web administrators to design, check, and maintain their Web sites.

Let us conclude by mentioning some directions for future work. We are currently extending our framework in order to provide a method for synthesizing the marking information semi-automatically (currently, marks are provided by the user). Besides, we plan to extend the GVERDI system with a graphical, high-level language for the specification of the conditions to be checked in order to make the system easy to use and intuitive even for the users who have no expertise in formal methods. Finally, in order to consider semistructured documents containing cycles, which could be easily generated —-for instance—- by using ID and IDREF XML attributes, a graph rewriting extension of our framework is also envisaged. In fact, trees are not enough to model loops, we believe that graph structures and the associated rewriting framework are essential to deal with such circular documents and to enhance the expressiveness of our specification language.

Acknowledgments We gratefully thank Javier García-Vivó for his help with the implementation of GVERDI and the experiments with the system.This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grant TIN 2004-7943-C04-02, by ICT for EU-India Cross Cultural Dissemination Project under grant ALA/95/23/2003/077-054, and by Generalitat Valenciana under grant GR03/025.

Appendix: Proofs of the technical results

In this appendix we provide the proofs of the main technical results of the paper.

Proposition 1 *Let* $\mathtt{s}_1 \equiv (V_1, E_1, r_1, \mathrm{label}_1)$, $\mathtt{s}_2 \equiv (V_2, E_2, r_2, \mathrm{label}_2)$ *be two Web page templates in* $\tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$. *A minimal page simulation* $\mathbf{S}$ *of* $\mathtt{s}_1$ *in* $\mathtt{s}_2$ *w.r.t.* $\sim$ *is a mapping* $\mathbf{S} : V_1 \to V_2$.

Proof Let $\mathtt{s}_1 \equiv (V_1, E_1, r_1, \mathrm{label}_1)$, $\mathtt{s}_2 \equiv (V_2, E_2, r_2, \mathrm{label}_2) \in \tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$. By Conditions 1 and 3 of Definition 5, and by using the fact that $\mathtt{s}_1$ has an underlying tree structure (in particular, it is a connected graph), we have that

$\forall\, v_1 \in V_1,\ \exists\, v_2 \in V_2$ such that $(v_1, v_2) \in \mathbf{S}$.

Moreover, the minimality of $\mathbf{S}$ ensures that

$\forall\, v_1 \in V_1,\ \exists$ a *unique* $v_2 \in V_2$ such that $(v_1, v_2) \in \mathbf{S}$

which implies that $\mathbf{S}$ is a mapping from V_1 to V_2.

Proposition 2 *Let* $\mathtt{W}$ *be a Web site, and* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification. Then, the procedure* CORRECTNESS-ERRORS$(\mathtt{W}, \mathtt{I_N}, R)$ *terminates, and for each pair* $(\mathtt{p}, \mathtt{l}\sigma)$, *which is returned,* $\mathtt{p}$ *is an incorrect Web page w.r.t.* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *and* $\mathtt{l}\sigma$ *is the corresponding incorrectness symptom.*

Proof First of all, let us prove the termination of the algorithm. The outer loop (lines 2–9) is executed $|W|$ times. The inner loop (lines 3–8) is executed $|I_{\mathtt{N}}|$ times. As for the partial rewrite step $\mathtt{p} \rightharpoonup^{\sigma}_{r_u} \mathtt{error}$ in line 4, it terminates, because it is just an application of the simulation algorithm [24], which terminates. The evaluation of the condition in line 5 terminates, since:

1. The problem of evaluating $\mathtt{X}_i\sigma \in \mathcal{L}(\mathtt{rexp}_i)$, $i = 1, \ldots, n$ boils down to the membership problem in regular languages, which is decidable (see [25]), so there exists an effective procedure which tests $(\mathtt{X}_i\sigma \in \mathcal{L}(\mathtt{rexp}_i), i = 1, \ldots, n)$ in finite time.
2. Evaluating the condition $[(\mathtt{s}_j = \mathtt{t}_j)\sigma, j = 1, \ldots, m,$ holds in $R]$ trivially terminates, since R is canonical (in particular, it is terminating).

The output command in line 6 clearly terminates. Summing up, we execute the block of terminating instructions, which is included in lines 4–7, $|W| * |I_{\mathtt{N}}|$ times, so the whole procedure terminates.

Let us prove the partial correctness of the algorithm, that is, if the procedure terminates producing the outcome $(\mathtt{p}, \mathtt{l}\sigma)$, then $\mathtt{p} \in \mathtt{W}$ is incorrect w.r.t. $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ and $\mathtt{l}\sigma$ is an incorrect symptom for $\mathtt{p}$.

Let $(\mathtt{p}, \mathtt{l}\sigma)$ be an output of the procedure. This implies that (1) $\mathtt{p} \rightharpoonup^{\sigma}_{r_u} \mathtt{error}$ for some $r \equiv (\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{X}_1 \,\mathtt{in}\, \mathtt{rexp}_1, \ldots, \mathtt{X}_n \,\mathtt{in}\, \mathtt{rexp}_n, \mathtt{s}_1 = \mathtt{t}_1, \ldots, \mathtt{s}_m = \mathtt{t}_m) \in \mathtt{I_N}$ (see line 4); (2) for each $i = 1, \ldots, n$, $(\mathtt{X}_i\sigma \in \mathcal{L}(\mathtt{rexp}_i)$, $i = 1, \ldots, n)$ (see line 5); and (3) for each $j = 1, \ldots, m$, $(\mathtt{s}_j = \mathtt{t}_j)\sigma$ holds in R (see line 5). Now, by simply applying Definition 9 , we get that $\mathtt{p} \in \mathtt{W}$ is incorrect w.r.t. $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ and $\mathtt{l}\sigma$ is an incorrect symptom for $\mathtt{p}$.

In order to prove Proposition 3, we need the following auxiliary definitions and results.

Definition 16 *Given a term $t \in \tau(\Sigma, \mathcal{V})$, the* height *of t, height(t), is defined as follows.*

$$\text{height}(t) = \begin{cases} 0 & \text{if } t \equiv X \in \mathcal{V} \text{ or } t \equiv c \in \tau(\Sigma, \mathcal{V}) \\ 1 + \max\{\text{height}(t_i) | i = 1, \ldots, n\} & \\ & \text{if } t \equiv f(t_1, \ldots, t_n) \in \tau(\Sigma, \mathcal{V}) \end{cases}$$

We can lift the notion of height to substitutions in the following way.

Definition 17 *Given a substitution* $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$, *the* height *of* σ, $\text{height}(\sigma)$, *is defined as follows.*

$$\text{height}(\sigma) = \max\{\text{height}(t_i) \mid i = 1, \ldots, n\}$$

Now we can prove two useful technical results.

Proposition 6 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded Web specification and*

$$\mu_0(\mathtt{s}_0) \rightharpoonup_{r_0}^{\sigma_0} \mu_1(\mathtt{s}_1) \rightharpoonup_{r_1}^{\sigma_1} \mu_2(\mathtt{s}_2) \rightharpoonup_{r_2}^{\sigma_2} \ldots$$

be a partial rewrite sequence, where $r_i \in \mathtt{I_M}$, $i = 0, 1, 2, \ldots$ *Then, for each* $\mathtt{s}_i$, $i = 1, 2, \ldots$,

$$\mathtt{s}_i \equiv \mathtt{r}_{i-1}\sigma_{i-1}$$

where $\mathtt{r}_{i-1}$ *is the right-hand side of the rule* r_{i-1}.

Proof It directly comes from Definition 8 and by the fact that $\mathit{Reduce}(\mathtt{r}\sigma, R) = \mathtt{r}\sigma$ when considering bounded Web specification, since no function of R can appear in the right-hand sides of the completeness rules.

In other words, Proposition 6 states that each term occurring in a partial rewrite sequence is an instance of the right-hand side of some completeness rule.

Proposition 7 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded Web specification and*

$$\mu_0(\mathtt{s}_0) \rightharpoonup_{r_0}^{\sigma_0} \mu_1(\mathtt{s}_1) \rightharpoonup_{r_1}^{\sigma_1} \mu_2(\mathtt{s}_2) \rightharpoonup_{r_2}^{\sigma_2} \ldots$$

be a partial rewrite sequence, where $r_i \in \mathtt{I_M}$, $i = 0, 1, 2, \ldots$ *Then, for each* $\mathtt{s}_i$, $i = 1, 2, \ldots$,

$$\text{height}(\mathtt{s}_i) \leq \text{height}(\mathtt{r}_{i-1}) + \text{height}(\sigma_{i-1})$$

where $\mathtt{r}_{i-1}$ *is the right-hand side of the rule* r_{i-1}.

Proof It is a direct consequence of Proposition 6 Definition 16 and Definition 17.

Lemma 1 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded Web specification and*

$$\mu_0(\mathtt{s}_0) \rightharpoonup_{r_0}^{\sigma_0} \mu_1(\mathtt{s}_1) \rightharpoonup_{r_1}^{\sigma_1} \mu_2(\mathtt{s}_2) \rightharpoonup_{r_2}^{\sigma_2} \ldots$$

be a (possibly infinite) partial rewrite sequence, where $r_i \in \mathtt{I_M}$, $i = 0, 1, 2, \ldots$ *Then, for each* $\mathtt{s}_i$, $i = 1, 2, \ldots$,

$$\text{height}(\mathtt{s}_i) \leq \text{height}(\mathtt{r}_{i-1}) + \text{height}(\sigma_0)$$

where $\mathtt{r}_{i-1}$ *is the right-hand side of rule* r_{i-1}.

Proof By contradiction, let $\mathtt{s}_{k'}$ be the first term appearing in the partial rewrite sequence such that

$$\text{height}(\mathtt{s}_{k'}) > \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_0),$$

where $\mathtt{r}_{k'-1}$ is the right-hand side of the rule $r_{k'-1}$. Clearly, $k' > 1$, as the claim trivially holds for $k' = 1$. Moreover, for each $j = 1, \ldots k' - 1$, we have

$$\text{height}(\mathtt{s}_j) \leq \text{height}(\mathtt{r}_{j-1}) + \text{height}(\sigma_0).$$

Now, let us focus on the partial rewrite step

$$\mathtt{s}_{k'-1} \rightharpoonup_{r_{k'-1}}^{\sigma_{k'-1}} \mathtt{s}_{k'}.$$

By Proposition 6, $\mathtt{s}_{k'-1} \equiv \mathtt{r}_{k'-2}\sigma_{k'-2}$. So, we have

$$\text{height}(\mathtt{s}_{k'-1}) \leq \text{height}(\mathtt{r}_{k'-2}) + \text{height}(\sigma_{k'-2}).$$

And, also $\text{height}(\sigma_{k'-2}) \leq \text{height}(\sigma_0)$.

Again, by Proposition 6, $\mathtt{s}_{k'} \equiv \mathtt{r}_{k'-1}\sigma_{k'-1}$. And, hence,

$$\text{height}(\mathtt{s}_{k'}) \leq \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_{k'-1}) \quad \text{by Proposition 7.}$$

Here, we distinguish two cases.
Since $\mathtt{s}_{k'-1} \equiv \mathtt{r}_{k'-2}\sigma_{k'-2}$, the partial rewrite step $\mathtt{s}_{k'-1} \rightharpoonup_{r_{k'-1}}^{\sigma_{k'-1}} \mathtt{s}_{k'}$ can be given on a vertex in $\mathtt{r}_{k'-1}$ (Case 1) or on a vertex belonging to a term in $\{t \mid X/t \in \sigma_{k'-2})\}$ (Case 2).

Case 1. Since the Web specification is bounded, no variables in the substructure of $\mathtt{r}_{k'-1}$, which is simulated by the left-hand side of rule $r_{k'-1}$, can be located at positions which are deeper than all the positions of the variables in the considered left-hand side. So, we must have $\text{height}(\sigma_{k'-1}) = \text{height}(\sigma_{k'-2})$. Therefore,

$$\begin{aligned} \text{height}(\mathtt{s}_{k'}) &\leq \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_{k'-1}) \\ &= \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_{k'-2}) \\ &\leq \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_0) \end{aligned}$$

which is a contradiction.
Case 2. Trivially, $\text{height}(\sigma_{k'-1}) \leq \text{height}(\sigma_{k'-2})$.
Hence,

$$\begin{aligned} \text{height}(\mathtt{s}_{k'}) &\leq \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_{k'-1}) \\ &\leq \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_{k'-2}) \\ &\leq \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_0) \end{aligned}$$

which also leads to a contradiction.

Thus, there cannot exist a term $\mathtt{s}_{k'}$ such that $\text{height}(\mathtt{s}_{k'}) > \text{height}(\mathtt{r}_{k'-1}) + \text{height}(\sigma_0)$ and the claim is proven.

Proposition 8 *Let* $\mathtt{W}$ *be a Web site,* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification. Then,*

$$\begin{aligned} &\{\mu(\mathtt{s}) \mid \varepsilon(\mathtt{p}) \rightharpoonup^m \mu(\mathtt{s}), 0 \leq m \leq k, \varepsilon(\mathtt{p}) \in \mathtt{W}\} \\ &\quad = \{\mu(\mathtt{s}) \mid \langle \mu(\mathtt{s}), \mathtt{q} \rangle \in \mathcal{J}_{\mathtt{M}} \uparrow^{\mathtt{W}} k\} \end{aligned}$$

Proof
($\supseteq$) Simple induction on the ordinal power k.
($\subseteq$) Simple induction on the length m of the rewrite sequence.

Corollary 1 *Let* $\mathtt{W}$ *be a Web site,* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a Web specification. Then,*

$$\{\mu(\mathtt{s}) \mid \varepsilon(\mathtt{p}) \rightharpoonup^* \mu(\mathtt{s}), \varepsilon(\mathtt{p}) \in \mathtt{W}\} = \{\mu(\mathtt{s}) \mid \langle \mu(\mathtt{s}), \mathtt{q} \rangle \in \mathit{lfp}(\mathcal{J}_\mathtt{M})\}$$

Now, we are ready to prove Proposition 3.

Proposition 3 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded Web specification and* $\mathtt{W}$ *be a Web site. Then, there exists* $k \in \mathbb{N}$ *such that* $\mathit{lfp}(\mathcal{J}_\mathtt{M}) = \mathcal{J}_\mathtt{M} \uparrow^\mathtt{W} k$.

Proof First, let us show that each $\mu(\mathtt{s})$, s.t. $\langle \mu(\mathtt{s}), \mathtt{q} \rangle \in \mathit{lfp}(\mathcal{J}_\mathtt{M})$, has a bounded height.

By Corollary 1, $\{\mu(\mathtt{s}) \mid \varepsilon(\mathtt{p}) \rightharpoonup^* \mu(\mathtt{s}), \varepsilon(\mathtt{p}) \in \mathtt{W}\} = \{\mu(\mathtt{s}) \mid \langle \mu(\mathtt{s}), \mathtt{q} \rangle \in \mathit{lfp}(\mathcal{J}_\mathtt{M})\}$.

Thus, for each $\mu(\mathtt{s})$, such that $\langle \mu(\mathtt{s}), \mathtt{q} \rangle \in \mathit{lfp}(\mathcal{J}_\mathtt{M})$, there exists a Web page $\varepsilon(\mathtt{p}) \in \mathtt{W}$ such that $\varepsilon(\mathtt{p}) \rightharpoonup^* \mu(\mathtt{s})$. Here, we distinguish two cases.

Case $(\varepsilon(\mathtt{p}) \rightharpoonup^0 \mu(\mathtt{s}))$. Let

$H_W = \max\{\mathrm{height}(\mathtt{p}) \mid \varepsilon(\mathtt{p}) \in \mathtt{W})\}$.

In this case, $\mu(\mathtt{s}) \equiv \varepsilon(\mathtt{p})$, hence

$\mathrm{height}(\mu(\mathtt{s})) = \mathrm{height}(\varepsilon(\mathtt{p})) \leq H_W$.

Case $(\varepsilon(\mathtt{p}) \rightharpoonup^+ \mu(\mathtt{s}))$. Let

$H_I = \max\{\mathrm{height}(\mathtt{r}) \mid \mathtt{r} \in \mathtt{Rhs_M}\}$

$H_S = \max\{\mathrm{height}(\sigma') \mid \varepsilon(\mathtt{p}) \rightharpoonup_{r'}^{\sigma'} \mu(\mathtt{s}), \varepsilon(\mathtt{p}) \in \mathtt{W}, r' \in \mathtt{I_M}\}$.

Now, we have $\varepsilon(\mathtt{p}) \rightharpoonup_{r_0}^{\sigma_0} \mu(\mathtt{s}_1) \rightharpoonup^* \mu(\mathtt{s})$. By Lemma 1, there exists $\mathtt{r} \in \mathtt{Rhs_M}$ such that

$\mathrm{height}(\mathtt{s}) \leq \mathrm{height}(\mathtt{r}) + \mathrm{height}(\sigma_0) \leq H_I + H_S$.

Thus, for each $\mu(\mathtt{s})$, such that $\langle \mu(\mathtt{s}), \mathtt{q} \rangle \in \mathit{lfp}(\mathcal{J}_\mathtt{M})$, we get

$\mathrm{height}(\mu(\mathtt{s})) \leq \max\{H_W, H_I + H_S\}$.

Since we have only a finite number of marked terms whose height is less than or equal to $\max\{H_W, H_I + H_S\}$, $\mathit{lfp}(\mathcal{J}_\mathtt{M})$ must be a finite set. Finally, since the operator $\mathcal{J}_\mathtt{M}$ is continuous (in particular, monotonic) and $\mathit{lfp}(\mathcal{J}_\mathtt{M})$ is a finite set, then there exists a natural number k such that $\mathit{lfp}(\mathcal{J}_\mathtt{M}) = \mathcal{J}_\mathtt{M} \uparrow^\mathtt{W} k$.

In order to prove Proposition 4, we first give some auxiliary results.

Lemma 2 *Let* $\mathtt{s}_1$ *and* $\mathtt{s}_2$ *be two Web page templates in* $\tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag}, \mathcal{V})$. *Then,* $\mathtt{s}_1 \cong \mathtt{s}_2$ *and* $\mathtt{s}_2 \cong \mathtt{s}_1$ *iff there exists a substitution* σ

$\mathtt{s}_2\sigma$ *partially matches* $\mathtt{s}_1$ *via some substitution* σ'.

Proof

($\Rightarrow$) Let us assume that

$\mathtt{s}_1 \cong \mathtt{s}_2$ or $\mathtt{s}_2 \cong \mathtt{s}_1$.

Case $\mathtt{s}_1 \cong \mathtt{s}_2$. By Definition 5, we also have $\mathtt{s}_1 \cong \mathtt{s}_2\sigma$, for every σ. Directly, by Definition 7, there exists a substitution σ' s.t. $\mathtt{s}_2\sigma$ partially matches $\mathtt{s}_1$ via σ'.

Case $\mathtt{s}_2 \cong \mathtt{s}_1$. $\mathtt{s}_1$ partially matches $\mathtt{s}_2$ via σ''. By Definition 5, we also have $\mathtt{s}_2\sigma'' \cong \mathtt{s}_1$. Indeed, $\mathtt{s}_2\sigma''$ is embedded into $\mathtt{s}_1$. Therefore, $\mathtt{s}_1 \cong \mathtt{s}_2\sigma''$, and $\mathtt{s}_2\sigma''$ partially matches $\mathtt{s}_1$ via the empty substitution ϵ. So, if we take $\sigma' = \epsilon$ and $\sigma = \sigma''$, then there exists σ and σ' such that $\mathtt{s}_2\sigma$ partially matches $\mathtt{s}_1$ via σ'.

($\Leftarrow$) Since there exists a substitution σ s.t. $\mathtt{s}_2\sigma$ partially matches $\mathtt{s}_1$ via some substitution σ', we have that $\mathtt{s}_1 \cong \mathtt{s}_2\sigma$. Here we can distinguish two cases: (1) there is a minimal, injective simulation of $\mathtt{s}_1$ in a subterm of $\mathtt{s}_2$ w.r.t. $\sim$ and hence $\mathtt{s}_1 \cong \mathtt{s}_2$; (2) there is a minimal, injective simulation of $\mathtt{s}_1$ in $\mathtt{s}_2\sigma$ w.r.t. $\sim$, which also involves some terms occurring in σ. In this case, we have that, for some $X/t \in \sigma$, t is simulated by some subterm of $\mathtt{s}_1$. It is not difficult to see that, by Definitions 4 and 5, there always exists a minimal, injective simulation between a variable and an arbitrary term. Exploiting this fact, we can state that each variable in $\mathtt{s}_2$, which is replaced by some term in $\mathtt{s}_2\sigma$, simulates the corresponding subterm in $\mathtt{s}_1$, thus $\mathtt{s}_2 \cong \mathtt{s}_1$.

Lemma 3 *Let* $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ *be a bounded* Web specification and* $r \equiv (\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{q} \rangle) \in \mathtt{I_M}$ *be a function-dependent rule w.r.t. R. Let* $\mu_1(\mathtt{s}_1)$ *and* $\mu_2(\mathtt{s}_2)$ *be two marked Web page templates. If* $\mu_1(\mathtt{s}_1) \rightharpoonup_r^\sigma \mu_2(\mathtt{s}_2)$, *then there do not exist a Web page template* $\mu_3(\mathtt{s}_3)$ *and substitution* σ' *s.t.* $\mu_2(\mathtt{s}_2) \rightharpoonup_{r'}^{\sigma'} \mu_3(\mathtt{s}_3)$, *with* $r' \in \mathtt{I_M}$.

Proof Let us consider $\mu_1(\mathtt{s}_1) \rightharpoonup_r^\sigma \mu_2(\mathtt{s}_2)$. Then, by Definition 8,

$\mathtt{s}_2 = \mathrm{Reduce}(\mathtt{r}\sigma, R)$.

We consider two cases.

Case $\mathtt{s}_2 \in \tau(\Sigma_R, \mathcal{V})$. $\mathtt{s}_2 = \mathit{Reduce}(\mathtt{r}\sigma, R)$ is just an irreducible form computed by the canonical TRS R. No partial rewriting steps can be applied to $\mathtt{s}_2$, because (1)

$\{\mathtt{f} \mid \mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_\mathtt{n}) \rightharpoonup \mu(\mathtt{r}) \in \mathtt{I_m}\} \cap \Sigma_R = \emptyset$,

since $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ is bounded*; (2) the function calls cannot be applied to arguments containing symbols in

$\{\mathtt{f} \mid \mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_\mathtt{n}) \rightharpoonup \mu(\mathtt{r}) \in \mathtt{I_m}\}$.

Case $\mathtt{s}_2 \in \tau(\mathcal{T}\mathrm{ext} \cup \mathcal{T}\mathrm{ag} \cup \Sigma_R, \mathcal{V}) \setminus \tau(\Sigma_R, \mathcal{V})$.
$\mathtt{s}_2 = \mathrm{Reduce}(\mathtt{r}\sigma, R) = \mathtt{r}^*\sigma^*$, where $\mathtt{r}^*$ is the right-hand side of the function-independent version of r w.r.t. R, and σ^* is a suitable substitution. Since $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ is bounded*, condition 2 of Definition 12 holds. Thus, by applying Lemma 2, we can ensure that the subterms that occur at positions coming from $\mathtt{r}^*$ cannot be partially rewritten in a subsequent step. Moreover, for each $X/t \in \sigma^*$, t does not contain any subterm which can be partially rewritten, because (1)

$$\{\mathtt{f} \mid \mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n) \rightharpoonup \mu(\mathtt{r}) \in \mathtt{I_m}\} \cap \Sigma_R = \emptyset,$$

since $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ is bounded*; (2) the function calls are never applied to arguments containing symbols in $\{\mathtt{f} \mid \mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n) \rightharpoonup \mu(\mathtt{r}) \in \mathtt{I_m}\}$. Therefore, there are no subterms in $\mathtt{s}_2 = \mathtt{r}^*\sigma^*$ which can be partially rewritten any longer.

Lemma 4 *Let $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ be a bounded* Web specification and*

$$\mu_0(\mathtt{s}_0) \rightharpoonup_{r_0}^{\sigma_0} \mu_1(\mathtt{s}_1) \rightharpoonup_{r_1}^{\sigma_1} \mu_2(\mathtt{s}_2) \rightharpoonup_{r_2}^{\sigma_2} \ldots$$

be a (possibly infinite) partial rewrite sequence, where $r_i \in \mathtt{I_M}$, $i = 0, 1, 2, \ldots$ Then, there exists $k \in \mathbb{N}$ s.t. for each $\mathtt{s}_i$, $i = 0, 1, 2, \ldots$,

$$\mathrm{height}(\mathtt{s}_i) \leq k.$$

Proof Let us consider the partial rewrite sequence

$$\mu_0(\mathtt{s}_0) \rightharpoonup_{r_0}^{\sigma_0} \mu_1(\mathtt{s}_1) \rightharpoonup_{r_1}^{\sigma_1} \mu_2(\mathtt{s}_2) \rightharpoonup_{r_2}^{\sigma_2} \ldots$$

We distinguish two cases.

Case 1. First consider that the partial rewrite sequence only contains function-independent rules. Since, $(\mathtt{I_N} \uplus \{r \in \mathtt{I_M} \mid \mathtt{r} \text{ is function-independent w.r.t.} R\}, R)$ is bounded, we can simply apply Lemma 1. So,

$$\mathrm{height}(\mathtt{s}_i) \leq \mathrm{height}(\mathtt{r}_{i-1}) + \mathrm{height}(\sigma_0)$$

where $\mathtt{r}_{i-1}$ is the right-hand side of rule r_{i-1}, $i = 1, 2, \ldots$ Let $k' = \max\{\mathrm{height}(\mathtt{r}) \mid \mathtt{r} \in \mathrm{Rhs}_{\mathtt{M}}\}$. By taking $k = \max\{\mathrm{height}(\mathtt{s}_0), k' + \mathrm{height}(\sigma_0)\}$, we prove the claim.
Case 2. Now, assume that function-dependent completeness rules are also used in the sequence. Let us suppose that $\mu_j(\mathtt{s}_j) \rightharpoonup_{r_j}^{\sigma_j} \mu_{j+1}(\mathtt{s}_{j+1})$ is the first partial rewrite step of the sequence in which a function-dependent rule r_i occurs. By Lemma 3, $\mu_{j+1}(\mathtt{s}_{j+1})$ cannot be partially rewritten any longer, therefore the sequence must be finite. So, by inspecting the heights of every term appearing in the sequence, we can find the one with the maximum height h. Finally, it is enough to choose $k = h$ and the claim in proven.

Proposition 4 *Let $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ be a bounded* Web specification and $\mathtt{W}$ be a Web site. Then, there exists $k \in \mathbb{N}$ such that $lfp(\mathcal{J}_{\mathtt{M}}) = \mathcal{J}_{\mathtt{M}} \uparrow^{\mathtt{W}} k$.*

Proof Similarly to the proof of Proposition 3, by Corollary 1, $\{\mu(\mathtt{s}) \mid \varepsilon(\mathtt{p}) \rightharpoonup^* \mu(\mathtt{s}), \varepsilon(\mathtt{p}) \in \mathtt{W}\} = \{\mu(\mathtt{s}) \mid \langle \mu(\mathtt{s}), \mathtt{q} \rangle \in lfp(\mathcal{J}_{\mathtt{M}})\}$.

Thus, for each $\mu(\mathtt{s})$, such that $\langle \mu(\mathtt{s}), \mathtt{q} \rangle \in lfp(\mathcal{J}_{\mathtt{M}})$, there exists a partial rewrite sequence $\varepsilon(\mathtt{p}) \rightharpoonup^* \mu(\mathtt{s})$, where $\varepsilon(\mathtt{p}) \in \mathtt{W}$. As $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ is bounded*, every marked term appearing in the partial rewrite sequence $\varepsilon(\mathtt{p}) \rightharpoonup^* \mu(\mathtt{s})$ has a bounded height by Lemma 4 and hence $lfp(\mathcal{J}_{\mathtt{M}})$ is a finite set.

Finally, since the operator $\mathcal{J}_{\mathtt{M}}$ is continuous (in particular, monotonic) and the set $lfp(\mathcal{J}_{\mathtt{M}})$ is finite, then there exists a natural number k such that $lfp(\mathcal{J}_{\mathtt{M}}) = \mathcal{J}_{\mathtt{M}} \uparrow^{\mathtt{W}} k$.

Proposition 5 *Let $\mathtt{W}$ be a Web site, $(\mathtt{I_N} \uplus \mathtt{I_M}, R)$ be a bounded* Web specification, and $\mathtt{Req}_{\mathtt{M,W}}$ be the set of requirements for $\mathtt{W}$ w.r.t. $\mathtt{I_M}$. Then, the procedure* Completeness–errors*$(\mathtt{W}, \mathtt{I_M}, R)$ terminates. Moreover, for each error message regarding term $\mathtt{e}$, which is returned by the procedure, there exists $\langle \mu(\mathtt{e}), \mathtt{q} \rangle \in \mathtt{Req}_{\mathtt{M,W}}$, with $\mathtt{q} \in \{\mathtt{A}, \mathtt{E}\}$, which is not satisfied in $\mathtt{W}$.*

Proof First of all, let us prove the termination of procedure Completeness–errors$(\mathtt{W}, \mathtt{I_M}, R)$. Line 2 computes the sets of requirements $\mathtt{Req}_{\mathtt{M,W}}$. Since $\mathtt{W}$ is a bounded* Web specification, by Proposition 4, the $lfp(\mathcal{J}_{\mathtt{M}})$ is a finite set which is computed in finite time. Therefore, $\mathtt{Req}_{\mathtt{M,W}} \leftarrow lfp(\mathcal{J}_{\mathtt{M}}) \setminus \mathtt{W}$ is also computed in finite time.

The loop within lines 3–18 is executed $|\mathtt{Req}_{\mathtt{M,W}}|$ times. It simply takes every single requirement belonging to $\mathtt{Req}_{\mathtt{M,W}}$ and analyzes it in order to discover whether it is satisfied.

The assignment in line 4 computes the test set $\mathtt{TEST}_{\langle \mu(\mathtt{e}), \mathtt{q} \rangle}$ which terminates by Definition 13: actually, it is just an application of the simulation algorithm [24] to a finite set of (marked) terms, i.e., the Web site $\mathtt{W}$.

The emptiness test in lines 5–7 trivially terminates as well as the *case* statement in lines 8–17, which includes some checks based on the simulation algorithm [24].

Now, let us prove the partial correctness of the algorithm: that is, if an error message regarding $\mathtt{e}$ (incompleteness symptom) is returned by the procedure, then $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$, where $\mathtt{q} \in \{\mathtt{A}, \mathtt{E}\}$, is not satisfied in $\mathtt{W}$.

If an error message regarding $\mathtt{e}$ (incompleteness symptom) is returned by the procedure, then it refers to a requirement $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$. Let us consider the iteration of the loop in lines 3–18 which checks $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$. Note that an error message can be risen in three cases. Whenever

1. $\mathtt{TEST}_{\langle \mu(\mathtt{e}), \mathtt{q} \rangle} = \emptyset$.
2. There exists $\mathtt{p} \equiv (V, E, r, \mathrm{label}) \in \mathtt{TEST}_{\langle \mu(\mathtt{e}), \mathtt{A} \rangle}$ s.t. $\mathtt{e} \ncong \mathtt{p}_{|v}$, with $v \in V$.
3. For each $\mathtt{p} \equiv (V, E, r, \mathrm{label}) \in \mathtt{TEST}_{\langle \mu(\mathtt{e}), \mathtt{E} \rangle}$, $\mathtt{e} \ncong \mathtt{p}_{|v}$, with $v \in V$.

In case 1, requirement $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$ is not satisfied in $\mathtt{W}$ directly by Definition 14 (see point 1) [resp., Definition 15 (see point 1)] when $\mathtt{q} = \mathtt{A}$ (resp., $\mathtt{q} = \mathtt{E}$).

In case 2, requirement $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$, where $\mathtt{q} = \mathtt{A}$, is not satisfied in $\mathtt{W}$ by Definition 14 (see point 2).

In case 3, requirement $\langle \mu(\mathtt{e}), \mathtt{q} \rangle$, where $\mathtt{q} = \mathtt{E}$, is not satisfied in $\mathtt{W}$ by Definition 15 (see point 2).

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web. From Relations to Semistructured Data and XML. Morgan Kaufmann (2000)
2. Alpuente, M., Ballis, D., Falaschi, M.: A rewriting-based framework for Web sites verification. In: Proceedings of 1st International Workshop on Ruled-Based Programming (RULE'04), vol. 124(1). ENTCS, Elsevier (2004)
3. Alpuente, M., Ballis, D., Falaschi, M.: VERDI: an automated tool for Web sites Verification. In: Alferes, J.J., Leite. J. (eds) Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04), vol. 3229 of Lecture Notes in Computer Science, pp. 726–729. Springer, Berlin Heidelberg New York (2004)
4. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, Cambridge (1998)
5. Ballis, D.: Rule-based Software Verification and Correction. PhD thesis, University of Udine and Technical University of Valencia (2005)
6. Ballis, D., García Vivó, J.: A rule-based system for Web site verification. In: Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites (WWV'05). ENTCS, Elsevier, 2005. To appear.
7. Baxter, I.D., Ricca, F., Tonella, P.: Web application transformations based on rewrite rules. Inform. Softw. Technol. **44** (13), (2002)
8. Bertino, E., Mesiti, M., Guerrin, G.: A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications. Inform. Syst. **29**(1), 23–46 (2004)
9. Bezem, M.: TeReSe, Term Rewriting Systems, chapter Mathematical background (Appendix A). Cambridge University Press,Cambridge (2003)
10. Bry, F., Schaffert, S.: Towards a declarative query and transformation language for XML and semistructured data: simulation unification. In: Proceedings of the International Conference on Logic Programming (ICLP'02) vol. 2401 of LNCS. Springer Berlin Heidelberg New York (2002)
11. Bry, F., Schaffert, S.: The XML query language xcerpt: design principles, examples, and semantics. Technical report, Available at: http://www.xcerpt.org(2002)
12. Buneman, P., Davidson, S.B., Hillebrand, G.G., Suciu, D.: A query language and optimization techniques for unstructured data. In: Proceedings of ACM SIGMOD International Conference on Management of Data (ICMD'96) (1996)
13. Capra, L., Emmerich, W., Finkelstein, A., Nentwich C.: XLINKIT: a consistency checking and smart link generation service. ACM Trans. Internet Technol. **2** (2), 151–185 (2002)
14. Cortesi, A., Dovier, A., Quintarelli, E., Tanca, L.: Operational and abstract semantics of a graphical query language. Theore. Comput. Sci. **275**, 521–560 (2002)
15. Dershowitz, N., Plaisted, D.: Rewriting. Handbook Automated Reasoning **1**, 535–610 (2001)
16. Despeyroux, T., Trousse, B.: Semantic verification of Web sites using natural semantics. In: Proceedings of 6th Conference on Content-Based Multimedia Information Access (RIAO'00) (2000)
17. Di Sciascio, E., Donini, F.M., Mongiello, M., Piscitelli, G.: Web applications design and maintenance using symbolic model checking. In: Proceedings 7th European Conference on Software Maintenance and Reengineering, pp. 63. IEEE Computer Society (2003)
18. Easterbrook, S.M., Nuseibeh, B., Russo, A.: Leveraging inconsistency in software development. IEEE Comp. **33** (4), 24–29 (2000)
19. Ellmer, E., Emmerich, W., Finkelstein, A., Nentwich, C.: Flexible consistency checking. ACM Trans. Softw. Eng. **12**(1), 28–63 (2003)
20. Fan, W., Libkin, L.: On XML integrity constraints in the presence of DTDs. J. ACM **49**(3), 368–406 (2002)
21. Fernandez, M., Florescu, D., Levy, A., Suciu, D.: Verifying integrity constraints on Web sites. In: Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99) vol. 2 pp. 614–619. Morgan Kaufmann (1999)
22. Fernandez, M.F., Suciu, D.: Optimizing regular path expressions using graph schemas. In: Proceedings of International Conference on Data Engineering (ICDE'98), pp. 14–23 (1998)
23. M. Hanus (ed.). Curry: an integrated functional logic language. Available at: http://www-i2.informatik.rwthaachen.de/~ hanus/curry (1999)
24. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: IEEE Sympo Foundations Comput. Sci. 453–462 (1995)
25. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
26. Hosoya, H., Pierce, B.: Regular expressions pattern matching for XML. In: Proceedings of 25th ACM SIGPLAN-SIGACT International Symposium POPL, pp. 67–80. ACM (2001)
27. Hussmann, H.: Unification in conditional-equational Theories. In: Proceedings of ALP'88, pp. 31–40. Springer LNCS 343 (1988)
28. Imagiware. Inc. Doctor HTML: quality assessment for the Web. Available at: http://www.doctor-html.com/RxHTML/.
29. Kirchner, C., Qian, Z., Singh, P.K,. Stuber, J.: Xemantics: a Rewriting Calculus-Based Semantics of XSLT. Rapport de recherche A01-R-386, LORIA (2001)
30. Klop, J.W.: Term rewriting systems. In: Abramsky, S., Gabbay, D., Maibaum, T., (eds.) Handbook of Logic in Computer Science vol. I, pp. 1–112. Oxford University Press, Oxford (1992)
31. Moreno-Navarro, J.J., Rodriguez-Artalejo.: BABEL: a functional and logic programming language based on a constructor discipline and narrowing. In: Grabowski, I., Lescanne, P., Wechler, W., (eds.) Proceedings of the International Conference on Algebraic and Logic Programming, pp. 223–232 Springer LNCS 343 (1988)
32. Nentwich, C., Emmerich, W., Finkelstein, A.: Consistency management with repair actions. In: Proceedings of the 25th International Conference on Software Engineering (ICSE'03). IEEE Computer Society (2003)
33. Nesbit, S.: HTML Tidy: keeping it clean Available at: http://www.webreview.com/2000/06-16/webauthors/06-16-00-3.shtml(2000)
34. The Open Group. Unix Regular Expressions. Available at: http://www.opengroup.org/onlinepubs/7908799/xbd/re.html.

35. Academisa Sinica Computing Centre. The schematron: an XML structure validation language using pattern in trees. Available at: http://xml.ascc.net/resource/schematron/schematron.html
36. World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0, second edition Available at: http://www.w3.org (1999)
37. World Wide Web Consortium (W3C). XML path language (XPath) Available at: http://www.w3.org (1999)
38. World Wide Web Consortium (W3C). Extensible HyperText Markup Language (XHTML) Available at: http://www.w3.org (2000)