

Fixing Web Sites Using Correction Strategies¹

D. Ballis

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

D. Romero

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

Abstract—The development and the maintenance of Web sites are difficult tasks. To maintain the consistency of ever-larger, complex Web sites, Web administrators need effective mechanisms that aid them in fixing every possible inconsistency. In this paper, we present an extension of a methodology for semi-automatically repairing faulty Web site which we developed in a previous work. As a novel contribution, we define two correction strategies with the aim of increasing the level of automation of our repair method. Specifically, the proposed strategies minimize both the amount of information to be changed and the number of repair actions to be executed in a faulty Web site to make it correct.

I. INTRODUCTION

A Web site needs constant updating and maintenance to evolve and stay current. Typically, these are not trivial tasks, since large and complex Web sites are nowadays quite common, and very often they are designed in a way that it is rather difficult to keep the information correct and up-to-date. We believe that formal approaches can bring many benefits to such important issues, giving support to 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 [1], a repair framework for inconsistent distributed documents is presented that complements the tool xlinkit [2]. 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 [3], [4] an extension for the tool CDET [5] 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 [6]. Current research in this field focuses on the derivation of active rules that automatically trigger repair actions leading to a consistent state after each update [7].

In our previous work on GVERDI [8], [9], [10], we presented a rewriting-like approach to Web site verification and correction. Our methodology allows us to automatically recognize erroneous information appearing in a Web site w.r.t. a given formal specification, and then to repair the detected bugs by means of the execution of a sequence of repair actions which are semi-automatically inferred by the system. Since different repair actions can be executed to repair the same error, our method is tuned to deliver a set of correct repair actions to choose between.

Our contribution. Our repair framework [9] along with all the other repair methodology we cited above do not investigate the relations/dependencies among errors. Such an analysis can be a potential source of optimization and may increase the level of automation of a repair system. Indeed, errors in a given Web site are often deeply interrelated. This fact suggests us that correcting a given bug may lead to an automatic fix of a “related” bug without executing any other repair action.

In this paper, we extend our repair methodology [9] in several ways. First, we carry out a systematic analysis on the relations among correctness errors, then we exploit it in order to formalize two possible correction strategies: the \mathcal{M} -strategy allows one to minimize the number of repair actions to be executed, while the \mathcal{MNO} -strategy reduces the amount of information to be changed/removed to fix the Web site. In both cases, it is worth noting that the number of errors we need to correct to get a repaired Web site W is much less than the total number of errors occurring in W . Consequently, employing such strategies guarantees a better performance of our repair system. To our knowledge, no repair system supports such kind of optimization based on repair strategies, although it can lead to a faster and simpler correction of the faulty Web site.

Plan of the paper. The rest of the paper is structured as follows. Section II summarizes some preliminary definitions and notations. In Section III, we formulate a simple method for translating XHTML/XML documents into Herbrand terms and we recall the notion of simulation used to recognize erroneous patterns inside the Web site. In Section IV, we formalize correctness errors that can be detected as an outcome of the verification technique. Section V provides an analysis of the relations among correctness errors, while in Section VI the

¹This work has been partially supported by the EU (FEDER) and Spanish MEC TIN-2004-7943-C04-02 project, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Daniel Romero is also supported by ALFA grant LERNet AML/19.0902/97/0666/II-0472-FA

notion of correction strategy is given and the \mathcal{M} -strategy as well as \mathcal{MNO} -strategy are formulated. Section VII concludes and discusses future work.

II. PRELIMINARIES

We call a finite set of symbols *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 [11] (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 . t_{1u} 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 . A *substitution* $\sigma \equiv \{X_1/t_1, X_2/t_2, \dots\}$ 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) $X_i\sigma = t_i$, $i = 1, \dots, n$, and (iii) $X\sigma = X$, for any $X \in \mathcal{V} \setminus \{X_1, \dots, X_n\}$. By ε we denote the *empty substitution*. Given a substitution σ , the *domain* of σ is the set $Dom(\sigma) = \{X | X\sigma \neq X\}$. Given the substitution σ_1 and σ_2 , such that $Dom(\sigma_2) \subseteq Dom(\sigma_1)$, by σ_1/σ_2 we define the substitution $\{X/t \in \sigma_1 \mid X \in Dom(\sigma_1) \setminus Dom(\sigma_2)\} \cup \{X/t \in \sigma_2 \mid X \in Dom(\sigma_1) \cap Dom(\sigma_2)\} \cup \{X/X \mid X \notin Dom(\sigma_1)\}$. An *instance* of a term t is defined as $t\sigma$, where σ is a substitution. 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 [12], [13]). 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 $\lambda \rightarrow \rho$, $\lambda, \rho \in \tau(\Sigma, \mathcal{V})$, $\lambda \notin \mathcal{V}$ and $Var(\rho) \subseteq Var(\lambda)$. We will often write just R instead of (Σ, R) . A rewrite step is the application of a rewrite rule to an expression. A term s *rewrites* to a term t via $r \in R$, $s \rightarrow_r t$ (or $s \rightarrow_R t$), if there exist a position $u \in O_\Sigma(s)$, $r \equiv \lambda \rightarrow \rho$, 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. $s \rightarrow t$). A term s is a *irreducible form* (or *normal form*) w.r.t. R , if there is no term t s.t. $s \rightarrow_R t$. t is the irreducible form of s w.r.t. R (in symbols $s \rightarrow_R^! t$) if $s \rightarrow_R^* t$ and t is irreducible.

We say that a TRS R is *terminating*, if there exists no infinite rewrite sequence $t_1 \rightarrow_R t_2 \rightarrow_R \dots$. A TRS R is

confluent if, for all terms s, t_1, t_2 , such that $s \rightarrow_R^* t_1$ and $s \rightarrow_R^* t_2$, there exists a term t s.t. $t_1 \rightarrow_R^* t$ and $t_2 \rightarrow_R^* t$. 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 $s \rightarrow_R^! z$ and $t \rightarrow_R^! z$. A *condition* is a finite set of equations. We say that a condition C *holds* in a canonical TRS R , if for each $eqn \in C$, eqn holds in R . The *empty condition* \emptyset trivially holds in any canonical TRS R .

III. DENOTATION OF WEB SITES

Let us consider two alphabets T and Tag . We denote the set T^* by $Text$. An object $t \in Tag$ is called *tag element*, while an element $w \in Text$ 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 $\tau(Text \cup Tag)$ (see Figure 1). Note that XML/XHTML tag attributes can be considered as common tagged elements, and hence translated in the same way. A *Web site* is a finite collection of ground terms $\{p_1 \dots p_n\}$.

In the following, we will also consider terms of the non-ground term algebra $\tau(Text \cup Tag, \mathcal{V})$, which may contain variables. An element $s \in \tau(Text \cup Tag, \mathcal{V})$ is called *Web page template*. In our methodology, Web page templates are used for specifying erroneous/incorrect patterns which may be recognized in the Web pages. In order to mechanize the detection of such patterns inside a Web site, we will employ a *tree simulation* mechanism which is described in the next section.

A. The simulation relation

The notion of *simulation* in our context allows us to analyze and extract the partial structure of a Web page which is subject to verification. 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.

Simulations have been used in a number of works dealing with querying, transformation, and verification of semistructured data (cf. [14], [15], [16], [17], [18]).

Our notion of simulation, \trianglelefteq , is an adaptation of Kruskal's *embedding* (or "syntactically simpler") relation [19] where we ignore the usual *diving rule*¹ [20].

Definition 3.1 (simulation): The *simulation* relation

$$\trianglelefteq \subseteq \tau(Text \cup Tag, \mathcal{V}) \times \tau(Text \cup Tag, \mathcal{V})$$

on Web page templates is the least relation satisfying the following rules:

- $X \trianglelefteq t$, for each $X \in \mathcal{V}$, $t \in \tau(Text \cup Tag, \mathcal{V})$;

¹The diving rule allows one to "strike out" a part of the term at the right-hand side of the relation \trianglelefteq . Formally, $s \trianglelefteq f(t_1, \dots, t_n)$, if $s \trianglelefteq t_i$, for some i .

<pre> <book> <code> GR </code> <title> The Art of Gardening </title> <author> <name> Gino </name> <surname> Rossi </surname> </author> <year> 2006 </year> </book> </pre>	<pre> book (code (GR) , title (The Art of Gardening) , author (name (Gino) , surname (Rossi ,) , year (2006))) </pre>
---	---

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

- $f(t_1, \dots, t_m) \sqsubseteq g(s_1, \dots, s_n)$ iff $f \equiv g$ and $t_i \sqsubseteq s_{\pi(i)}$, for $i = 1, \dots, m$, and some injective function $\pi : \{1, \dots, m\} \rightarrow \{1, \dots, n\}$.

Given two Web pages s_1 and s_2 , if $s_1 \sqsubseteq s_2$ we say that s_1 *simulates* (or *is embedded* or *recognized* into) s_2 . We also say that s_2 *embeds* s_1 . Note that, in Definition 3.1, for the case when m is 0 we have $c \sqsubseteq c$ for each constant symbol c .

Let us illustrate this definition by means of a rather intuitive example.

Example 3.1: Consider the following Web page templates (called s_1 and s_2 , respectively):

```

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

```

We observe that the structure of s_1 can be recognized inside the structure of s_2 , hence $s_1 \sqsubseteq s_2$, while $s_2 \not\sqsubseteq s_1$.

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

IV. ERRORS AND REPAIR ACTIONS

In our previous work [8], we defined a specification language along with a verification technique for the definition and the validation of formal properties over Web sites. Among the features our framework provides, it allows one to detect erroneous/forbidden information in a Web site yielding as outcome a set of correctness error evidences which basically represent pieces of faulty information (i.e. *correctness errors*). More formally, a *correctness error evidence* (also called *error evidence*) is a quintuple (p, w, l, σ, C) such that

- 1) $l\sigma$ is a ground instance of a faulty term l which is embedded in a subterm $p|_w$ of a given Web page p (in symbols, $l\sigma \sqsubseteq p|_w$);

- 2) C is a condition such that $Var(C) \subseteq Var(l)$ and $C\sigma$ holds in a given canonical TRS R .

In other words, $l\sigma$ represents the erroneous information which is contained in some subterm of the Web page p , whenever (an instance of) the associated condition holds; in this case, we say that (p, w, l, σ, C) *represents* a correctness error in p .

We denote the set of all the correctness error evidences which are detected in a Web page p by $\mathbb{E}_N(p)$.

In [9], we proposed a novel framework for fixing erroneous as well as incomplete information in a Web site. For this purpose, in correspondence with error evidences, we introduced a catalogue of *repair actions* which can be applied to the faulty Web site. The methodology infers a set of suitable repair actions that are (semi)automatically generated and executed in order to remove inconsistencies and wrong data from the Web site. The primitive repair actions we consider for repairing correctness errors are the following.

- **delete** (p, w, t) deletes the occurrence of the term t in the subterm $p|_w$ from the Web page p and returns the modified Web page.
- **change** (p, w, t) replaces the subterm $p|_w$ in p with the term t and returns the modified Web page.

The former repair action gets rid of a piece of wrong information included in a Web page by deleting it, whereas the latter allows us to replace the data that are responsible for the error. While performing a **delete** (p, w, t) action is always safe, since it does not introduce any new bug; the execution of a **change** (p, w, t) might generate brand new correctness error evidences, if we admit the insertion of arbitrary, unchecked information t . To this respect, in [9] we established conditions to guarantee a safe behaviour of any change action; global as well as local properties have been formalized in order to ensure that, after having run a change action, the number of correctness errors evidences decreases. More formally,

Definition 4.1: Let $p \in \tau(\mathcal{Text} \cup \mathcal{Tag})$ be a Web page. Then, the repair action $p' \equiv \mathbf{change}(p, w, t)$ is *safe*, iff $\mathbb{E}_N(t) = \emptyset$ and $|\mathbb{E}_N(p')| < |\mathbb{E}_N(p)|$, where $|S|$ is the function

returning the cardinality of a given set S .

Throughout this paper, we assume that change actions are always safe. We say that a Web page p is *repaired*, if $\mathbb{E}_N(p) = \emptyset$.

Example 4.1: Let us consider the (piece of) Web page of Figure 2(a). Such a document models some features regarding a book. Let us suppose that the code of every book must be computed by concatenating the surname of the author together with the publication year. Let R be a canonical TRS modeling the usual concatenation function $++$. Thus, p contains an error which can be easily recognized by running our verification system [10], which generates the following error correctness evidence:

$$(p, \Lambda, \text{book}(\text{code}(X), \text{author}(\text{surname}(Y)), \text{year}(Z)), \{X/GR, Y/Rossi, Z/2006\}, \{X = Y ++ Z\}).$$

A possible repair for such an error can be obtained by running the safe repair action **change**($p, 1, \text{code}(\text{Rossi}2006)$). The result of this operation is shown in Figure 2(b).

V. CORRECTNESS ERROR DEPENDENCIES

Typically, a given Web page can contain several correctness errors which may be somehow connected. Since the execution of a repair action might fix more related errors simultaneously, it is crucial to discover whether an error depends on other errors. In this section, we analyze the dependencies among error correctness evidences. Later on, we will exploit this information in order to develop two correction strategies with the aim of minimizing the amount of information the user need to change or delete.

First of all, let us consider the simple position ordering among error correctness evidences, which can be induced from the positions of the errors in the considered Web page. Such an ordering is formalized by means of the following definition.

Definition 5.1: Let $e_1 \equiv (p, w_1, l_1, \sigma_1, C_1)$ and $e_2 \equiv (p, w_2, l_2, \sigma_2, C_2)$ be two correctness error evidences in $\mathbb{E}_N(p)$. Then, $e_1 \preceq e_2$ iff $w_1 \leq w_2$.

We say that e_1 and e_2 are not comparable (w.r.t. \preceq) iff $e_1 \not\preceq e_2$ and $e_2 \not\preceq e_1$.

By exploiting the ordering of Definition 5.1, we are able to prove the following result.

Proposition 5.1: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page, and $e_i = (p, w_i, l_i, \sigma_i, C_i) \in \mathbb{E}_N(p)$, $i = 1, \dots, n$, such that $e_1 \preceq e_2 \preceq \dots \preceq e_n$. The following results hold:

- If $p' \equiv \text{change}(p, w_1, t)$ is a safe repair action, then $p'_{|w_1} \equiv t$ is repaired.
- If $p' \equiv \text{delete}(p, w_1, t)$ is a repair action, then $p'_{|w_1}$ is repaired.

In other words, Proposition 5.1 states that repairing a given correctness error evidence $e_1 \equiv (p, w_1, l_1, \sigma_1, C_1)$ allows us to automatically fix any error which is included in the term $p_{|w_1}$. But, what happens when errors are not comparable w.r.t. \preceq , or we decide to fix an error which is not the smallest in the ordering? Is it still possible to fix more than one error at

a time? In the following, we deepen our analysis about the relation among correctness error evidences in order to answer these questions. Let us start giving an auxiliary definition.

Definition 5.2: Let $e_1 \equiv (p, w_1, l_1, \sigma_1, C_1)$ and $e_2 \equiv (p, w_2, l_2, \sigma_2, C_2)$ be two correctness error evidences in $\mathbb{E}_N(p)$. We say that e_2 *overlaps* e_1 in w (in symbols, $e_2 \overline{\mathcal{X}}_w e_1$), iff (i) $e_1 \preceq e_2$, and (ii) there exists $w \equiv \min(\text{Emb}_{l_1}(p_{|w_1}) \cap \text{Emb}_{l_2}(p_{|w_2}))$, where $\min(X) = w$ s.t. $w \leq w_i$ for all $w_i \in X$. When position w is not relevant or clear from the context, we simply write e_2 *overlaps* e_1 or $e_2 \overline{\mathcal{X}} e_1$.

By notation $e_2 \not\overline{\mathcal{X}} e_1$, we denote that e_2 does not overlap e_1 . Given two correctness errors evidences e_1 and e_2 of a Web page p , we can distinguish three possible scenarios:

- 1) e_1 and e_2 are not comparable w.r.t. \preceq (see Figure 3(a));
- 2) $e_1 \preceq e_2$ and e_2 does not overlap e_1 (see Figure 3(b));
- 3) e_2 overlaps e_1 (see Figure 3(c)).

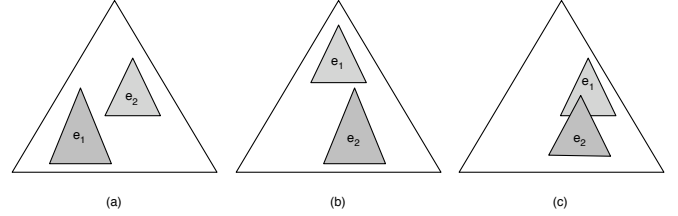


Fig. 3. Distinct kinds of error dependencies.

In Scenario 1, correctness error evidences e_1 and e_2 are completely independent, and hence the repair of one of them does not affect the correction of the other one. This fact together with Proposition 5.1 lead us to an obvious optimization of the correction framework which is formalized by the \mathcal{M} -strategy described in Section VI.

Let us consider now Scenario 2. In this case, still by Proposition 5.1, we are able to automatically fix e_2 by just repairing e_1 . Vice versa, fixing e_2 will not help in fixing e_1 , as stated in the next proposition.

Proposition 5.2: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Let $e_1 \equiv (p, w_1, l_1, \sigma_1, C_1)$ and $e_2 \equiv (p, w_2, l_2, \sigma_2, C_2)$ be two correctness error evidences in $\mathbb{E}_N(p)$ such that $e_1 \preceq e_2$ and $e_2 \not\overline{\mathcal{X}} e_1$. If $p' \equiv \text{action}(p, w_2, t)$, $\text{action} \in \{\text{delete}, \text{change}\}$, then (i) $p'_{|w_2}$ is repaired, (ii) $(p', w_1, l_1, \sigma', C_1) \in \mathbb{E}_N(p')$ for some substitution σ' .

Example 5.1: Consider the Web page $p \equiv f(g(a), h(b))$ and the following correctness error evidences $e_1 \equiv (p, \Lambda, f(X), \{X/h(b)\}, \emptyset)$, $e_2 \equiv (p, 2, h(Y), \{Y/b\}, \emptyset)$. Thus, $e_1 \preceq e_2$ and e_2 does not overlap e_1 . Now observe that we can fix e_2 by either removing subterm $h(b)$ or changing subterm $h(b)$ with a suitable repaired term t . In both cases, such a repair will not fix e_1 .

In Scenario 3, e_1 and e_2 are “connected”, since $e_1 \preceq e_2$ and e_2 overlaps e_1 . Roughly speaking, this fact tells us that the correctness error evidence e_2 is partly “contained” in e_1 and thus fixing e_2 might also yield a fix for e_1 . Anyway, this is not always the case as the next example shows.

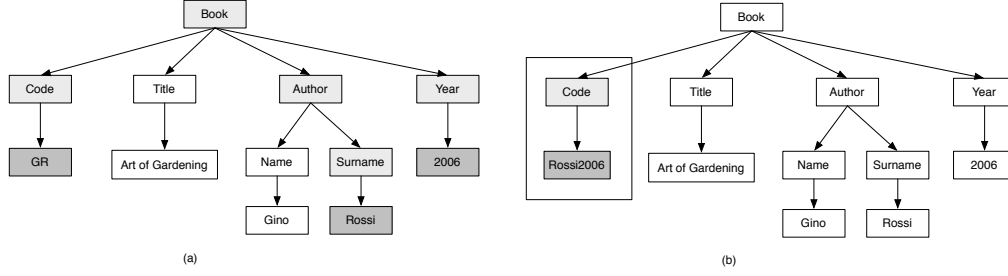


Fig. 2. (a) A Web page p which embeds an error correctness evidence. (b) The Web page p' generated by the execution of the repair action $\text{change}(p, 1, \text{code}(\text{Rossi2006}))$.

Example 5.2: Consider the Web page $p \equiv f(g(a), h(b))$ and the following correctness error evidences $e_1 \equiv (p, \Lambda, f(h(X)), \{X/b\}, \emptyset)$, $e_2 \equiv (p, 2, h(X), \{X/b\}, \emptyset)$. Thus, $e_1 \preceq e_2$ and e_2 overlaps e_1 . We can fix e_2 by changing, for instance, $h(b)$ with $h(a)$. As you can observe such a fix would not repair e_1 automatically, while by removing $h(b)$ or by replacing $h(b)$ with term $l(c)$ we would fix both e_2 and e_1 just by executing a single repair action.

As Example 5.2 illustrates, some conditions are necessary in order to automatically achieve a fix for e_1 by simply correcting e_2 . The next proposition clarifies the ingredients we need to this purpose.

Proposition 5.3: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Let $e_1 \equiv (p, w_1, l_1, \sigma_1, C_1)$ and $e_2 \equiv (p, w_2, l_2, \sigma_2, C_2)$ be two correctness error evidences in $\mathbb{E}_N(p)$ such that $e_1 \preceq e_2$ and e_2 overlaps e_1 in w . The following results hold:

- If $p' \equiv \text{delete}(p, w_2, t)$, then (i) $p'_{|w_2}$ is repaired, (ii) $(p', w_1, l_1, \sigma') \notin \mathbb{E}_N(p')$, with σ' substitution;
- If $p' \equiv \text{change}(p, w_2, t)$ and $l_1|_w \not\triangleq t$, then (i) $p'_{|w_2}$ is repaired, (ii) $(p', w_1, l_1, \sigma') \notin \mathbb{E}_N(p')$, with σ' substitution;
- If $p' \equiv \text{change}(p, w_2, t)$, $l_1|_w \sigma' \triangleq t$ for some substitution σ' , and $C_1(\sigma_1/\sigma')$ does not hold, then (i) $p'_{|w_2}$ is repaired, (ii) $(p', w_1, l_1, \sigma_1/\sigma', C_1) \notin \mathbb{E}_N(p')$.

Roughly speaking, Proposition 5.3 states that

- when a delete action is chosen to fix a correctness error evidence e_2 , which overlaps a smaller (w.r.t. \preceq) correctness error evidence e_1 , such an action will always fix e_1 as well.
- Instead, when a repair action $p' \equiv \text{change}(p, w_2, t)$ is taken in order to fix e_2 , some extra conditions which constrains the term t to be inserted must be fulfilled to automatically fix e_1 . Basically, these conditions establish that either (an instance of) the faulty term l_1 is not recognized in p' or, if such an instance is detected, the associated condition does not hold. This suffices to guarantee that $(p', w_1, l_1, \sigma_1/\sigma', C_1) \notin \mathbb{E}_N(p')$.

VI. CORRECTION STRATEGIES

As we explained in Section IV, a given correctness error evidence e in a Web page p can be fixed by executing a suitable repair action a . By (e, a) we denote a pair containing a repair

action a that fixes e . Moreover, by notation $p' = a(p)$ we intend the execution of the repair action a on the Web page p which returns the Web page p' .

Definition 6.1: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page, and $\mathbb{E}(p) = \{e_1, \dots, e_n\}$ be the set of correctness error evidences of p . A *correction strategy* for p is a sequence

$$\langle (e_1, a_1), \dots, (e_n, a_n) \rangle,$$

where a_1, \dots, a_n are repair actions such that

- 1) $p_0 = p$;
- 2) $p_i = a_i(p_{i-1})$, $0 < i \leq n$.

and p_n is repaired.

Roughly speaking, given a faulty Web page p , a correction strategy for p allows one to fix all the bugs in p by running all the repair actions occurring in the strategy.

As we shown in Section V, fixing a correctness error evidence may automatically repair others bugs. This fact suggests us that a correctness strategy does not necessary contain a pair (e, a) for any correctness error evidence e which appears in a faulty Web page. In the following, we describe two possible correction strategies which exploit the results of Section V: the former aims at minimizing the number of actions which are needed to repair a Web page; instead, the purpose of the latter one is to reduce the amount of information to be removed/changed while correcting a Web site. In both cases, we assume that for any $e \in \mathbb{E}_N(p)$, we have an *error/action* pair (e, a) at hand, and we call the set containing such pairs $\mathbb{EA}(p)$. In other words, we associate a repair action a with every correctness error evidence e .

A. The minimal strategy.

First of all we provide a partial ordering over $\mathbb{EA}(p)$ which is directly induced by the ordering \preceq over correctness error evidences.

Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Given $(e_1, a_1), (e_2, a_2) \in \mathbb{EA}(p)$,

$$(e_1, a_1) \sqsubseteq_T (e_2, a_2) \text{ iff } e_1 \preceq e_2.$$

We say that $(e, a) \in \mathbb{EA}(p)$ is *minimal* w.r.t. \sqsubseteq_T iff there does not exist $(e', a') \in \mathbb{EA}(p)$ such that $(e', a') \sqsubseteq_T (e, a)$.

Now, let us observe the following facts.

- **Fact 1:** By Proposition 5.1, we note that for any $(e, a), (e', a') \in \mathbb{EA}(p)$ such that $(e, a) \sqsubseteq_T (e', a')$, the

execution of the repair action a will fix both e and e' . Therefore fixing a correctness error evidence e which corresponds to a minimal $(e, a) \in \mathbb{EA}(p)$ w.r.t. \sqsubseteq_T will fix all the correctness error evidences e' which are greater than e without running any other repair action.

- **Fact 2:** Given $(e_1, a_1), (e_2, a_2) \in \mathbb{EA}(p)$ both minimal w.r.t. \sqsubseteq_T , e_1 and e_2 are not comparable w.r.t. \preceq .

In the light of these facts, it should be rather clear that it suffices to fix those errors corresponding to minimal error/action pairs to fix the whole Web page.

Definition 6.2 (Minimal strategy): Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page, and $\mathbb{E}(p)$ be the set of correctness error evidences of p . A *minimal* strategy (or \mathcal{M} -strategy) for p is a sequence $\langle (e_1, a_1), \dots, (e_m, a_m) \rangle, (e_i, a_i) \in \mathbb{EA}(p), i = 1, \dots, m$, such that each (e_i, a_i) is minimal w.r.t. \sqsubseteq_T .

Roughly speaking, only the repair actions associated with errors evidences rooted at minimal positions are executed to make the Web page correct.

Proposition 6.1: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Then the \mathcal{M} -strategy for p is a correction strategy for p .

Moreover, since minimal error/action pairs only refer to not comparable (w.r.t. \preceq), and thus independent, correctness error evidences, we may run each repair action of the \mathcal{M} -strategy in parallel, whenever a parallel architecture is available, speeding up the correction process.

Example 6.1: Consider the Web page

$$p \equiv f(g(10), h(d), 20)$$

together with the following error/action pairs:

$$\begin{aligned} & ((p, \Lambda, f(g(X), 20), \{X/10\}, \{X < 20\}), \\ & \text{change}(p, \Lambda, f(g(20), 10)) \end{aligned}$$

and $((p, 2, h(Y), \{Y/d\}, \emptyset), \text{delete}(p, 2, h(d)))$. In this case, the \mathcal{M} -strategy corresponds to the unary sequence

$$\begin{aligned} & \langle ((p, \Lambda, f(g(X), 20), \{X/10\}, \{X < 20\}), \\ & \text{change}(p, \Lambda, f(g(20), 10))) \rangle. \end{aligned}$$

Taking into consideration only minimal error/action pairs allows us to define a correction strategy which minimizes the number of actions we need to perform for repairing a Web page as the next result states.

Proposition 6.2: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page and T be the \mathcal{M} -strategy for p . Then, for every correction strategy S for p , $\text{length}(T) \leq \text{length}(S)$, where $\text{length}(\cdot)$ computes the number of error/action pairs of a given correction strategy.

B. Minimal non-overlapping strategy.

The \mathcal{M} -strategy typically forces the user to modify/introduce a lot of information in a Web page p , even if only minor changes are required to fix p . Let us see an example.

Example 6.2: Let us consider the Web page

$$p \equiv f(g(a), k(m(c)), h(a))$$

and the set $\mathbb{E}_N(p) = \{(p, \Lambda, f(g(X), h(Y)), \{X/a, Y/a\}, \{X=Y\}), (p, 1, g(a), \varepsilon, \emptyset)\}$. The \mathcal{M} -strategy would only fix the “minimal” error at the root position. This fact might force

the user to provide a quite big amount of information in case a change action is taken, even when a close variant of p was enough to fix the bug.

For instance, if the chosen change action was $\text{change}(p, \Lambda, f(g(b), k(m(c)), h(a)))$, the user should re-enter the whole Web page p with just a small change at position 1.1.

Instead, if we repaired $(p, 1, g(a), \varepsilon, \emptyset)$ by means of the following action $\text{change}(p, 1, g(b))$, the user would correct both errors by introducing a smaller amount of information. The idea behind the minimal non-overlapping strategy is thus to “push” the corrections towards the leaves of the Web page as much as possible and to automatically propagate them up to the root position.

Obviously, given two error evidences e and e' such that $e' \preceq e$, correcting e does not guarantee to automatically fix e' (see, for instance, Example 5.1). Indeed, by Proposition 5.2, whenever an error evidence e' does not overlap a given error evidence e , there is no possibility to automatically spread a correction for e up to e' . On the other hand, under suitable conditions, overlapping error evidences allow one to infer a repair on e' by just fixing e (see Proposition 5.3).

Therefore, the strategy works as follows. First of all, given a Web page p , we partition $\mathbb{E}_N(p)$ into the two following sets:

- $\text{NOVL}(p) = \{e \in \mathbb{E}_N(p) \mid \nexists e', e' \sqsupset e\}$
- $\text{OVL}(p) = \mathbb{E}_N(p) \setminus \text{NOVL}(p)$.

Clearly, $\mathbb{E}_N(p) = \text{NOVL}(p) \cup \text{OVL}(p)$. We call error evidences in $\text{NOVL}(p)$ (resp., in $\text{OVL}(p)$) *non-overlapping* (resp., *overlapping*) error evidences. Note that a non-overlapping error evidence e cannot be automatically fixed by executing a repair action on an error evidence e' such that $e \preceq e'$, since correction effects cannot be propagated up. However, this is the case of the overlapping error evidences which may be implicitly affected by other repairs. Indeed, the following facts hold.

Fact 1. Given an overlapping error evidence e , there must exist a non-overlapping error evidence e' such that $e \preceq e'$.

Fact 2. Let e, e_0, e_1, \dots, e_n , $n \geq 0$, be correctness error evidences. If e is an overlapping error evidence such that e_0 overlaps e and $e \preceq e_n \preceq e_{n-1}, \dots, \preceq e_0$, then e_i overlaps e , $i = 1, \dots, n$.

These facts along with Proposition 5.1 hint us that fixing only non-overlapping error evidences suffices to get a repaired Web page; since (i) all the error evidences which are greater w.r.t. \preceq than the considered non-overlapping error evidences will be repaired as stated by Proposition 5.1; (ii) for each overlapping error evidence e , there is always $e' \in \text{NOVL}(p)$ which overlaps e , hence repairing e' will also fix e whenever the following *safeness* property is fulfilled:

Definition 6.3: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Let $e \equiv (p, w, l, \sigma, C) \in \text{NOVL}(p)$, and $(e, \text{change}(p, w, t)) \in \mathbb{EA}(p)$. Then, the *safeness* property for $(e, \text{change}(p, w, t)) \in \mathbb{EA}(p)$ states that for each

$e' \equiv (p, w', l', \sigma', C') \in \text{OVL}(p)$ such that $e' \preceq e$, one of the following conditions must hold:

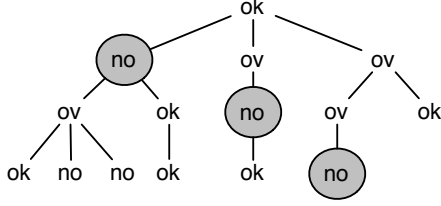


Fig. 4. The \mathcal{MNO} -strategy

- $l'_w \not\leq t$, or
- $l'_w \sigma' \leq t$ for some substitution σ' , and $C'(\sigma/\sigma')$ does not hold.

Note that the safeness property directly comes from Proposition 5.3 which guarantees the automatic propagation of the repairs. Moreover, observe that such property only affects change actions, since delete actions always enable the correction propagation.

Let us consider the following partial ordering over the elements in $\text{NOVL}(p)$. Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Given $(e_1, a_1), (e_2, a_2) \in \text{NOVL}(p)$,

$$(e_1, a_1) \sqsubseteq_B (e_2, a_2) \text{ iff } e_1 \preceq e_2.$$

We say that $(e, a) \in \text{NOVL}(p)$ is *minimal* w.r.t. \sqsubseteq_B iff there does not exist $(e', a') \in \text{NOVL}(p)$ such that $(e', a') \sqsubseteq_B (e, a)$.

Now, we are ready to provide the minimal non-overlapping strategy.

Definition 6.4 (Minimal non-overlapping strategy): Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page, and $\text{NOVL}(p)$ be the set of non-overlapping correctness error evidences of p . A *minimal non-overlapping strategy* (or \mathcal{MNO} -strategy) for p is a sequence $\langle (e_1, a_1), \dots, (e_m, a_m) \rangle$, $(e_i, a_i) \in \mathbb{EA}(p)$, $i = 1, \dots, m$, such that

- each (e_i, a_i) is minimal w.r.t. \sqsubseteq_B and $e_i \in \text{NOVL}(p)$;
- if a_i is a change action, then the safeness property for (e_i, a_i) must hold.

Proposition 6.3: Let $p \in \tau(\text{Text} \cup \text{Tag})$ be a Web page. Then the \mathcal{MNO} -strategy for p is a correction strategy for p . In Figure 4, we show how the \mathcal{MNO} -strategy works. For the sake of simplicity we just label each node of the given Web page with

- *ok*, if no error evidence is rooted at the considered node;
- *ov*, if an overlapping error evidence is rooted at the considered node;
- *no*, if a non-overlapping error evidence is rooted at the considered node.

The Web page contains 9 errors, but we just need to fix 3 errors to get a repaired Web page. Precisely, these errors correspond to the minimal non-overlapping error evidences occurring in the Web page.

Example 6.3: Consider the Web page

$$p \equiv f(g_1(h_1(a_1, a_2), h_2(b_1, b_2)), g_2(h_3(c_1), h_4))$$

and

$$\mathbb{E}_N(p) = \{(p, 1, g_1, \varepsilon, \emptyset), (p, 1.1, h_1(a_2), \varepsilon, \emptyset), (p, 1.1.2, a_2, \varepsilon, \emptyset), (p, 2, g_2(h_3(X)), \{X/c_1\}, \emptyset), (p, 2.1, h_3(c_1), \varepsilon, \emptyset), (p, 2.1.1, c_1, \varepsilon, \emptyset)\}$$

Hence,

$$\text{NOVL}(p) = \{(p, 1, g_1, \varepsilon, \emptyset), (p, 1.1.2, a_2, \varepsilon, \emptyset), (p, 2.1.1, c_1, \varepsilon, \emptyset)\}$$

$$\text{OVL}(p) = \{(p, 1.1, h_1(a_2), \varepsilon, \emptyset), (p, 2, g_2(h_3(X)), \{X/c_1\}, \emptyset), (p, 2.1, h_3(c_1), \varepsilon, \emptyset)\}$$

The \mathcal{MNO} -strategy only corrects minimal non-overlapping error evidences. A possible \mathcal{MNO} -strategy for p might be:

$$\langle ((p, 1, g_1, \varepsilon, \emptyset), \mathbf{delete}(p, 1, g_1(h_1(a_1, a_2), h_2(b_1, b_2))), ((p, 2.1.1, c_1, \varepsilon, \emptyset), \mathbf{change}(p, 2.1.1, c_4)) \rangle.$$

Note that the safeness property for

$$((p, 2.1.1, c_1, \varepsilon, \emptyset), \mathbf{change}(p, 2.1.1, c_4))$$

is fulfilled. The execution of the correction strategy yields the following repaired Web page $f(g_2(h_3(c_4), h_4))$.

Finally, observe that we needed to fix only two errors out of six, and just minor fixes were necessary to make the original Web page correct.

VII. CONCLUSIONS

Maintaining contents of Web sites is an open and urgent problem since outdated, incorrect/forbidden information is becoming very frequent in the World Wide Web. In this paper, we presented a significant extension of our previous work about Web site correction [9], which improved several aspects of the repair methodology: (i) we provided a detailed analysis of errors which clarified the relation among correctness errors in a Web site; (ii) by exploiting the results of the analysis, we formulated two correction strategies which reduce the number of repair actions and the amount of information needed to fix a given Web site; (iii) the considered correction strategies increase the level of automation of the repair method, since the user just has to fix a small number of correctness errors to make the whole Web site correct. As a further work, we intend to integrate both the \mathcal{M} -strategy and the \mathcal{MNO} -strategy into our repair system GVERDI [10], which is publicly available at www.dsic.upv.es/users/elpl/GVerdiR/.

REFERENCES

- [1] C. Nentwich, W. Emmerich, and A. Finkelstein, "Consistency Management with Repair Actions," in *Proc. of the 25th International Conference on Software Engineering (ICSE'03)*. IEEE Computer Society, 2003.
- [2] L. Capra, W. Emmerich, A. Finkelstein, and C. Nentwich, "XLINKIT: a Consistency Checking and Smart Link Generation Service," *ACM Transactions on Internet Technology*, vol. 2(2), pp. 151–185, 2002.
- [3] J. Scheffczyk, U. M. B. P. Rödiger, and L. Schmitz, "S-dags: Towards efficient document repair generation," in *Proc. 2nd Int. Conf. on Computing, Communications and Control Technologies*, vol. 2, 2004, pp. 308–313.
- [4] J. Scheffczyk, P. Rödiger, U. M. Borghoff, and L. Schmitz, "Managing inconsistent repositories via prioritized repairs," in *Proc. of the 2004 ACM Symposium on Document Engineering (DocEng '04)*. ACM Press, 2004, pp. 137–146.

- [5] J. Scheffczyk, U. M. Borghoff, P. Rödiger, and L. Schmitz, "Consistent document engineering: formalizing type-safe consistency rules for heterogeneous repositories," in *Proc. of the 2003 ACM Symposium on Document Engineering (DocEng '03)*. ACM Press, 2003, pp. 140–149.
- [6] L. Bertossi and J. Pinto, "Specifying Active Rules for Database Maintenance," in *Transactions and Database Dynamics, 8th Int'l Workshop on Foundations of Models and Languages for Data and Objects*, ser. Lecture Notes in Computer Science, G. Saake, K. Schwarz, and C. Türker, Eds., vol. 1773. Springer, 1999, pp. 112–129.
- [7] E. Mayol and E. Teniente, "A Survey of Current Methods for Integrity Constraint Maintenance and View Updating," in *Proc. of Advances in Conceptual Modeling: ER '99*, ser. Lecture Notes in Computer Science, vol. 1727. Springer, 1999, pp. 62–73.
- [8] M. Alpuente, D. Ballis, and M. Falaschi, "Automated Verification of Web Sites Using Partial Rewriting," *Software Tools for Technology Transfer*, 2006, to appear.
- [9] M. Alpuente, D. Ballis, M. Falaschi, and D. Romero, "A Semi-automatic Methodology for Repairing Faulty Web Sites," in *Proc. of the 4th IEEE Int'l Conference on Software Engineering and Formal Methods (SEFM'06)*. IEEE Computer Society Press, 2006, to appear.
- [10] D. Ballis and J. G. Vivó, "A Rule-based System for Web Site Verification," in *Proc. of 1st Int'l Workshop on Automated Specification and Verification of Web Sites (WWV'05)*, vol. 157(2). ENTCS, Elsevier, 2005.
- [11] N. Dershowitz and D. Plaisted., "Rewriting," *Handbook of Automated Reasoning*, vol. 1, pp. 535–610, 2001.
- [12] F. Baader and T. Nipkow, *Term Rewriting and All That*. Cambridge University Press, 1998.
- [13] J. Klop, "Term Rewriting Systems," in *Handbook of Logic in Computer Science*, S. Abramsky, D. Gabbay, and T. Maibaum, Eds. Oxford University Press, 1992, vol. I, pp. 1–112.
- [14] F. Bry and S. Schaffert, "The XML Query Language Xcerpt: Design Principles, Examples, and Semantics," Tech. Rep., 2002, available at: <http://www.xcerpt.org>.
- [15] —, "Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification," in *Proc. of the Int'l Conference on Logic Programming (ICLP'02)*, ser. Lecture Notes in Computer Science, vol. 2401. Springer-Verlag, 2002.
- [16] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web. From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.
- [17] M. F. Fernandez and D. Suciu, "Optimizing Regular Path Expressions Using Graph Schemas," in *Proc. of Int'l Conference on Data Engineering (ICDE'98)*, 1998, pp. 14–23.
- [18] E. Bertino, M. Mesiti, and G. Guerrin, "A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and its Applications," *Information Systems*, vol. 29, no. 1, pp. 23–46, 2004.
- [19] M. Bezem, *TeReSe, Term Rewriting Systems*. Cambridge University Press, 2003, ch. Mathematical background (Appendix A).
- [20] M. Leuschel, "Homeomorphic Embedding for Online Termination of Symbolic Methods," in *The Essence of Computation*, ser. Lecture Notes in Computer Science, T. Æ. Mogensen, D. A. Schmidt, and I. H. Sudborough, Eds., vol. 2566. Springer, 2002, pp. 379–403.