A Semi-Automatic Methodology for Repairing Faulty Web Sites *

María Alpuente
DSIC, Universidad Politécnica de Valencia,
Camino de Vera s/n, Apdo. 22012,
46071 Valencia, Spain.
Email: alpuente@dsic.upv.es

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

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

Daniel Romero
DSIC, Universidad Politécnica de Valencia,
Camino de Vera s/n, Apdo. 22012,
46071 Valencia, Spain.
Email: dromero@dsic.upv.es

Abstract

The development and maintenance of Web sites are difficult tasks. To maintain the consistency of ever-larger, complex Web sites, Web administrators need effective mechanisms that assist them in fixing every possible inconsistency. In this paper, we present a novel methodology for semi-automatically repairing faulty Web sites which can be integrated on top of an existing rewriting-based verification technique developed in a previous work. Starting from a categorization of the kinds of errors that can be found during the Web verification activities, we formulate a stepwise transformation procedure that achieves correctness and completeness of the Web site w.r.t. its formal specification while respecting the structure of the document (e.g. the schema of an XML document). Finally, we shortly describe a prototype implementation of the repairing tool which we used for an experimental evaluation of our method.

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 and repairing.

A lot of research work has been invested in consistency management and repairing of software applications and databases, whereas similar technologies are much less mature for Web systems. In [13], a repair framework for inconsistent distributed documents is presented that complements the tool xlinkit [7]. The main contribution is the semantics that maps xlinkit's first order logic language to a catalogue of repairing actions that can be used to interactively correct rule violations though it does not predict whether a repair action can provoke new errors to appear. Also, it is not possible to detect whether two formulae expressing a requirement for the Web site are incompatible. Similarly, in [15, 17] an extension for the tool CDET [16] is presented. This extension includes a mechanism to remove inconsistencies from sets of interrelated documents, which first generates direct acyclic graphs (DAGs) representing the relations between documents and then repairs are directly derived from such DAGs. In this case, temporal rules are supported and interference and compatibility of repairs are not completely neglected. Unfortunately, this compatibility is expensive to check for temporal rules. Both approaches rely on basic techniques borrowed from the field of active databases [5]. Current research in this field focuses on the derivation of active rules that automatically fire repair actions leading to a consistent state after each update [12].

In our previous work on GVERDI [1, 2], we presented a rewriting-like approach to Web site specification and verification. Our methodology allows us to specify the integrity conditions for the Web sites and then diagnose errors by computing the requirements not fulfilled by a given Web site, that is, by finding out incorrect/forbidden patterns and missing/incomplete Web pages. We believe that our approach is particularly suitable for checking large static Web sites, e.g. digital libraries, which contain a number of deeply interconnected XML documents.

*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.

In this paper, we aim to complement our methodology with a tool-independent technique for semi-automatically repairing the errors found during that verification phase. First, we formalize the kinds of errors that can be found in a Web site w.r.t. a Web site specification. Then, we classify the *repair actions* that can be performed to repair each kind of error. Since different repair actions can be executed to repair a given error, our method is tuned to deliver a set of correct and complete repair actions to choose between. Our repair methodology is formulated in two phases. First, all the necessary actions to make the Web site *correct* are performed. Once correctness of the Web site has been achieved, the user is given the option to execute all the necessary actions to make it *complete*. Moreover, this methodology allows us to manage the problems that might arise from the interaction of repair actions.

The rest of the paper is structured as follows. Section 2 summarizes some preliminary definitions and notations about term rewriting systems. In Section 3, first we recall the Web verification framework of [1], which is based on tree simulation, and then we categorize the different kinds of errors that can be found as an outcome of the verification technique. Section 4 describes our repairing methodology for faulty Web sites, while Section 5 describes the system we implemented. Section 6 concludes and discusses future work.

2. Preliminaries

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 [8] (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. Terms are viewed as labelled trees in the usual way. Positions are represented by sequences of natural numbers denoting an access path in a term. The empty sequence Λ denotes the root position. By notation $w_1.w_2$, we denote the concatenation of position w_1 and position w_2. Positions are ordered by the prefix ordering, that is, given the positions w_1, w_2, $w_1 \leq w_2$ if there exists a position x such that $w_1.x = w_2$. Given $S \subseteq \Sigma \cup \mathcal{V}$, $O_S(t)$ denotes the set of positions of a term t which are rooted by symbols in S. Moreover, for any position x, $\{x\}.O_S(t) = \{x.w \mid w \in O_S(t)\}$. $t_{|u}$ is the subterm at the position u of t. $t[r]_u$ is the term t with the subterm rooted at the position u replaced by r. Given a term t, we say that t is *ground*, if no variables occur in t. Syntactic equality between objects is represented by $\equiv$.

A *substitution* $\sigma \equiv \{X_1/t_1, X_2/t_2, \ldots\}$ is a mapping from the set of variables $\mathcal{V}$ into the set of terms $\tau(\Sigma, \mathcal{V})$ satisfying the following conditions: (i) $X_i \neq X_j$, whenever $i \neq j$, (ii) $X_i\sigma = t_i$, $i = 1, \ldots, n$, and (iii) $X\sigma = X$, for any $X \in \mathcal{V} \setminus \{X_1, \ldots, X_n\}$. By $Var(s)$ we denote the set of variables occurring in the syntactic object s.

Term rewriting systems provide an adequate computational model for functional languages. In the sequel, we follow the standard framework of term rewriting (see [4, 10]). 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 $Var(\rho) \subseteq Var(\lambda)$. We will often write just R instead of (Σ, R). Sometimes, we denote the signature of a TRS (Σ, R) by Σ_R.

A rewrite step is the application of a rewrite rule to an expression. A term s *rewrites* to a term t via $r \in R$, $s \to_r t$ (or $s \to_R t$), if there exist a position $u \in O_\Sigma(s)$, $r \equiv \lambda \to \rho$, and a substitution σ such that $s_{|u} \equiv \lambda\sigma$ and $t \equiv s[\rho\sigma]_u$. When no confusion can arise, we will omit any subscript (i.e. $s \to t$). 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 $t_1 \to_R t_2 \to_R \ldots$ A TRS R is *confluent* if, for all terms s, t_1, t_2, such that $s \to^*_R t_1$ and $s \to^*_R t_2$, there exists a 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 a unique *irreducible form*.

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

3. Rewriting-based Web Verification

In this section, we briefly recall the formal verification methodology proposed in [1], which is able to detect erroneous as well as missing information in a Web site.

In our framework, a *Web page* is either an XML [18] or an XHTML [19] document, which we assume to be well-formed, since there are plenty of programs and online services which are able to validate XHTML/XML syntax and perform link checking (e.g. [14], [9]). As Web pages are provided with a tree-like structure, they can be straightforwardly encoded into ordinary terms of a suitable term algebra $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$, where $\mathcal{T}ext \cup \mathcal{T}ag$ is a signature containing the text and the tags on which we build our Web pages. Note that XML/XHTML tag attributes can be considered as common tagged elements, and hence translated in the same way. Therefore, *Web sites* can be represented as finite sets of (ground) terms.

In Figure 1, we present a Web site $\mathtt{W}$ of a research group, which contains information about group members affiliation, scientific publications, research projects, teaching and

personal data.

In the following, we will also consider terms of the non-ground term algebra $\tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$, which may contain variables. An element $\mathtt{s} \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ is called *Web page template*.

3.1. Web specification language

A Web specification is a triple (I_N, I_M, R), where R, I_N, and I_M are finite set of rules. The set R contains the definition of some auxiliary functions which are executed by standard rewriting [10].

The set I_N describes constraints for detecting erroneous Web pages (*correctNess rules*). As the amount of faulty information is typically a small portion of the whole content of a Web site, our correctness rules model erroneous patterns rather than correct/safe patterns. We believe that this approach facilitates both the specification and the verification of correctness properties. Formally, a correctness rule has the following form: $\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{C}$, with $Var(\mathtt{C}) \subseteq Var(\mathtt{l})$, where $\mathtt{l}$ is a term, $\mathtt{error}$ is a reserved constant, and $\mathtt{C}$ is a (possibly empty) finite sequence containing membership tests (e.g. $\mathtt{X} \in \mathtt{rexp}$) w.r.t. a given regular language[1], and/or equations over terms. For the sake of expressiveness, we also allow to write inequalities of the form $\mathtt{s} \neq \mathtt{t}$ in $\mathtt{C}$, which hold whenever the corresponding equation $\mathtt{s} = \mathtt{t}$ does not hold. When $\mathtt{C}$ is empty, we simply write $\mathtt{l} \rightharpoonup \mathtt{error}$. The meaning of a correctness rule $\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{C}$, where $\mathtt{C} \equiv (\mathtt{X_1\ in\ rexp_1}, \ldots, \mathtt{X_n\ in\ rexp_n}, \mathtt{s_1} = \mathtt{t_1} \ldots \mathtt{s_m} = \mathtt{t_m})$, is the following. We say that $\mathtt{C}$ *holds* for substitution σ, if (i) each structured text $\mathtt{X_i}\sigma$, $\mathtt{i} = 1, \ldots, \mathtt{n}$, is contained in the language of the corresponding regular expression $\mathtt{rexp_i}$; (ii) each instantiated equation (resp. inequality) $(\mathtt{s_i} = \mathtt{t_i})\sigma$ (resp. $(\mathtt{s_i} \neq \mathtt{t_i})\sigma$) , $\mathtt{i} = 1, \ldots, \mathtt{m}$, holds in R.

The Web page $\mathtt{p}$ is considered incorrect if an instance $\mathtt{l}\sigma$ of $\mathtt{l}$ is *recognized* within $\mathtt{p}$, and $\mathtt{C}$ holds for σ.

The third set of rules I_M specifies some properties for detecting incomplete/missing Web pages (*coMpleteness rules*). A completeness rule is defined as $\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{q} \rangle$, where $\mathtt{l}$ and $\mathtt{r}$ are terms and $\mathtt{q} \in \{\mathtt{E}, \mathtt{A}\}$. 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 can contain functions, which are defined in R. Intuitively, the interpretation of a universal rule $\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{A} \rangle$ (respectively, an existential rule $\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{E} \rangle$) w.r.t. a Web site $\mathtt{W}$ is as follows: if (an instance of) $\mathtt{l}$ is recognized in $\mathtt{W}$, also (an instance of) the irreducible form of $\mathtt{r}$ must be recognized in *all* (respectively, *some*) of the Web pages which embed (an instance of) $\mathtt{r}$.

Sometimes, we may be interested in checking a given completeness property only on a subset of the whole Web site. For this purpose, some symbols in the right-hand sides of the rules are marked by means of the constant symbol $\sharp$. Marking information of a given rule r is used to select the subset of the Web site in which we want to check the condition formalized by r. More specifically, rule r is executed on all and only the Web pages embedding the marking information. A detailed example follows.

Example 3.1. *Consider the Web specification which consists of the following completeness and correctness rules along with a term rewriting system defining the string concatenation function $++$, the arithmetic operators $+$ and $*$ on natural numbers and the relational operator $\leq$.*

$$\mathtt{member(name(X),surname(Y))} \rightharpoonup \sharp\mathtt{hpage(fullname(X} ++ \mathtt{Y),\ status)}\ \langle \mathtt{E} \rangle$$
$$\mathtt{hpage(status(professor))} \rightharpoonup \sharp\mathtt{hpage(}\sharp\mathtt{status(}\sharp\mathtt{professor),\ teaching)}\ \langle \mathtt{A} \rangle$$
$$\mathtt{pubs(pub(name(X), surname(Y)))} \rightharpoonup \sharp\mathtt{members(member(name(X),\ surname(Y)))}\ \langle \mathtt{E} \rangle$$
$$\mathtt{courselink(url(X), urlname(Y))} \rightharpoonup \sharp\mathtt{cpage(title(Y))}\ \langle \mathtt{E} \rangle$$
$$\mathtt{hpage(X)} \rightharpoonup \mathtt{error} \mid \mathtt{X\ in\ [:TextTag:]^*sex[:TextTag:]^*}$$
$$\mathtt{blink(X)} \rightharpoonup \mathtt{error}$$
$$\mathtt{project(grant1(X), grant2(Y), total(Z))} \rightharpoonup \mathtt{error} \mid \mathtt{X} + \mathtt{Y} \neq \mathtt{Z}$$
$$\mathtt{project(grant1(X), grant2(Y))} \rightharpoonup \mathtt{error} \mid \mathtt{X} \neq \mathtt{Y} * 2$$
$$\mathtt{total(Z)} \rightharpoonup \mathtt{error} \mid \mathtt{Z} \geq 500000 = \mathtt{true}$$

This Web specification models some required properties for the Web site of Figure 1. 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 concatenating the name and the surname strings by means of the $++$ function. 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 marks which identify professor home pages. 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 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*

[1] Regular languages are represented by means of the usual Unix-like regular expressions syntax.

```
{(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(ugo),surname(blu),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),phone(4444),
           teaching(course(algebra))),
 (5) pubs(pub(name(ugo),surname(blu),title(blah1),blink(year(2003))),
          pub(name(anna),surname(gialli),title(blah2),year(2002))),
 (6) projects(project(pname(A1),grant1(1000),grant2(200),
                      total(1100),coordinator(fullname(mariorossi))),
              project(pname(B1),grant1(2000),grant2(1000),
                      projectleader(surname(gialli),name(anna)),
                      total(3000)))}
```

Figure 1. An example of a Web site for a research group

regular expression [: TextTag :]*sex[: TextTag :]**, which identifies the regular language of all the strings built over* $(\mathcal{T}ext \cup \mathcal{T}ag)$ *containing word* sex*. 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 last three rules respectively state that, for each research project, the total project budget must be equal to the sum of the grants, the first grant should be the double of the second one, and the total budget is less than 500000 euros.*

In our methodology, diagnoses are carried out by running Web specifications on Web sites. This is mechanized by means of *partial rewriting*, a novel rewriting technique which we obtain by replacing the traditional pattern-matching of term rewriting with a new mechanism based on *page* (tree) *simulation* (cf. [1]).

3.2. Simulation and partial rewriting

Partial rewriting extracts "some pieces of information" from a page, pieces them together, and then rewrites the glued term. The assembling is done by means of tree simulation, which recognizes the structure and the labeling of a given term (Web page template) inside a particular page of the Web site.

Our notion of simulation, $\trianglelefteq$, is an adaptation of Kruskal's *embedding* (or "syntactically simpler") relation [6] where we ignore the usual *diving* rule[2] [11].

Definition 3.1 (simulation)**.** *The* simulation *relation*

$$\trianglelefteq \subseteq \tau(\mathcal{T}ext \cup \mathcal{T}ag) \times \tau(\mathcal{T}ext \cup \mathcal{T}ag)$$

[2]The diving rule allows one to "strike out" a part of the term at the right-hand side of the relation $\trianglelefteq$. Formally, $s \trianglelefteq f(t_1, \ldots t_n)$, if $s \trianglelefteq t_i$, for some i.

on Web pages is the least relation satisfying the rule:

$\mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_\mathtt{m}) \trianglelefteq \mathtt{g}(\mathtt{s}_1, \ldots, \mathtt{s}_\mathtt{n})$ *iff* $\mathtt{f} \equiv \mathtt{g}$ *and* $\mathtt{t}_\mathtt{i} \trianglelefteq \mathtt{s}_{\pi(\mathtt{i})}$*, for* $\mathtt{i} = 1, \ldots, \mathtt{m}$*, and some injective function* $\pi : \{1, \ldots, \mathtt{m}\} \to \{1, \ldots, \mathtt{n}\}$.

Given two Web pages $\mathtt{s}_1$ and $\mathtt{s}_2$, if $\mathtt{s}_1 \trianglelefteq \mathtt{s}_2$ we say that $\mathtt{s}_1$ *simulates* (or *is embedded* or *recognized* into) $\mathtt{s}_2$. We also say that $\mathtt{s}_2$ *embeds* $\mathtt{s}_1$. Note that, in Definition 3.1, for the case when $\mathtt{m}$ is 0 we have $\mathtt{c} \trianglelefteq \mathtt{c}$ for each constant symbol $\mathtt{c}$. Also note that $\mathtt{s}_1 \not\trianglelefteq \mathtt{s}_2$ if either $\mathtt{s}_1$ or $\mathtt{s}_2$ contain variables.

Now we are ready to introduce the *partial rewrite* relation between Web page templates. W.l.o.g., we disregard conditions and/or quantifiers from the Web specification rules. Roughly speaking, given a Web specification rule $\mathtt{l} \rightharpoonup \mathtt{r}$, partial rewriting allows us to extract from, a given Web page $\mathtt{s}$, a subpart of $\mathtt{s}$ which is simulated by a ground instance of $\mathtt{l}$, and to replace $\mathtt{s}$ by a reduced, ground instance of $\mathtt{r}$. Let $\mathtt{s}, \mathtt{t} \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$. Then, $\mathtt{s}$ *partially rewrites* to $\mathtt{t}$ via rule $\mathtt{l} \rightharpoonup \mathtt{r}$ and substitution σ iff there exists a position $u \in O_{\mathcal{T}ag}(\mathtt{s})$ such that (i) $\mathtt{l}\sigma \trianglelefteq \mathtt{s}_{|u}$, and (ii) $\mathtt{t} = \mathit{Reduce}(\mathtt{r}\sigma, R)$, where function $\mathit{Reduce}(x, R)$ computes, by standard term rewriting, the irreducible form of x in R. Note that the context of the selected reducible expression $\mathtt{s}_{|u}$ is disregarded after each rewrite step. By notation $\mathtt{s} \rightharpoonup_I \mathtt{t}$, we denote that $\mathtt{s}$ is partially rewritten to $\mathtt{t}$ using some rule belonging to the set I.

3.3. Error diagnoses

In order to diagnose correctness as well as completeness errors, we follow the method we presented in [1]. We classify the kind of errors which can be found in a Web site

in terms of the different outputs delivered by our verification technique when is fed with a Web site specification. In Section 4, we will exploit this information to develop our repairing/correction methodology. Let us start by characterizing correctness errors.

Definition 3.2 (correctness error)**.** *Let* $\mathtt{W}$ *be a Web site and* (I_M, I_N, R) *be a Web specification. Then, the quadruple* $(\mathtt{p}, w, \mathtt{l}, \sigma)$ *is a* correctness error evidence *iff* $\mathtt{p} \in \mathtt{W}$*,* $w \in O_{\mathcal{T}ag}(\mathtt{p})$*, and* $\mathtt{l}\sigma$ *is an instance of the left-hand side* $\mathtt{l}$ *of a correctness rule belonging to* I_N *such that* $\mathtt{l}\sigma \trianglelefteq \mathtt{p}_{|w}$*.*

Given a correctness error evidence $(\mathtt{p}, w, \mathtt{l}, \sigma)$, $\mathtt{l}\sigma$ represents the erroneous information which is embedded in a subterm of the Web page $\mathtt{p}$, namely $\mathtt{p}_{|w}$.

We denote the set of all correctness error evidences of a Web site $\mathtt{W}$ w.r.t. a set of correctness rules I_N by $E_N(\mathtt{W})$. When no confusion can arise, we just write E_N.

As for completeness errors, we can distinguish three classes of errors: (i) *Missing Web pages*, (ii) *Universal completeness errors*, (iii) *Existential completeness errors*. In our framework, all the three kinds of completeness errors can be detected by partially rewriting Web pages to some expression $\mathtt{r}$ by means of the rules of I_M, and then checking whether $\mathtt{r}$ does not occur in a suitable subset of the Web site.

Definition 3.3 (Missing Web page)**.** *Let* $\mathtt{W}$ *be a Web site and* (I_M, I_N, R) *be a Web specification. Then the pair* $(\mathtt{r}, \mathtt{W})$ *is a* missing *Web page error evidence if there exists* $\mathtt{p} \in W$ *s.t.* $\mathtt{p} \rightharpoonup^+_{I_M} \mathtt{r}$ *and* $\mathtt{r} \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ *does not belong to* $\mathtt{W}$*.*

When a missing Web page error is detected, the evidence $(\mathtt{r}, \mathtt{W})$ signals that the expression $\mathtt{r}$ does not appear in the whole Web site $\mathtt{W}$. In order to formalize existential as well as universal completeness errors, we introduce the following auxiliary definition.

Definition 3.4. *Let* $\mathtt{P}$ *be a set of terms in* $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$ *and* $\mathtt{r} \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$*. We say that* $\mathtt{P}$ *is* universally *(resp.* existentially*)* complete *w.r.t.* $\mathtt{r}$ *iff for each* $\mathtt{p} \in \mathtt{P}$ *(resp. for some* $\mathtt{p} \in \mathtt{P}$*), there exists* $w \in O_{\mathcal{T}ag}(\mathtt{p})$ *s.t.* $\mathtt{r} \trianglelefteq \mathtt{p}_{|w}$*.*

Definition 3.5 (Universal completeness error)**.** *Let* $\mathtt{W}$ *be a Web site and* (I_M, I_N, R) *be a Web specification. Then the triple* $(\mathtt{r}, \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}, \mathtt{A})$ *is a* universal completeness error evidence*, if there exists* $\mathtt{p} \in \mathtt{W}$ *s.t.* $\mathtt{p} \rightharpoonup^+_{I_M} \mathtt{r}$ *and* $\{\mathtt{p}_1, \ldots, \mathtt{p}_n\}$ *is not universally complete w.r.t.* $\mathtt{r}$*,* $\mathtt{p}_i \in \mathtt{W}$*,* $i = 1, \ldots, n$*.*

Definition 3.6 (Existential completeness error)**.** *Let* $\mathtt{W}$ *be a Web site and* (I_M, I_N, R) *be a Web specification. Then the triple* $(\mathtt{r}, \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}, \mathtt{E})$ *is an* existential completeness error evidence*, if there exists* $\mathtt{p} \in \mathtt{W}$ *s.t.* $\mathtt{p} \rightharpoonup^+_{I_M} \mathtt{r}$ *and* $\{\mathtt{p}_1, \ldots, \mathtt{p}_n\}$ *is not existentially complete w.r.t.* $\mathtt{r}$*,* $\mathtt{p}_i \in \mathtt{W}$*,* $i = 1, \ldots, n$*.*

Note that Definition 3.5 (resp. Definition 3.6) formalizes the fact that the Web site $\mathtt{W}$ fails to fulfil the requirement that a piece of information must occur in *all* (resp. *some*) Web pages of a given subset of $\mathtt{W}$. We denote by $E_M(\mathtt{W})$ the set containing all the completeness error evidences w.r.t. I_M for a Web site $\mathtt{W}$ (missing Web pages as well as universal/existential completeness errors evidences). When no confusion can arise, we just write E_M.

The verification methodology of [1] generates the sets of correctness and completeness error evidences E_N and E_M mentioned above for a given Web site w.r.t. the input Web specification. Starting from these sets, in the following section we formulate a method for fixing the errors and delivering a Web site which is *correct* and *complete* w.r.t. the intended Web specification.

Definition 3.7 (Web site correctness)**.** *Given a Web specification* (I_M, I_N, R)*, a Web site* $\mathtt{W}$ *is* correct *w.r.t.* (I_M, I_N, R) *iff the set* E_N *of correctness error evidences w.r.t.* I_N *is empty.*

Definition 3.8 (Web site completeness)**.** *Given a Web specification* (I_M, I_N, R)*, a Web site* $\mathtt{W}$ *is* complete *w.r.t.* (I_M, I_N, R) *iff the set* E_M *of completeness error evidences w.r.t.* I_M *is empty.*

4. Repairing a faulty Web site

Given a faulty Web site $\mathtt{W}$ and the sets of errors $\mathtt{E}_N$ and $\mathtt{E}_M$, our goal is to modify the given Web site by adding, changing, and removing information in order to produce a Web site that is correct and complete w.r.t. the considered Web specification. For this purpose, in correspondence with the error categories distinguished in the previous section, we introduce a catalogue of *repair actions* which can be applied to the faulty Web site. Therefore, in our framework, fixing a Web site consists in selecting a set of suitable repair actions that are automatically generated, and executing them in order to remove inconsistencies and wrong data from the Web site. The primitive repair actions we consider are the following: **change**$(\mathtt{p}, w, \mathtt{t})$ which replaces the subterm $\mathtt{p}|_w$ in $\mathtt{p}$ with the term $\mathtt{t}$; **insert**$(\mathtt{p}, w, \mathtt{t})$ which modifies the term $\mathtt{p}$ by adding the term $\mathtt{t}$ into $\mathtt{p}_{|w}$; **add**$(\mathtt{p}, \mathtt{W})$ which adds the Web page $\mathtt{p}$ to the Web site $\mathtt{W}$; **delete**$(\mathtt{p}, t)$ which deletes all the occurrences of the term $\mathtt{t}$ in the Web page $\mathtt{p}$. Each repair action returns the modified/added Web page after its execution.

Note that it is possible that a particular error could be repaired by means of different actions. For instance, a correctness error can be fixed by deleting the incorrect/forbidden information, or by changing the data which rise that error. Similarly, a completeness error can be fixed by either 1) inserting the missing information, or 2) deleting all the data in the Web site that caused that error. Moreover, modifying or

inserting arbitrary information may cause the appearance of new correctness errors. In order to avoid this, we have to ensure that the data considered for insertion are *safe* w.r.t. the Web specification, i.e. they cannot fire any correctness rule. For this purpose, we introduce the following definition.

Definition 4.1. *Let (I_M, I_N, R) be a Web specification and $\mathtt{p} \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ be a Web page. Then,* $\mathtt{p}$ *is* safe *w.r.t. I_N, iff for each $w \in O_{\mathcal{T}ag}(\mathtt{p})$ and $(\mathtt{l} \rightharpoonup \mathtt{r} \mid \mathtt{C}) \in I_N$, either (i) there is no σ s.t. $\mathtt{l}\sigma \trianglelefteq \mathtt{p}_{|w}$; or (ii) $\mathtt{l}\sigma \trianglelefteq \mathtt{p}_{|w}$, but $\mathtt{C}\sigma$ does not hold.*

In the following, we develop a repairing methodology which gets rid of both, correctness and completeness errors. We proceed in two main phases. First, we deal with correctness errors. Some repair actions are automatically inferred and run in order to remove the wrong information from the Web site. After this phase, we will end up with a correct Web site which still can be incomplete. At this point, other repair actions are synthesized and executed in order to provide Web site completeness.

4.1. Fixing correctness errors

Throughout this section, we will consider a given Web site $\mathtt{W}$, a Web specification (I_M, I_N, R) and the set $E_N \neq \emptyset$ of the correctness error evidences w.r.t. I_N for $\mathtt{W}$. Our goal is to modify $\mathtt{W}$ in order to generate a new Web site which is correct w.r.t. (I_M, I_N, R). We proceed as follows: whenever a correctness error is found, we choose a possible repair action (among the different actions described below) and we execute it in order to remove the erroneous information, provided that it does not introduce any new bug.

Given $\mathtt{e} = (\mathtt{p}, w, \mathtt{l}, \sigma) \in E_N$, $\mathtt{e}$ can be repaired in two distinct ways: we can decide either 1) to remove the wrong content $\mathtt{l}\sigma$ from the Web page $\mathtt{p}$ (specifically, from $\mathtt{p}_{|w}$), or 2) to change $\mathtt{l}\sigma$ into a piece of correct information. Hence, it is possible to choose between the following repair strategies.

"Correctness through Deletion" strategy. In this case, we simply remove all the occurrences of the subterm $\mathtt{p}_{|w}$ of the Web page $\mathtt{p}$ containing the wrong information $\mathtt{l}\sigma$ by applying the repair action **delete**$(\mathtt{p}, \mathtt{p}_{|w})$.[3]

Example 4.1. *Consider the Web site in Figure 1 and the Web specification in Example 3.1. The term $\mathtt{l}\sigma \equiv \mathtt{p}_{|1.4} \equiv \mathtt{blink(year(2003))}$ embedded in the Web page* (5) *of* $\mathtt{W}$ *(which is also called* $\mathtt{p}$ *in this example) generates a correctness error evidence $(\mathtt{p}, 1.4, \mathtt{l}, \sigma)$ w.r.t. the rule $\mathtt{blink(x)} \rightharpoonup \mathtt{error}$ and hence a delete action will remove from* $\mathtt{p}$ *the subterm* $\mathtt{blink(year(2003))}$.

[3]Note that, instead of removing the whole subterm $\mathtt{p}_{|w}$, it would be also possible to provide a more precise though also time-expensive implementation of the **delete** action which only gets rid of the part $\mathtt{l}\sigma$ of $\mathtt{p}_{|w}$ which is responsible for the correctness error.

"Correctness through Change" strategy. Given a correctness error $\mathtt{e} = (\mathtt{p}, w, \mathtt{l}, \sigma) \in E_N$, we replace the subterm $\mathtt{p}_{|w}$ of the Web page $\mathtt{p}$ with a new term t introduced by the user. The new term t must fulfill some conditions which are automatically provided and checked in order to guarantee the correctness of the inserted information. In the following we show how to compute such constraints.

Roughly speaking, we first ensure that (i) t does not embed subterms which might fire some correctness rule (*local correctness property*). Next, we have to guarantee that t, within the context surrounding it, will not cause any new correctness error (*global correctness property*).

Local correctness property. We handle conditional and unconditional correctness rules separately. For conditional rules, we must look for solutions to the following problem.

Let us consider the correctness error evidence $\mathtt{e} = (\mathtt{p}, w, \mathtt{l}, \sigma) \in E_N$ and the associated repair action $\mathbf{change}(\mathtt{p}, w, t)$. We build the set of conditions

$$CS_{\mathtt{e}} \equiv \{\neg\mathtt{C} \mid \exists \quad (\mathtt{l} \rightharpoonup \mathtt{r} \mid \mathtt{C}) \in I_N, \text{ a position } w', \text{ a substitution } \sigma \text{ s.t. } \mathtt{l}\sigma \trianglelefteq \mathtt{p}_{|w.w'}\}$$

We call $CS_{\mathtt{e}}$ the *constraint satisfaction problem* associated with $\mathtt{e}$. Roughly speaking, $CS_{\mathtt{e}}$ is obtained by collecting and negating all the conditions of those rules which detect correctness errors in $\mathtt{p}_{|w}$. Such collection of constraints, that can be solved manually or automatically by means of an appropriate constraint solver [3], can be used to provide suitable values for the term t to be inserted. We say that $CS_{\mathtt{e}}$ is *satisfiable* iff there exists at least one assignment of values for the variables occurring in $CS_{\mathtt{e}}$ that satisfies all the constraints. We denote by $Sol(CS_{\mathtt{e}})$ the set of all the assignments that verify the constraints in $CS_{\mathtt{e}}$. The restriction of $Sol(CS_{\mathtt{e}})$ to the variables occurring in σ is denoted by $Sol(CS_{\mathtt{e}})_{|\sigma}$. Let us see an example.

Example 4.2. *Consider the Web site* $\mathtt{W}$ *in Figure 1 and the Web specification of Example 3.1. The following subterm of Web page* (6)

```
project(pname(A1),grant1(1000),grant2(200),
        total(1100),
        coordinator(fullname(mariorossi)))
```

causes a correctness error $\mathtt{e}$ *w.r.t. the rule*

$$\mathtt{project(grant1(X), grant2(Y), total(Z))} \rightharpoonup \mathtt{error} \mid \mathtt{X + Y \neq Z}.$$

The error can be fixed by changing the values of the grants and the total amount, according to the solution of the constraint satisfaction problem $CS_{\mathtt{e}}$ that follows.

```
{X+Y=Z, X=Y*2, Z < 500000}
```

The constraints in $CS_{\mathtt{e}}$ come from the conditions of the last three rules of the considered Web specification. An admissible solution, which can be chosen by the user, might be

$$\{\mathtt{X/1000}, \mathtt{Y/500}, \mathtt{Z/1500}\} \in Sol(CS)$$

and the term t to be inserted might be

```
project(pname(A1),grant1(1000),grant2(500),
        coordinator(fullname(mariorossi)),
        total(1500))
```

which does not contain incorrect data.

Sometimes $CS_{\mathtt{e}}$ might be not solvable, since there are two or more correctness rules demanding incompatible conditions for correctness, and thus the user is asked to fix the Web specification before proceeding.

Let us now consider unconditional rules. Example 4.2 shows how to get rid of a correctness error just by changing some values which occur into a piece of wrong information. However, sometimes we may need to change not only the values of the variables but also the structure of the term containing the erroneous data. In this case, it might happen that we introduce a "forbidden" structure which can fire some unconditional correctness rule. Note that conditional rules cannot introduce incorrect data in t, since we assume that t has been chosen according to the solution of the constraint satisfaction problem $CS_{\mathtt{e}}$. Therefore, in order to ensure correctness, the following *structural correctness property* on the structure of term t must hold for unconditional rules.

$$\forall\, \mathtt{l} \rightharpoonup \mathtt{r} \in I_N, w \in O_{\mathcal{T}ag}(t), \text{substitution } \sigma, \mathtt{l}\sigma \not\trianglelefteq t_{|w}. \tag{1}$$

Roughly speaking, the structural correctness property (1) defined above ensures that no unconditional correctness rule can be triggered on t. This is achieved by checking that no left-hand side of an unconditional correctness rule is embedded into t.

Example 4.3. *Consider again Example 4.2 and the following term, which modifies the values and the structure of the faulty Web page.*

```
project(pname(A1),grant1(1000),grant2(500),
        projectleader(surname(gialli),
        name(anna)),blink(total(1500)))
```

If we decided to replace the wrong information with the term above, then the structural correctness property (1) would not be fulfilled, since the term `blink(total(1200))` *would fire rule* `(6)` *of the Web specification.*

Now we are ready to formalize the local condition for the "correctness through change" strategy.

Definition 4.2. *Given $\mathtt{e} = (\mathtt{p}, w, \mathtt{l}, \sigma) \in E_N$ and a repair action $\mathbf{change}(\mathtt{p}, w, t)$, we say that $\mathbf{change}(\mathtt{p}, w, t)$* obeys the local correctness property *iff*

- *for each conditional rule $(\mathtt{l} \rightharpoonup \mathtt{r} \mid \mathtt{C}) \in I_N$, $\mathtt{C} \neq \emptyset$, substitution σ' and position w', if $\mathtt{l}\sigma' \trianglelefteq t_{|w'}$ then*
  1. *$\sigma' \in Sol(CS_{\mathtt{e}})_{|\sigma'}$, when $\neg\mathtt{C} \in CS_{\mathtt{e}}$;*
  2. *$\mathtt{C}\sigma'$ does not hold, when $\neg\mathtt{C} \notin CS_{\mathtt{e}}$.*
- *for unconditional rules, the structural correctness property (1) holds.*

The local correctness property guarantees that a change action is "locally safe", in other words the term t which replaces the wrong information is safe w.r.t. I_N as stated by the following proposition.

Proposition 4.1. *Let (I_M, I_N, R) be a Web specification, $\mathtt{e} = (\mathtt{p}, w, \mathtt{l}, \sigma) \in E_N$ and $\mathbf{change}(\mathtt{p}, w, t)$ be a repair action. If $\mathtt{p}' \equiv \mathbf{change}(\mathtt{p}, w, t)$ obeys the local correctness property, then $\mathtt{p}'_{|w} \equiv t$ is safe w.r.t. I_N.*

Note that it is possible to repair several errors simultaneously by applying one single change action, since a change action affecting a term p may eventually fix all the correctness errors which occur in all the subterms of p. More formally, the following result holds.

Proposition 4.2. *Let (I_M, I_N, R) be a Web specification, $\mathtt{e_i} = (\mathtt{p}, w_i, \mathtt{l}_i, \sigma_i) \in E_N$, $i = 1, \ldots, n$, $w_1 \leq w_2 \leq \ldots \leq w_n$, and $\mathtt{p}' \equiv \mathbf{change}(\mathtt{p}, w_1, t)$ be a repair action that obeys the local correctness property. Then, $\mathtt{p}'_{|w_1} \equiv t$ is safe w.r.t. I_N.*

Now we proceed with the

Global correctness property. Whenever we fix some wrong data by executing a repair action $\mathbf{change}(\mathtt{p}, w, t)$, it is not enough to ensure that the term t to be introduced is locally safe (local correctness property), we also need to consider t within the context that surrounds it in p. If we don't pay attention to such a global condition, some subtle correctness errors might arise as witnessed by the following example.

Example 4.4. *Consider the Web page $\mathtt{p} \equiv \mathtt{f}(\mathtt{g}(\mathtt{a}), \mathtt{b}, \mathtt{h}(\mathtt{c}))$, and the following correctness rule set*

$$I_N \equiv \{(\mathtt{r1})\ \mathtt{f}(\mathtt{g}(\mathtt{b})) \rightharpoonup \mathtt{error}, (\mathtt{r2})\ \mathtt{g}(\mathtt{a}) \rightharpoonup \mathtt{error}\}.$$

The Web page p *contains a correctness error according to rule* `(r2)`. *The Web page $\mathtt{f}(\mathtt{g}(\mathtt{b}), \mathtt{b}, \mathtt{h}(\mathtt{c}))$ is obtained from* p *by executing, for instance, the repair action*

$$\mathbf{change}(\mathtt{f}(\mathtt{g}(\mathtt{a}), \mathtt{b}, \mathtt{h}(\mathtt{c})), 1, \mathtt{g}(\mathtt{b})).$$

Although the term $\mathtt{g}(\mathtt{b})$ is safe w.r.t. I_N (i.e. it does not introduce any new correctness error), the replacement of $\mathtt{g}(\mathtt{a})$ with $\mathtt{g}(\mathtt{b})$ in p *produces a new correctness error which is recognizable by rule* `(r1)`.

In order to avoid such kinds of undesirable repairs, we define the following global correctness property, which simply prevents a new term t from firing any correctness rule when inserted in the Web page to be fixed.

The following definition is auxiliary. Let $\mathtt{s}, \mathtt{t} \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ s.t. $\mathtt{s} \trianglelefteq \mathtt{t}$. We define the set $Emb_{\mathtt{s}}(\mathtt{t})$ as the set of all the positions in $\mathtt{t}$ which embed some subterm of $\mathtt{s}$. For instance, consider the terms $f(k, g(c))$, and $f(b, g(c), k)$. Then, $f(k, g(c)) \trianglelefteq f(b, g(c), k)$, and $Emb_{f(k,g(c))}(f(b, g(c), k)) = \{\Lambda, 2, 2.1, 3\}$.

Definition 4.3. *Let (I_M, I_N, R) be a Web specification, $\mathtt{p}' \equiv \mathbf{change}(\mathtt{p}, w, t)$ be a repair action producing the Web page $\mathtt{p}'$. Then, $\mathbf{change}(\mathtt{p}, w, t)$* obeys the global correctness property *if, for each correctness error evidence $\mathtt{e} = (\mathtt{p}', w', \mathtt{l}, \sigma)$ w.r.t. I_N such that $w' \leq w$,*

$$\{w\}.O_{\mathcal{T}ag}(t) \cap \{w'\}.Emb_{\mathtt{l}\sigma}(\mathtt{p}'_{|w'}) = \emptyset$$

The idea behind Definition 4.3 is that any error $\mathtt{e}$ in the new page $\mathtt{p}' \equiv \mathbf{change}(\mathtt{p}, w, t)$, obtained by inserting term t within $\mathtt{p}$, is not a consequence of this change but already present in a different sub-term of $\mathtt{p}$. For this purpose, we require that (the set of positions of) the wrong information $\mathtt{l}\sigma$ does not "overlap" the considered term t.

The execution of a change action which obeys the global as well as the local correctness property, decreases the number of correctness errors of the original Web site as stated by the following proposition.

Proposition 4.3. *Let (I_M, I_N, R) be a Web specification and $\mathtt{W}$ be a Web site. Let $E_N(\mathtt{W})$ be the set of correctness error evidences w.r.t. I_N of $\mathtt{W}$, and $(\mathtt{p}, w, \mathtt{l}, \sigma) \in E_N(\mathtt{W})$. By executing a repair action $\mathbf{change}(\mathtt{p}, w, t)$, which obeys the local as well as the global correctness property, we have that $|E_N(\mathtt{W}')| < |E_N(\mathtt{W})|$ where $\mathtt{W}' \equiv \mathtt{W} \setminus \{\mathtt{p}\} \cup \{\mathbf{change}(\mathtt{p}, w, t)\}$.*

4.2. Fixing completeness errors

In this section, we address the problem of repairing an incomplete Web site $\mathtt{W}$. Without loss of generality, we assume that $\mathtt{W}$ is an incomplete but correct Web site w.r.t. a given Web specification (I_M, I_N, R). Such an assumption will allow us to design a repair methodology which "completes" the Web site and does not introduce any incorrect information.

Let $E_M(\mathtt{W})$ be the set of completeness error evidences risen by I_M for the Web site $\mathtt{W}$. Any completeness error evidence belonging to $E_M(\mathtt{W})$ can be repaired following distinct strategies and thus by applying distinct repair actions. On the one hand, we can think of adding the needed data, whenever a Web page or a piece of information in a Web page is missing. On the other hand, all the information that caused the error might be removed to get rid of the bug. In both cases, we must ensure that the execution of the chosen repair action does not introduce any new correctness/completeness error to guarantee the termination and the soundness of our methodology.

"Completeness through Insertion" strategy. We consider two distinct kinds of repair actions, namely **add**$(\mathtt{p}, \mathtt{W})$ and **insert**$(\mathtt{p}, w, t)$, according to the kind of completeness error we have to fix. The former action adds a new Web page $\mathtt{p}$ to a Web site $\mathtt{W}$ and thus will be employed whenever the system has to fix a given missing Web page error. The latter allows us to add a new piece of information t to (a subterm of) an incomplete Web page $\mathtt{p}$, and therefore is suitable to repair universal as well as existential completeness errors. More specifically, the insertion repair strategy works as follows.

Missing Web page errors. Given a missing Web page error evidence $(\mathtt{r}, \mathtt{W})$, we fix the bug by adding a Web page $\mathtt{p}$, which embeds the missing expression $\mathtt{r}$, to the Web site $\mathtt{W}$. Hence, the Web site $\mathtt{W}$ will be "enlarged" by effect of the following **add** action: $\mathtt{W} = \mathtt{W} \cup \{\mathbf{add}(\mathtt{p}, \mathtt{W})\}$, where $\mathtt{r} \trianglelefteq \mathtt{p}_{|w}$ for some $w \in O_{\mathcal{T}ag}(\mathtt{p})$.

Existential completeness errors. Given an existential completeness error evidence $(\mathtt{r}, \{\mathtt{p}_1, \mathtt{p}_2, \ldots, \mathtt{p}_n\}, \mathtt{E})$, we fix the bug by inserting a term t, that embeds the missing expression $\mathtt{r}$, into an arbitrary page $\mathtt{p}_i$, $i = 1, \ldots, n$. The position of the new piece of information t in $\mathtt{p}_i$ is typically provided by the user, who must supply a position in $\mathtt{p}_i$ where t must be attached. The **insert** action will transform the Web site $\mathtt{W}$ in the following way: $\mathtt{W} = \mathtt{W} \setminus \{\mathtt{p}_i\} \cup \{\mathbf{insert}(\mathtt{p}_i, w, t)\}$, where $\mathtt{r} \trianglelefteq \mathtt{p}_{i|w}$ for some $w \in O_{\mathcal{T}ag}(\mathtt{p})$.

Universal completeness errors. Given a universal completeness error evidence $(\mathtt{r}, \{\mathtt{p}_1, \mathtt{p}_2, \ldots, \mathtt{p}_n\}, \mathtt{A})$, we fix the bug by inserting a term t_i, that embeds the missing expression $\mathtt{r}$, into every Web page $\mathtt{p}_i$, $i = 1, \ldots, n$ not embedding $\mathtt{r}$. The position of the new piece of information t_i in each $\mathtt{p}_i$ is typically provided by the user, who must supply a postition w_i in $\mathtt{p}_i$ where t_i must be attached. In this case, we will execute a sequence of **insert** actions, exactly one for each incomplete Web page $\mathtt{p}_i$. Therefore, the Web site $\mathtt{W}$ will be transformed in the following way. For each $\mathtt{p}_i \in \{\mathtt{p}_1, \mathtt{p}_2, \ldots, \mathtt{p}_n\}$ such that $\mathtt{r} \not\trianglelefteq \mathtt{p}_{i|w_j}$ for each $w_j \in O_{\mathcal{T}ag}(\mathtt{p}_i)$, $\mathtt{W} = \mathtt{W} \setminus \{\mathtt{p}_i\} \cup \{\mathbf{insert}(\mathtt{p}_i, w_i, t_i)\}$, where $\mathtt{r} \trianglelefteq \mathtt{p}_{|w_i}$ for some $w_i \in O_{\mathcal{T}ag}(\mathtt{p}_i)$.

Both the add action and the insert action introduce new information in the Web site which might be potentially dangerous, since it may contain erroneous as well as incomplete data. It is therefore important to constrain the kind of information a user can add. In order to preserve correctness, we compel the user to only insert safe information in the sense of Definition 4.1. Hence, the new data being added by the

execution of some repair action cannot fire a correctness rule subsequently.

Proposition 4.4. *Let (I_M, I_N, R) be a Web specification and* $\mathtt{W}$ *be a correct Web site w.r.t. (I_M, I_N, R). Let* $\mathtt{p}_1 \equiv \mathbf{insert}(\mathtt{p}, w, t)$ *and* $\mathtt{p}_2 \equiv \mathbf{add}(\mathtt{p}_2, \mathtt{W})$.

- *If* $\mathtt{p}_1$ *is safe w.r.t. I_N, then* $\mathtt{W} \setminus \{\mathtt{p}\} \cup \{\mathtt{p}_1\}$ *is correct w.r.t. (I_M, I_N, R).*
- *If* $\mathtt{p}_2$ *is safe w.r.t. I_N, then* $\mathtt{W} \cup \{\mathtt{p}_2\}$ *is correct w.r.t. (I_M, I_N, R).*

Additionally, we want to prevent the execution of the repair actions from introducing new completeness errors, that is, we just want to fix all and only the initial set of completeness errors of the Web site $\mathtt{W}$, namely $E_M(\mathtt{W})$. Given a completeness error evidence $\mathtt{e}$, we use notation $\mathtt{e}(\mathtt{r})$ to make evident the unsatisfied requirement $\mathtt{r}$ signaled by $\mathtt{e}$.

Definition 4.4. *Let (I_M, I_N, R) be a Web specification and* $\mathtt{W}$ *be a Web site w.r.t. (I_M, I_N, R). Let $E_M(\mathtt{W})$ be the set of completeness error evidences of* $\mathtt{W}$ *w.r.t. I_M.*

- *the repair action* $\mathtt{p}_1 \equiv \mathbf{insert}(\mathtt{p}, w, t)$ *is* acceptable *w.r.t. (I_M, I_N, R) and* $\mathtt{W}$ *iff*
  1. $\mathtt{p}_1$ *is safe w.r.t. (I_M, I_N, R);*
  2. $\mathtt{r} \trianglelefteq t_{|w}$, $w \in O_{\mathcal{T}ag}(t)$, *for some* $\mathtt{e}(\mathtt{r}) \in E_M(\mathtt{W})$*;*
  3. *if* $\mathtt{W}' \equiv \mathtt{W} \setminus \{\mathtt{p}\} \cup \{\mathtt{p}_1\}$, *then* $E_M(\mathtt{W}') \subset E_M(\mathtt{W})$.
- *the repair action* $\mathtt{p}_2 \equiv \mathbf{add}(\mathtt{p}_2, \mathtt{W})$ *is* acceptable *w.r.t. (I_M, I_N, R) and* $\mathtt{W}$ *iff*
  1. $\mathtt{p}_2$ *is safe w.r.t. (I_M, I_N, R);*
  2. $\mathtt{r} \trianglelefteq \mathtt{p}_{2|w}$, $w \in O_{\mathcal{T}ag}(\mathtt{p}_2)$, *for some* $\mathtt{e}(\mathtt{r}) \in E_M(\mathtt{W})$*;*
  3. *if* $\mathtt{W}' \equiv \mathtt{W} \cup \{\mathtt{p}_2\}$, *then* $E_M(\mathtt{W}') \subset E_M(\mathtt{W})$.

Definition 4.4 guarantees that the information which is added by insert and add actions is correct and does not yield any new completeness error. More precisely, the number of completeness errors decreases by effect of the execution of such repair actions.

Example 4.5. *Consider the Web specification of Example 3.1 and the universal completeness error evidence* $(\mathtt{hp}(\mathtt{status}(\mathtt{professor}), \mathtt{teaching}), \mathtt{p}_1, \mathtt{p}_2, \mathtt{A})$ *where* $\mathtt{p}_1$ *and* $\mathtt{p}_2$ *are the home pages of* Mario Rossi *and* Anna Gialli *in the Web site of Figure 1. To fix the error, we should add some information to Web page (2), while Web page (4) is complete w.r.t. the requirement* $\mathtt{hp}(\mathtt{status}(\mathtt{professor}), \mathtt{teaching})$. *Consider the pieces of information*

$$t_1 \equiv \mathtt{teaching}(\mathtt{course}(\mathtt{title}(\mathtt{logic}), \mathtt{syllabus}(\mathtt{blah})))$$
$$t_2 \equiv \mathtt{teaching}(\mathtt{courselink}(\mathtt{url}(\mathtt{www.mycourse.com}), \mathtt{urlname}(\mathtt{Logic}))).$$

If we introduce term t_1, the corresponding insert action is acceptable. However, inserting term t_2 would produce a new completeness error (i.e. a broken link error).

"Completeness through Deletion" strategy. When dealing with completeness errors, sometimes it is more convenient to delete incomplete data instead of completing them. In particular, this option can be very useful, whenever we want to get rid of out-of-date information as illustrated in Example 4.6 below. The main idea of the deletion strategy is to remove all the information in the Web site that caused a given completeness error. The strategy is independent of the kind of completeness error we are handling, since the missing information is computed in the same way for all the three kinds of errors by partially rewriting the original Web pages of the Web site. In other words, given the missing expression $\mathtt{r}$ of a completeness error evidence $\mathtt{e}(\mathtt{r})$ (that is, a missing page error evidence $(\mathtt{r}, \mathtt{W})$, or an existential completeness error evidence $(\mathtt{r}, \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}, \mathtt{E})$, or a universal completeness error evidence $(\mathtt{r}, \{\mathtt{p}_1, \ldots, \mathtt{p}_n\}, \mathtt{A})$), there exists a Web page $\mathtt{p} \in \mathtt{W}$ such that $\mathtt{p} \rightharpoonup^+ \mathtt{r}$. Therefore, we proceed by computing and eliminating from the Web pages all the terms occurring in the partial rewrite sequences that lead to a missing expression $\mathtt{r}$. Since this problem is generally undecidible, some restrictions on the specification language must be considered (for more details, see the *bounded* specifications in [1, 2]).

More formally, given a Web specification (I_M, I_N, R), a Web site $\mathtt{W}$ and a completeness error evidence $\mathtt{e}(\mathtt{r})$, the Web site $\mathtt{W}$ will change in the following way.

For each $t_1 \rightharpoonup t_2 \rightharpoonup \ldots \rightharpoonup \mathtt{r}$, where $t_1 \trianglelefteq \mathtt{p}_{|w}$, $w \in O_{\mathcal{T}ag}(\mathtt{p})$, $\mathtt{p} \in \mathtt{W}$

$$\mathtt{W} \equiv \{\mathtt{p} \in \mathtt{W} \mid t_i \not\trianglelefteq \mathtt{p}_{|w}, \forall\, w \in O_{\mathcal{T}ag}(\mathtt{p}), i = 1, \ldots, n\} \cup \{\mathbf{delete}(\mathtt{p}, t_i) \mid \mathtt{p} \in \mathtt{W}, t_i \trianglelefteq \mathtt{p}_{|w}, w \in O_{\mathcal{T}ag}(\mathtt{p}) \; i = 1, \ldots, n\}$$

Example 4.6. *Consider the Web specification of Example 3.1, the Web site* $\mathtt{W}$ *of Figure 1 and the missing Web page error evidence* $(\mathtt{hpage}(\mathtt{fullname}(\mathtt{ugoblu}), \mathtt{status}), \mathtt{W})$*, which can be detected in* $\mathtt{W}$ *by using the completeness rules in I_M. The missing information is obtained by means of the following partial rewrite sequence:*

$$\mathtt{pub}(\mathtt{name}(\mathtt{ugo}), \mathtt{surname}(\mathtt{blu}), \mathtt{title}(\mathtt{blah1}), \mathtt{blink}(\mathtt{year}(2003))) \rightharpoonup \mathtt{member}(\mathtt{name}(\mathtt{ugo}), \mathtt{surname}(\mathtt{blu})) \rightharpoonup \mathtt{hpage}(\mathtt{fullname}(\mathtt{ugoblu}), \mathtt{status})$$

By choosing the deletion strategy, we would delete all the information regarding the group membership and the publications of Ugo Blu *from the Web site.*

As in the case of the insertion strategy, we have to take care about the effects of the execution of the repair actions.

More precisely, we do not want the execution of any **delete** action to introduce new completeness errors. For this purpose, we consider the following notion of *acceptable* delete action.

Definition 4.5. *Let (I_M, I_N, R) be a Web specification and $\mathtt{W}$ be a Web site w.r.t. (I_M, I_N, R). Let $E_M(\mathtt{W})$ be the set of completeness error evidences of $\mathtt{W}$ w.r.t. $I_M(\mathtt{W})$. The repair action $\mathtt{p}_1 \equiv \mathbf{delete}(\mathtt{p}, t)$ is* acceptable *w.r.t. (I_M, I_N, R) and $\mathtt{W}$ iff $\mathtt{W}' \equiv \mathtt{W} \setminus \{\mathtt{p}\} \cup \{\mathtt{p}_1\}$ implies $E_M(\mathtt{W}') \subset E_M(\mathtt{W})$.*

5. Implementation

The basic methodology presented so far has been partially implemented in the preliminary prototype GVERDI-R (Graphical Verification and Rewriting for Debugger Internet sites), which is written in Haskell (GHC v6.2.2) and publicly available together with a set of examples at http://www.dsic.upv.es/users/elp/GVerdi. The system worked in a very satisfactory way on our experiments, including all the examples in this paper. We are currently coupling the system with appropriate constraint solvers in order to help the user to input correct value assignments under the "Correctness through Change" strategy.

6. Conclusions

Maintaining contents of Web sites is an open and urgent problem since outdated, incorrect and incomplete information is becoming very frequent in the World Wide Web. In this paper, we presented a semi–automatic methodology for repairing Web sites which has a number of advantages over other potential approaches (and hence can be used as a useful complement to them): 1) in contrast to the active database Web management techniques, we are able to predict whether a repair action can cause new errors to appear and assist the user in reformulating the action; 2) by solving the constraint satisfaction problem associated to the conditions of the Web specification rules, we are also able to aid users to fix erroneous information by suggesting ranges of correct values; 3) our methodology smoothly integrates on top of existing rewriting-based web verification frameworks such as [1, 2], which offer the expressiveness and computational power of functions and allow one to avoid the encumbrances of DTDs and XML rule languages.

Let us conclude by discussing further work. For the benefit of the user who prefers not to express himself in a formal language, we are currently working on a graphical notation for rules and for the repair actions, which will be automatically translated into our formalism. Moreover, to increase the level of automation of our repair method, we are working on possible correction strategies which minimize both the amount of information to be changed and the number of repair actions to be executed.

References

[1] M. Alpuente, D. Ballis, and M. Falaschi. Automated Verification of Web Sites Using Partial Rewriting. *Software Tools for Technology Transfer*, 2006. Accepted for publication. To appear.

[2] M. Alpuente, D. Ballis, and M. Falaschi. A Rewriting-based Framework for Web Sites Verification. ENTCS, 124(1), 2005.

[3] K. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[4] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[5] L. Bertossi and J. Pinto. Specifying Active Rules for Database Maintenance. In *Proc. of FMLDO '99*, LNCS 1773, pp. 112–129. Springer, 1999.

[6] M. Bezem. *TeReSe, Term Rewriting Systems*, chapter Mathematical background (Appendix A). Cambridge University Press, 2003.

[7] L. Capra, W. Emmerich, A. Finkelstein, and C. Nentwich. XLINKIT: a Consistency Checking and Smart Link Generation Service. *ACM TOIT*, 2(2):151–185, 2002.

[8] N. Dershowitz and D. Plaisted. Rewriting. *Handbook of Automated Reasoning*, 1:535–610, 2001.

[9] Imagiware. Doctor HTML. Available at: `http://www.doctor-html.com/`.

[10] J. Klop. Term Rewriting Systems. In *Handbook of Logic in Computer Science*, vol. I, pp. 1–112. Oxford University Press, 1992.

[11] M. Leuschel. Homeomorphic Embedding for Online Termination of Symbolic Methods. In *The Essence of Computation*, LNCS 2566, pp. 379–403. Springer, 2002.

[12] E. Mayol and E. Teniente. A Survey of Current Methods for Integrity Constraint Maintenance and View Updating. In *Proc. of ER '99*, LNCS 1727, pp. 62–73. Springer, 1999.

[13] C. Nentwich, W. Emmerich, and A. Finkelstein. Consistency Management with Repair Actions. In *Proc. of ICSE '03*. IEEE Computer Society, 2003.

[14] S. Nesbit. HTML Tidy: Keeping it clean, 2000. Available at `http://tidy.sourceforge.net`.

[15] J. Scheffczyk, , U. M. B. P. Rödig, and L. Schmitz. S-dags: Towards efficient document repair generation. In *Proc. of CCCT '04*, vol. 2, pp. 308–313, 2004.

[16] J. Scheffczyk, U. M. Borghoff, P. Rödig, and L. Schmitz. Consistent document engineering: formalizing type-safe consistency rules for heterogeneous repositories. In *Proc. of DocEng '03*, pp. 140–149. ACM Press, 2003.

[17] J. Scheffczyk, P. Rödig, U. M. Borghoff, and L. Schmitz. Managing inconsistent repositories via prioritized repairs. In *Proc. of DocEng '04*, pp. 137–146. ACM Press, 2004.

[18] WWW Consortium (W3C). Extensible Markup Language (XML) 1.0, second edition, 1999.

[19] WWW Consortium (W3C). Extensible HyperText Markup Language (XHTML), 2000.