DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSIDAD POLITÉCNICA DE VALENCIA

P.O. Box: 22012 E-46071 Valencia (SPAIN)

Informe Técnico / Technical Report

Ref. No.: DSIC-II/19/07 **Pages:** 16
Title: Abstract Web Site Verification in WebVerdi-M
Author(s): M. Alpuente, D. Ballis, M. Falaschi, P. Ojeda and D. Romero
Date: 26 Movember 2007
Keywords: web verification, abstract interpretation, maude

V^{wo} B^{wo}
Leader of research Group Author(s)

Abstract Web Site Verification in WebVerdi-M *

M. Alpuente P. Ojeda D. Romero
Universidad Politécnica de Valencia
Camino de Vera s/n, Apdo. 22012,
46071 Valencia, Spain.
{alpuente,pojeda,dromero}@dsic.upv.es

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

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

Abstract

In this paper, we present an abstract framework for Web site verification which improves the performance of a previous, rewriting-based Web verification methodology. The approximated framework is formalized as a source-to-source transformation for compressing (and abstracting) Web sites which is parametric w.r.t. the chosen abstraction. This transformation significantly reduces the size of the Web documents by dropping or merging contents that do not influence the properties to be checked. This allows us to reuse all verification facilities of the previous system WebVerdi-M *to efficiently analyze Web sites. In order to ensure that the verified properties are not affected by the abstraction, we develop a methodology which derives an abstraction of Web sites from their Web specification. We believe that this methodology can be successfully embedded into several frameworks. In particular we have developed a prototypical implementation which shows a huge speedup w.r.t. a previous methodology which did not use this transformation.*

*This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN2004-7943-C04-01 and TIN2007-68093-C02-02, the Generalitat Valenciana under grant GV06/285, and Integrated Action Hispano-Alemana HA2006-0007, and progetto FIRB, Internazionalizzazione 2004, n. RBIN04M8S8. Pedro Ojeda is also supported by the Generalitat Valenciana under FPI grant BFPI/2007/076.

1 Introduction

Despite the exponential WWW growth and the success of the Semantic Web, there is limited support today to specifying, verifying and repairing Web sites at a semantic level. Some sophisticated Web site management tools have been recently proposed that provide helpful facilities, including active rules that automatically fire repair actions throughout the rich navigational structure of Web sites. Unfortunately, these tools are mostly focused towards syntactic checking and Web sites restructuring (see [4, 5, 6] for a wider discussion).

A rewriting-based approach to Web-site verification and repair was developed in [4, 8]. The methodology applies to static-HTML/XML Web sites and can discover flaws in Web sites that are not addressed by classical tools [3, 20, 29] as these mainly focus on modeling navigational aspects and user interaction; see [2] for a comparison of different modeling methods for Web site verification and testing. In a nutshell, our framework comes with a Web specification language for defining correctness and completeness conditions on Web sites. Then, a rewriting-based verification technique is applied to recognize forbidden/incorrect patterns and incomplete/missing Web pages. This is done by means of a novel technique, called *partial rewriting*, in which the traditional pattern matching mechanism is replaced by a suitable technique based on an *homeomorphic embedding* relation for recognizing patterns inside semistructured documents.

The verification methodology of [4] is implemented in the prototype WebVerdi-M (Web Verification and Rewrit-

ing for Debugging Internet sites with Maude) [6], whose Web verification engine is written in Maude [14]. For correctness checking, it shows impressive performance thanks to the Associativity-Commutativity (AC) pattern matching and metalevel features supported by Maude (for instance, verifying correctness over a 10Mb XML document with 302000 nodes takes less than 13 seconds). In fact, both resource allocation and elapsed time scale linearly. Unfortunately, for the verification of completeness, a (finite) fixpoint computation is typically needed which leads to unsatisfactory performance, and the verification tool is only able to efficiently process XML documents whose size is not bigger than 1Mb.

In this paper, we develop an approach to Web site verification which makes use of an approximation technique based on abstract interpretation [16, 17] that greatly improves on previous performance. We provide an abstraction scheme where both the Web site and the Web specification rules are translated into constructions of the original source languages. We also ascertain the conditions which ensure the correctness of the approximation, so that the resulting abstract rewriting engine safely supports accurate Web site verification. Since our framework is parametric w.r.t. the considered abstraction, we precisely characterize the conditions which allow to ensure the correctness of the abstraction, which is implemented by a source-to-source transformation of concrete Web sites and Web specifications into abstract ones. Thanks to this source-to-source approximation scheme, all facilities supported by our previous verification system are straightforwardly adapted and reused with very little effort. It is also worth mentioning that our abstract verification methodology can be seen as a step forward towards more sophisticated management of dynamic components of Web sites, which can generate a potentially infinite number of Web pages and thus cannot be handled by the more standard methodology in [4].

Related Work In the literature, abstract interpretation frameworks have been scarcely applied to analyse Web sites. Actually, we have found very few works addressing this issue, and all of them focus on the dynamic aspects of the distributed system underlying the Web site. For instance, in [26] an abstract approach is developed which allows one to analyse the communication protocols of a particular distributed system with the aim of enforcing a correct global behavior of the system. [23] uses abstract interpretation for secret property verification: the methodology applies Input/Output abstract set descriptions to finite state machines in order to validate cryptographic protocols implementing secure Web transactions.

To the best of our knowledge, this work develops the first methodology based on abstract interpretation techniques which is general enough to support the verification of static as well as dynamic aspects of Web sites. Our inspiration comes from the area of approximating (XML) query answering [12, 30], where XML queries are executed on compressed versions of XML data (i.e., document synopses) in order to obtain fast, albeit approximate, answers. Roughly speaking, document synopses represent *abstractions* of the original data on which abstract computations (i.e., queries) are performed.

In our methodology, both the XML documents (Web pages) and the constraints (Web specification rules) are approximated via an abstraction function. Then, the verification process is carried out using the abstract descriptions of the considered XML data. This approach results in a powerful abstract verification methodology which pays off in practice. In fact, the preliminary experiments reported in Section 5 are quite encouraging and verify the viability of the proposed techniques in terms of both accuracy and processing time on large datasets.

Plan of the Paper The paper is organized as follows. Section 2 recalls some standard notions, and introduces Web site descriptions. In Section 3, we briefly recall the Web verification methodology of [4]. Section 4 formalizes the notion of abstract Web specification and introduces the key idea behind our method, which is given by an approximation algorithm which derives an abstraction of Web sites from their Web specifications and drastically reduces the size of the Web documents. Then we formalize the abstract partial rewriting and illustrate how the original rewriting-based Web-site verification technique of [4] can be safely approximated by the abstract framework. Experiments with a prototypical implementation of our method are described in Section 5. Finally, Section 6 concludes.

2 Preliminaries

We call a finite set of symbols *alphabet*. By $\mathcal{V}$ we denote a countably infinite set of variables and Σ denotes a set of *function symbols* (also called *operators*), or *signature*. We consider varyadic signatures as in [18] (i.e., signatures in which symbols do not have a fixed arity).

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. Given $S \subseteq \Sigma \cup \mathcal{V}$, $O_S(t)$ denotes the set of positions of a term t that are rooted by symbols in S. $t_{|u}$ 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 variable occurs in t. $\tau(\Sigma, \mathcal{V})$ and $\tau(\Sigma)$ denote the *non-ground term algebra* built on $\Sigma \cup \mathcal{V}$ and the *term algebra* built on Σ, respectively.

Syntactic equality between objects is represented by $\equiv$. Given a set S, sequences of elements of S are build by using constructors $\epsilon :: S^*$ (the empty sequence) and $. :: S \times S^* \rightarrow S^*$.

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

2.1 Web Site Description

Let us consider two alphabets T and $\mathcal{T}ag$. We denote the set T^* by $\mathcal{T}ext$. An object $t \in \mathcal{T}ag$ is called *tag* element, while an element $w \in \mathcal{T}ext$ is called *text* element. Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of the term algebra $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$ [4] as shown in Figure 1.

```
<people>                    people(
 <person>                    person(
   <id>per0</id>                id(per0),
   <name>Conte</name>           name(Conte)
 </person>                    )
</people>                   )
```

Figure 1. A Web page and its corresponding encoding as a ground term

Note that XML/XHTML tag attributes can be considered as common tagged elements, and hence translated in the same way. 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 $s \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ 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 describe a Web site, we use the formulation given in [28]. We use an alphabet $\mathcal{P}$ to give names to Web pages as well as to express the different transitions between Web pages.

Definition 2.1 (immediate successors) *The immediate successors relation for a given Web page p is defined by* $\rightarrow_p = \{(p, p') \subseteq \mathcal{P} \times \mathcal{P} \mid p'$ *is directly accessible from* $p\}$.

Definition 2.1 establishes a relationship between the page p and its immediate successors (i.e., the pages $p_1, \ldots, p_n$ which are commonly pointed to from p by means of hyperlinks).

The pair $(\mathcal{P}, \rightarrow_{\mathcal{P}})$, where $\rightarrow_{\mathcal{P}} = \bigcup_{p \in \mathcal{P}} \rightarrow_p$, is an *Abstract Reduction System* (see ARS [7], Chapter 2). We will use the associated computational relations $\rightarrow_{\mathcal{P}}$, $\rightarrow^+_{\mathcal{P}}$, etc., to describe the dynamic behavior of a Web site. In this context, the reachability of a given Web page p' from another page p can be expressed as $p \rightarrow^*_{\mathcal{P}} p'$.

Definition 2.2 (Web site) *A Web site is defined as a set of reachable Web pages from an initial Web page, and is denoted by*

$$W = \{p_1, \ldots, p_n\},\ \textit{s.t.}\quad \exists i, 1 \leq i \leq n,\ \forall j, 1 \leq j \leq n,\ p_i \rightarrow^*_W p_j$$

Definition 2.2 formalizes the idea that a Web site has an initial Web page which allows one to visit the whole Web site. Note that there may exist several initial Web pages of a given Web site.

Example 2.3 *The algebraic description of a simple Web site modeling an on-line auction system is shown in Figure 2. It contains information regarding open and closed auctions, auctioned items, and registered users.*

3 Rewriting-based Web Verification

In this section, we briefly recall the formal verification methodology proposed in [4], which allows us to detect forbidden/erroneous contents as well as missing information in a Web site. This methodology is able to recognize and exactly locate the source of a possible discrepancy between the Web site and the properties required in the Web specification.

3.1 The Web specification language

A Web specification is a triple (R, I_N, I_M), where R, I_N, and I_M are a finite set of rules. The set R contains the definition of some auxiliary functions which the user would like to provide, such as string processing, arithmetic, boolean operators, etc. R is formalized as a term rewriting system, which is handled by standard rewriting [19, 24, 31]. The rewriting mechanism executes function calls by simply reducing them to their irreducible form (that is, a term that cannot be rewritten further). The set I_N describes constraints for detecting erroneous Web

Web site $W = \{p_1, p_2, p_3, p_4, p_5\}$, where

p_1) list-items(
item(id(ite0),name(racket),state(sold),
description(Wilson tennis racket),
incategories(category(cat1))),
item(id(ite1),name(shirt),state(available),
description(men's t-shirts),
incategories(category(cat1),
category(cat2))),
item(id(ite2),name(shoes),state(sold),
description(women's shoes),
incategories(category(cat0),
category(cat2))))

p_2) list-categories(
pack(category(cat0)),
unit(category(cat1),
category(cat2)))

p_3) people(
person(id(per0),
name(Eliyahu),
email(eliyahu@cas.cz)),
person(id(per1),
name(Melski),
email(melski@cabofalso.com)),
person(id(per2),
name(Conte),
email(conte@forth.gr)),
person(id(per3)))

p_4) open-auctions(
open-auction(id(open-auction0),
item(ite0),
initial(48.51), reserve(77.5),
bidder(person(per0)),
seller(person(per1))))

p_5) closed-auctions(
closed-auction(
seller(person(per1)),
buyer(person(per0)),
item(ite0), price(77.5)),
closed-auction(
seller(person(per5)),
buyer(person(per0)),
item(ite2), price(45.2)))

Web specification (I_N, I_M, R), where $I_N = \{r_1, r_2\}$ and $I_M = \{r_3, r_4, r_5\}$,
r_1) open-auction(initial(X),reserve(Y)) $\rightharpoonup$ error : X > Y
r_2) person(email(X)) $\rightharpoonup$ error : X not in [:Text:]$^+$ @ [:Text:]$^+$
r_3) open-auction(seller(person(X))) $\rightharpoonup$ ♯person(♯id(X)) $\langle E \rangle$
r_4) list-items(item(incategories(X,Y))) $\rightharpoonup$ ♯list-categories(pack(X),unit(Y)) $\langle E \rangle$
r_5) people(person(id(X))) $\rightharpoonup$ ♯people(♯person(♯id(X)),name) $\langle A \rangle$
r_6) list-item(item(id(X),state(sold)) $\rightharpoonup$ ♯closed-auction(item(X)) $\langle E \rangle$

Figure 2. Web site and Web specification for an on-line auction system.

pages (*correctNess rules*). A correctness rule has the following syntax:

$$l \rightharpoonup error \mid C$$

where l is a term, $error$ is a reserved constant, and C is a (possibly empty) finite sequence which could contain membership tests of the form $X \in rexp$ w.r.t. a given regular language $rexp$;[1] and/or equations/inequalities over terms.

When C is empty, we simply write $l \rightharpoonup error$. Informally, the meaning of a correctness rule is the following: whenever (i) a "piece" of a given Web page $p \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ can be "recognized" to be an instance $l\sigma$ of l, and (ii) the corresponding instantiated condition $C\sigma$ holds in R, then Web page p is marked as an incorrect page with erroneous content $l\sigma$.

[1] Regular languages are represented by means of the usual Unix-like regular expression syntax; see e.g. rule r_1 of Example 3.1.

The third set of rules I_M specifies some properties for discovering incomplete/missing Web pages (*coMpleteness rules*). A completeness rule is defined as

$$l \rightharpoonup r \ \langle \mathtt{q} \rangle$$

where l and 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, as explained below. Right-hand sides r of completeness rules can contain functions, which are defined in R. Moreover, some symbols in the right-hand sides of the rules may be marked by means of the symbol ♯. Marking information in a given rule r is used to select the subset of the Web site where the condition formalized by the rule must be checked. Intuitively, the interpretation of a universal rule $l \rightharpoonup r \ \langle \mathtt{A} \rangle$ (respectively, an existential rule $l \rightharpoonup r \ \langle \mathtt{E} \rangle$) w.r.t. a Web

site W is as follows: if (an instance of) l is "recognized" in W, (an instance of) the irreducible form of r must also be "recognized" in *all* (respectively, *some*) of the Web pages that embed (an instance of) the marked structure of r.

Example 3.1 *A Web specification for the on-line auction system of Example 2.3 is shown in Figure 2. The first two rules are correctness rules while the other three rules are completeness rules. The first rule requires that, in a open auction, the reserve price (or the lower price at which a seller is willing to sell an item) is greater than the initial one. The second rule requires email addresses to contain the symbol @, by checking whether they belongs to the regular language given by the expression* [:Text:]$^+$ @ [:Text:]$^+$. *The third (existential) rule formalizes the following property: if there is a Web page with the information about the seller of an open-auction, then such a seller must be registered. The fourth rule also states an existential property: if there is an auctioned item that is listed in two or more categories, then at least one of these categories must be "unit" and the other one "pack". The fifth rule formalizes the following universal property: for each client registered in the Web site, there must exist a page containing his name. Finally, the last rule states that, for every item that is sold, a closed auction associated to the item must exist.*

3.2 Homeomorphic embedding 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 homeomorphic embedding, which allow us to verify whether a given Web page template is somehow "enclosed" within another one. Roughly speaking, we consider a simple embedding relation which closely resembles the notion of *simulation* [22]. This relation has been widely used in a number of works about querying, transformation, and verification of semistructured data (cf. [1, 9, 10, 11, 21]).

We give a definition of homeomorphic embedding, $\trianglelefteq$, which is an adaptation of the one proposed in [27], where (i) a simpler treatment of the variables is considered, (ii) function symbols with variable arities are allowed, (iii) the relative positions of the arguments of terms are ignored (i.e. $f(a, b)$ is not distinguishable from $f(b, a)$), and (iv) we ignore the usual *diving* rule[2] [27].

[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.

Definition 3.2 (homeomorphic embedding) *The* homeomorphic embedding *relation*

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

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

1. $X \trianglelefteq t$, *for all* $X \in \mathcal{V}$ *and* $t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$.
2. $f(t_1, \ldots, t_m) \trianglelefteq g(s_1, \ldots, s_n)$ *iff* $f \equiv g$ *and* $t_i \trianglelefteq s_{\pi(i)}$, *for* $i = 1, \ldots, m$, *and some total, injective function* $\pi : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$.

Whenever $s \trianglelefteq t$, we say that t *embeds* s (or s *is embedded* or "*recognized*" in t).

The intuition behind the above definition is that $s \trianglelefteq t$ iff s can be obtained from t by striking out certain parts. In other words, the structure of the template s appears within the specific Web data term t.

Let us illustrate Definition 3.2 by means of a rather intuitive example.

Example 3.3 *Consider the following Web page templates (called s_1 and s_2, respectively):*

$$people(person(email(X), id(Y)))$$

$$people(person(id(per0), \\ name(Eliyahu), \\ email(eliyahu@cas.cz)))$$

Note that the structure of s_1 can be recognized inside the structure of s_2 hence $s_1 \trianglelefteq s_2$, while $s_2 \not\trianglelefteq s_1$. Also note that the order of the arguments is irrelevant.

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

3.3 Web verification methodology

Roughly speaking, our verification methodology works as follows. First, by using the homeomorphic embedding relation of Definition 3.2, we check whether the left-hand side l of some Web specification rule is embedded into a given page p of the considered Web site. When the embedding test $l \trianglelefteq p$ succeeds, by extending the proof, we construct the biggest substitution[3] σ for the variables in $Var(l)$, such that $l\sigma \trianglelefteq p$. Then, depending on the nature of the Web specification rule (correction or completeness rule), we proceed as follows:

Correction rule. We evaluate the condition of the rule (instantiated by σ); a correctness error is signalled in the case when the error condition is fulfilled. Formally, the definition of a correctness error is as follow.

Definition 3.4 (correctness error) *Let W be a Web site and (I_N, I_M, R) be a Web specification. Then, the quadruple (p, w, l, σ) is a correctness error iff $p \in W$, $w \in O_{\mathcal{T}ag}(p)$, and $l\sigma$ is an instance of the left-hand side l of a correctness rule belonging to I_N such that $l\sigma \trianglelefteq p_{|w}$.*

Completeness rule. We apply a partial rewriting step to p, that is, we replace the Web page p embedding $l\sigma$ with the instantiated right-hand side $r\sigma$, in symbols $p \rightharpoonup r\sigma$. Each term $r\sigma$ generated by partial rewriting is considered a *requirement*.
Then, by a new homeomorphic embedding test, we check whether the given requirement $r\sigma$ is recognized within some page of the considered Web site. Otherwise, a completeness error (*missing Web page*, *universal error* or *existential error*) is signalled. From the requirements computed so far, new requirements are generated and subsequently checked by applying further partial rewriting steps. A fixpoint computation is performed which may involve the execution of several completeness rules. The formal definition of completeness errors is below.

Definition 3.5 (missing Web page) *Let W be a Web site and (I_N, I_M, R) be a Web specification. Then the pair (r, W) is a missing Web page error if there exists $p \in W$ s.t. $p \rightharpoonup^+_{I_M} r$ and $r \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ does not belong to W.*

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

Definition 3.6 *Let P be a set of terms in $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$ and $r \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$. We say that P is universally (resp. existentially) complete w.r.t. r iff for each $p \in P$ (resp. for some $p \in P$), there exists $w \in O_{\mathcal{T}ag}(p)$ s.t. $r \trianglelefteq p_{|w}$.*

Definition 3.7 Universal and Existential completeness error. *Let W be a Web site and (I_N, I_M, R) be a Web specification. Then the triple $(r, \{p_1, \ldots, p_n\}, q)$, with $q = A$ (resp. $q = E$) is a universal (resp. existential) completeness error, if there exists $p \in W$ s.t. $p \rightharpoonup^+_{I_M} r$ and $\{p_1, \ldots, p_n\}$ is not universally (resp. existentially) complete w.r.t. r, $p_i \in W, i = 1, \ldots, n$.*

Example 3.8 *Consider again the Web specification and Web site W of Example 3.1. Then, the set of completeness requirements computed in the fixpoint by our verification methodology is*

$\mathcal{S}_{req} = \{rq_1, rq_2, rq_3, rq_4, rq_5, rq_6, rq_7, rq_8, rq_9, rq_{10}\}$, *where*

$rq_1)\langle \sharp person(\sharp id(per1)), E\rangle$,
$rq_2)\langle \sharp person(\sharp id(per5)), E\rangle$,
$rq_3)\langle \sharp list\text{-}categories(pack(category(cat1)), unit(category(cat2))), E\rangle$,
$rq_4)\langle \sharp list\text{-}categories(pack(category(cat0)), unit(category(cat2))), E\rangle$,
$rq_5)\langle \sharp closed\text{-}auction(item(ite0)), E\rangle$,
$rq_6)\langle \sharp closed\text{-}auction(item(ite2)), E\rangle$,
$rq_7)\langle \sharp people(\sharp person(\sharp id(per0), name)), A\rangle$,
$rq_8)\langle \sharp people(\sharp person(\sharp id(per1), name)), A\rangle$,
$rq_9)\langle \sharp people(\sharp person(\sharp id(per2), name)), A\rangle$,
$rq_{10})\langle \sharp people(\sharp person(\sharp id(per3), name)), A\rangle$

Now, by a new homeomorphic embedding test, we check whether each requirement in the set $\mathcal{S}_{req}$ is recognized inside some page of the Web site W. This produces the following set of completeness errors

$\mathbb{E}_M = \{e_1, e_2, e_3\}$, *where*
$e_1)\langle rq_2, \{p_3\}, E\rangle$
$e_2)\langle rq_3, \{p_2\}, E\rangle$
$e_3)\langle rq_{10}, \{p_3\}, A\rangle$

Note that no correctness error exists in W w.r.t. the considered Web specification.

[3]The substitution σ is easily obtained by composing the bindings X/t, which can be recursively gathered during the *homeomorphic embedding* test $X \trianglelefteq t$, for $X \in l$ and $t \in p$.

4 Abstract Web site verification

Let us first introduce the notion of an abstract domain. Our first definition is motivated by the fact that we want to formalize the abstraction as a source–to–source transformation which translates Web documents and Web specification rules into constructions of the very same languages, hence our concrete and abstract domains do coincide.

Definition 4.1 abstract non-ground term algebra, poset. *Let $\mathcal{D} = (\tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}), \leq)$ be the standard domain of (equivalence classes of) terms, ordered by the standard partial order $\leq$ induced by the preorder on terms given by the relation of being "more general".*

Then the domain of abstract terms $\mathcal{D}^\alpha$ is equal to $\mathcal{D}$.

We define the abstraction (t^α) of a term t as: $t^\alpha = \alpha(t)$. Our framework is parametric w.r.t. the abstraction function α, which can be used to tune the accuracy of the approximation.

For example, depending on the Web specification, an abstraction function for digital libraries could be required to discriminate text that appears immediately below the tags that introduce books, or journals, while information appearing at deeper nesting levels within the page probably would not be distinguished. For on–line auctioning, a convenient abstraction function would be better defined as to distinguish registered bidders and sellers, and auctioned items.

Definition 4.2 (term abstraction α) *Let $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$ be a text abstraction function.*

$$\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$$
$$\alpha(t) = \hat{\alpha}(\epsilon, t)$$

where the auxiliary function $\hat{\alpha}$ is given by

$$\hat{\alpha} :: \mathcal{T}ag^* \times \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$$
$$\hat{\alpha}(\_, x) = x, \text{ if } x \in \mathcal{V}$$
$$\hat{\alpha}(c, f(t_1, \ldots, t_n)) = f(\hat{\alpha}(c.f, t_1), .., \hat{\alpha}(c.f, t_n)), \text{ if } f \in \mathcal{T}ag$$
$$\hat{\alpha}(c, w) = atext(c, w), \text{ if } w \in \mathcal{T}ext$$

Particularly, the reader may notice that elements of $\mathcal{T}ext$ are abstracted by taking into account the chain of *tags* under which a particular piece of text appears. This is formalized by means of the text abstraction function

$$atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$$

which is left undefined and is actually the formal parameter of the definition.

The text abstraction function $atext$ should be conveniently fixed in order to tune the abstraction for each particular domain. For instance, in the case where no *tag* distinction is needed, each element in *text* could be simply replaced by some abstract fresh, constant symbol d.

Example 4.3 *Consider again the Web page p_2 of Example 2.3. By fixing*

$$atext(\_, w) = first(w), \text{ where } first(x.xs) = x,$$

the resulting abstraction of p_2 is

$$\alpha(p_2) = list\text{-}categories(\\ pack(category(c)),\\ unit(category(c),\\ category(c)))$$

Note that this abstraction function produces indistinctness among leaves of a term that influence the properties to be verified. As a consequence of this lack of precision, by fixing this abstration we would not be able to detect the error e_2 of Example 3.8 anymore.

In order to achieve correctness of the abstraction, we restrict our interest to text abstraction functions $atext$ which distinguish those pieces of text that are observed by the Web specification rules and then potentially affect the result of the verification.

The following auxiliary functions allow us to know whether a sequence of tags is recognized within some rule of the Web specification. This allows us to determine whether the term occuring at the corresponding position in the Web page needs to be carefully considered. Let $\mathcal{J}$ be a Web specification and $s \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$. Let $c \in \mathcal{T}ag^*$ and $t \in \mathcal{T}ext$. The function $text\_term(c, t)$ returns the term that is obtained by composing as a term the tags of the sequence c together with the piece of text t. For instance, $text\_term(f.g, a) = f(g(a))$. We define the boolean function $gen\_emb_{\mathcal{J}}(s)$ that returns $True$ if there is a (sub)term t occuring in the left–hand or rigth–hand side of a rule in $\mathcal{J}$ such that some instance t' of t embeds s, in symbols $s \trianglelefteq t'$; otherwise it returns $False$. For instance, for the Web specification $\mathcal{J} = \{f(h(Y), g(X)) \rightharpoonup m(X)\}$ and the term $f(g(a))$, $gen\_emb_{\mathcal{J}}(f(g(a))) = True$.

When no confusion can arise, we just write $gen\_emb\_tt_{\mathcal{J}}(c, s)$ by $gen\_emb_{\mathcal{J}}(text\_term(c, s))$.

Now text abstraction functions are required to obey the following correctness condition w.r.t. W.

Definition 4.4 (correctness condition w.r.t. W) *Let W be a Web site and $\mathcal{J}$ be a Web specification. Let $s, t \in \mathcal{T}ext$ be any two pieces of text in W. For every $c \in \mathcal{T}ag^*$ such that $gen\_emb\_tt_{\mathcal{J}}(c, s) \equiv True$ and $gen\_emb\_tt_{\mathcal{J}}(c, t) \equiv True$, the text abstraction function $atext$ satisfies*

$$s \not\equiv t \Rightarrow\ atext(c, s) \not\equiv atext(c, t)$$

The condition above formalizes the idea that, whenever two pieces of text are indistinguishable in the abstract domain, then they are also indistinguishable in the concrete domain.

4.1 Abstract Web specification

The abstraction of the correctness and completeness Web specification rules is simply based on abstracting the terms occurring in the left-hand and right-hand sides of the rules. In particular, the conditional parts of the correctness rules are not abstracted, and hence we let concrete conditions to be applied to concrete as well as abstract data.

Definition 4.5 (abstract specification rule) *Let*

$$\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$$

be a term abstraction function with text abstraction function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$.

Let $rl_M \equiv l \rightharpoonup r\langle q\rangle$ *be a completeness rule and* $rl_N \equiv l \rightharpoonup error|C$ *be a correctness rule. We denote by* rl_M^α *(resp.* rl_N^α*) the abstraction of* rl_M *(resp.* rl_N*), where* $rl_M^\alpha \equiv \alpha(l) \rightharpoonup \alpha(r)\langle q\rangle$ *(resp.* $rl_N^\alpha \equiv \alpha(l) \rightharpoonup error|C$*).*

Example 4.6 *Consider the completeness rule* r_6 *of Example 3.1. By fixing the text abstraction function* $atext(c, x) = last(x)$ *where* $last(w)$ *returns the last element of the sequence* w*, the computed abstract completeness rule* $\alpha(r_6)$ *is* $list\text{-}item(item(\ id(X), state(d)) \rightharpoonup \sharp closed\text{-}auction(item(X))\langle E\rangle$.

When no confusion can arise, we just write $rl_M^\alpha \equiv l^\alpha \rightharpoonup r^\alpha\langle q\rangle$ (resp. $rl_N^\alpha \equiv l^\alpha \rightharpoonup error|C$). The Web specification (I_N, I_M, R) is lifted to $(I_N^\alpha, I_M^\alpha, R)$ element–wise.

To ensure the soundness of the abstract framework, we need to precisely relate the satisfiability of the conditions over abstract and concrete data. Specifically, we require that the fulfillment of an abstract condition implies the fulfillment of the corresponding concrete description.

Definition 4.7 (correctness condition w.r.t. I_N**)** *Let* $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ *be a term abstraction function with text abstraction function* $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$. *Let* $rl \equiv l \rightharpoonup error|C$ *be a correctness rule. Then,* α *is* correct *w.r.t.* rl *iff for each substitution* $\sigma \equiv \{X_1/t_1, \ldots, X_n/t_n\}$

$$C\sigma^\alpha \textit{ holds} \quad \Rightarrow \quad C\sigma \textit{ holds}$$

where $\sigma^\alpha \equiv \{X_1/\alpha(t_1), \ldots, X_n/\alpha(t_n)\}$.

Moreover, α *is* correct *w.r.t.* I_N *if it is correct w.r.t. every correctness rule of* I_N.

4.2 Abstract Web site

When navigating a Web site, it is common to find a number of pages that have a similar structure but different contents. This happens very often when pages are dynamically generated by some script which extracts contents from a database (e.g. in Amazon's Web site). This can make our simple analysis impracticable unless we are able to provide a mechanism to drastically reduce the Web size. In order to ensure that the verified properties are not affected by the abstraction, in this section we develop an abstract methodology which derives an approximation of web sites from the considered Web specifications.

Let us introduce a compression function for terms which reduces the size of each singular Web page by dropping some arguments from the term that represents the page, possibly reducing the number of branches of the tree. We use this function as a preprocess prior to the abstraction of a given Web site.

4.2.1 Web Compression pre–processing

Let (I_N, I_M, R) be a Web specification and $s, t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$. We define three auxiliary functions $root$, $join$, and max_ar. We denote by $root(s)$ the function symbol at the top of s, in symbols $root(f(s_1, \ldots, s_n)) = f$. The function $join(s, t)$ returns the term that is obtained by concatenating the arguments of terms s and t (if they exist), whenever $root(s) = root(t)$. For instance, $join(f(a, b, c), f(b, e)) = f(a, b, c, b, e)$. Finally, function $max\_ar(f, I_M)$ returns the maximal arity of f in I_M.

Definition 4.8 (correctness condition w.r.t. I_M**)** *Let* (I_N, I_M, R) *be a Web specification. Then, the term* $f(t_1, ..., t_n) \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ *is compressed by using function* COMPRESS *given in Algorithm 1, which packs together those subterms which are rooted by the same root symbol while ensuring that the arity of* f *after the transformation is not smaller than* $max\_ar(f, I_M)$.

The idea behind Definition 4.8 is as follows. Roughly speaking, all arguments with the same root symbol f ocurring at level i are joined together. Then, compression recursively proceeds to level $(i + 1)$. The condition that the maximal arity of f in I_M must be respected is essential for the correctness of our method, as this condition ensures that a partial rewrite step on an abstract term is always enabled, whenever the corresponding partial rewrite step can be executed in the concrete domain. Let us see an example.

Algorithm 1 Term Compression Transformation.

Input:
Term $t = f(t_1, \ldots, t_n)$
I_M a set of completeness rules
Output:
Term $f(t'_1, \ldots, t'_m)$, with $m \leq n$

function COMPRESS (t)
  if $n = 0$ **then**
    $\leftarrow f$
  else if $max\_ar(f, I_M) < n$ **and**
      $\exists\, i, j$ s.t. $root(t_i) = root(t_j)$ **then**
    $t' \leftarrow join(t_i, t_j)$
    $\leftarrow$ COMPRESS$(f(t_1, \ldots, t_{i-1}, t', t_{i+1}, \ldots, t_{j-1}, t_{j+1}, \ldots, t_n), I_M)$
  else
    $\leftarrow f($COMPRESS$(t_1, I_M), \ldots,$ COMPRESS$(t_n, I_M))$
  end if
end function

Example 4.9 *Consider the Web page p_1 and the completeness rule r_4 of Example 3.1. The left–hand side of rule r_4 is embedded in p_1, in symbols*

$$list\text{-}items(item(incategories(X, Y))) \trianglelefteq p_1$$

If we naïvely compress p_1 without respecting the maximal arity in I_M of function symbol "$incategories$", we would get

p'_1 = *list-items(*
  item(id(ite0), name(racket),
    description(Wilson tennis racket),
    incategories(category(cat1))),
  item(id(ite0), name(shirt),
    description(men's t-shirts),
    incategories(category(cat1,cat2))),
  item(id(ite2), name(shoes),
    description(women's shoes),
    incategories(category(cat0,cat2))))

Unfortunately,

$$list\text{-}items(item(incategories(X, Y))) \not\trianglelefteq p'_1$$

since in p'_1 the arity of the symbol "$incategories$" is lower than in the lhs of r_4.

Now we are ready to formalize our notion of Web site approximation.

4.2.2 Web site abstraction

In order to approximate a Web site, we start from an initial Web page and recursively apply the successor relation ($\rightarrow$), while implementing a simple *depth-first* search (DFS) [15].

Definition 4.10 (Abstract Web Site) *Let W be a Web site, p be an initial page of W, and (I_N, I_M, R) be a Web specification. Then, the abstraction of W is defined by:*

$$\alpha(W) = \text{DFS}(p, \emptyset, I_M)$$

Where function DFS *is given in Algorithm 2.*

Note that, after applying the transformation above, the information in the Web pages as well as the number of pages in the Web site can be significantly reduced.

Algorithm 2 Web site abstraction.

Input:
$p :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$
$W^\alpha :: set(\tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}))$
I_M a set of completeness rules
Output:
W^α = set $(\tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}))$

function DFS (p, W^α)
  $p^\alpha \leftarrow$ COMPRESS$(\alpha(p), I_M)$
  $W^\alpha \leftarrow W^\alpha \cup \{p^\alpha\}$
  for all i s.t. $(p, p_i) \in \rightarrow_p$ **and**
      COMPRESS$(\alpha(p_i), I_M) \notin W^\alpha$ **do**
    $W^\alpha \leftarrow$ DFS$(p_i,\ W^\alpha)$
  end for
  $\leftarrow W^\alpha$
end function

4.3 Abstract verification soundness

The abstraction function given in Definition 4.2 defines an abstraction by a source-to-source transformation. Due to this source-to-source approximation scheme, all facilities supported by our previous verification system can be straightforwardly adapted and reused with very little effort.

Informally, our abstract verification methodology applies to the considered abstract descriptions of the Web site and Web specification. Given a Web specification (I_N, I_M, R) and a Web site W, we first generate the corresponding abstractions $(I^\alpha_N, I^\alpha_M, R)$ and W^α. Then — since we consider a source to source transformation — we apply our original verification algorithm [4] to analyse W^α w.r.t. $(I^\alpha_N, I^\alpha_M, R)$. We call *abstract* correctness (resp., completeness) error, each correctness (resp., completeness) error which is detected in W^α using $(I^\alpha_N, I^\alpha_M, R)$ by the verification methodology.

In order to guarantee the soundness of the abstract diagnosis, we have to ensure that, when fed with the abstracted data, the partial rewriting relation, $\rightharpoonup$, correctly

approximates the behavior of the partial rewriting relation over the corresponding concrete representation. In the following, we present some results which state the soundness of our abstract representation. First of all we introduce the notion of *abstract embedding*, which is used to establish a relation between concrete and abstract terms.

Definition 4.11 (abstract embedding) *The* abstract embedding *relation*

$$\trianglelefteq^\sharp \subseteq \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \times \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$$

w.r.t. a function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \to \mathcal{T}ext$ *on Web page templates is the least relation satisfying the rules:*

1. $X \trianglelefteq^\sharp t$, *for all* $X \in \mathcal{V}$ *and* $t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$.
2. $f(t_1, \ldots, t_m) \trianglelefteq^\sharp g(s_1, \ldots, s_n)$ *iff* $f \equiv g$ *and* $t_i \trianglelefteq^\sharp s_{\pi(i)}$, *for* $i = 1, \ldots, m$, *and some total function* $\pi :: \{1, \ldots, m\} \to \{1, \ldots, n\}$.
3. $c \trianglelefteq^\sharp c'$ *iff* $c' \equiv atext(x, c)$ *for some* $x \in \mathcal{T}ag^*$.

Given $t_1, t_2 \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ such that $t_1 \trianglelefteq^\sharp t_2$ w.r.t. $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \to \mathcal{T}ext$, we say that t_2 safely approximates t_1.

Basically, Definition 4.11 slightly modifies Definition 3.2 by allowing the detection of noninjective embeddings and the renaming of some constants. In other words, if $t_1 \trianglelefteq^\sharp t_2$, two distinct paths in t_1 may be mimicked (i.e. simulated) by a single path appearing in t_2 modulo (a possible) renaming of some leaves of t_1.

Example 4.12 *Consider the terms* $t_1 \equiv f(g(a), g(b))$ *and* $t_2 \equiv f(g(d, e))$ *which are depicted in Figure 3. Then,* $t_1 \not\trianglelefteq t_2$ *and* $t_2 \not\trianglelefteq t_1$, *but we have* $t_1 \trianglelefteq^\sharp t_2$ *w.r.t. the function* $\{(tag, a) \mapsto c, (tag', b) \mapsto d \mid tag, tag' \in \mathcal{T}ag^*\}$ *as shown in Figure 3 by means of dashed arrows. Moreover, note that the two distinct* t_1*'s edges from* f *to* g *are represented in* t_2 *by a single edge from* f *to* g.

By using Definition 4.11, we are able to represent and map any concrete path of a concrete term into a path of an abstract term. Somehow the structure and the labeling of a concrete term t may be represented in a compressed and suitable relabeled version of t in which many paths of t can be mapped to one shared path of the abstract description of t as stated by the following proposition.

Proposition 4.13 *Let* $t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$. *Let* $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \to \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ *be a term abstraction function whose text abstraction function is* $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \to \mathcal{T}ext$. *Then,* $t \trianglelefteq^\sharp \alpha(t)$ *w.r.t.* $atext$.

PROOF. **(sketch)** We proceed by induction on the structure of t.

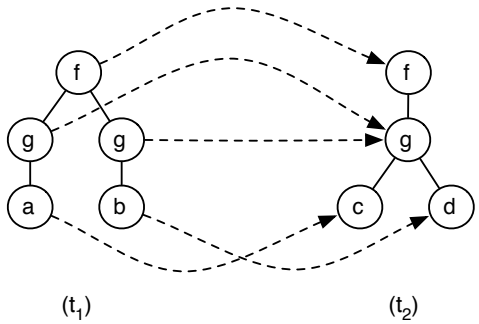

Figure 3. Abstract embedding

Case $t \equiv X, X \in \mathcal{V}$**.** By Definition 4.11, $X \trianglelefteq^\sharp t'$ for any $t' \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ w.r.t. $atext$. Hence $X \trianglelefteq^\sharp \alpha(X)$ w.r.t. $atext$.

Case $t \equiv w, w \in \mathcal{T}ext$**.** By Definition 4.2, $atext(seq, w) \equiv c$, for some $seq \in \mathcal{T}ag^*$. Hence, $w \trianglelefteq^\sharp \alpha(w)$ w.r.t. $atext$.

Case $t \equiv f(t_1, \ldots, t_n), n > 0$**.** By Definition 4.2, we have

$$root(\alpha(t)) \equiv root(t).$$

Hence, $\alpha(t) \equiv f(t'_1, \ldots, t'_m)$. Observe that, whenever $m < n$, some root symbols of terms $t_1, \ldots, t_n$ have been compressed with the aim of reducing the arity of f. Moreover, note that compression preserves the paths of t. That is, if there exists a path from x to y in t, then there exists a corresponding path in $\alpha(t)$. Thus, each t'_j of $\alpha(t)$ may correspond to many t_i's which are included in t. Therefore, we can define the total (possibly, non-injective) function $\pi :: \{1, \ldots, n\} \to \{1, \ldots, m\}$ in the following way: $\pi(i) = j$, where $root(t_i) \equiv root(t_j)$ and t'_j in $f(t'_1, \ldots, t'_m)$ corresponds to t_i in $f(t_1, \ldots, t_n)$. By using such a definition of π and the inductive hypothesis, we get $t_i \trianglelefteq^\sharp t'_{\pi(i)}$ w.r.t $atext$. Consequently, $t \trianglelefteq^\sharp \alpha(t)$ w.r.t. $atext$.

□

Roughly speaking, Proposition 4.13 says that $\alpha(t)$ safely approximates the (concrete) term t.

Example 4.14 *Consider again the terms* t_1 *and* t_2 *of Figure 3. Let* $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \to \mathcal{T}ext$ *be defined as* $\{(f.g, a) \mapsto c, (f.g, b) \mapsto d\}$. *Assume that there exists a Web specification in which the maximal arity of* g *equals to* 1, *so that the compression of the* t_1*'s nodes labeled with* g *is enabled. Then,* $\alpha(t_1) \equiv t_2$ *and* $t_2 \trianglelefteq^\sharp \alpha(t_1)$ *w.r.t. function* $atext$.

Thus, Propositon 4.13 states that abstract terms simulate the concrete terms through an abstract embedding (i.e., the $\trianglelefteq^\sharp$ relation w.r.t. a suitable renaming function). In the following, we demostrate that such a property is preserved by partial rewriting. In other words, whenever a partial rewrite step $t_1 \rightharpoonup t_2$ is executed in the concrete domain, a partial rewrite step over the abstract counterpart is enabled, in symbols $\alpha(t_1) \rightharpoonup t'_2$, such that t_2 still simulates the obtained abstract term t'_2 w.r.t. $\trianglelefteq^\sharp$. The following property holds.

Proposition 4.15 *Let $s, t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$. Let $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ be a term abstraction function with text abstraction function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$. Let $\mathcal{J} \equiv (I_N, I_M, R)$ be a Web specification, and $\mathcal{J}^\alpha \equiv (I^\alpha_N, I^\alpha_M, R)$ be the abstract version of $\mathcal{J}$. If $s \rightharpoonup_{I_M} t$, then*

- $\alpha(s) \rightharpoonup_{I^\alpha_M} t'$
- $t \trianglelefteq^\sharp t'$ *w.r.t.* $atext$

PROOF. **(sketch)** Consider $s, t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ such that $s \rightharpoonup_{I_M} t$ via the rule $l \rightharpoonup r \langle q \rangle \in I_M$, $q \in \{\mathtt{A}, \mathtt{E}\}$. Therefore, $t \equiv r\sigma$ for some substitution $\sigma = \{X_1/s_1, \ldots, X_n/s_n\}$. Now, observe the following two facts.

(i) Given the abstract rule $\alpha(l) \rightharpoonup \alpha(r) \langle q \rangle \in I^\alpha_M$, by applying Definition 4.5, we have $\alpha(l) \equiv l$ (resp., $\alpha(r) \equiv r$) modulo renaming of some constants in l (resp. r) via $atext$. In particular, $l \trianglelefteq^\sharp \alpha(l)$ w.r.t. $atext$ (resp., $r \trianglelefteq^\sharp \alpha(r)$ w.r.t. $atext$).

(ii) Given a term $t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$, the arity of each symbol f appearing in $\alpha(t)$ is reduced as far as it overcomes the maximal arity of f appearing in I_M (see Definition 4.8).

Fact (i) and Fact (ii) imply that if $l \trianglelefteq s_{|w}$, then $\alpha(l) \trianglelefteq \alpha(s)_{|w'}$, $w \in O_{\mathcal{T}ag}(s)$, $w' \in O_{\mathcal{T}ag}(\alpha(s))$. Therefore, $\alpha(s) \rightharpoonup_{I^\alpha_M} t'$ via $\alpha(l) \rightharpoonup \alpha(r) \langle q \rangle \in I^\alpha_M$, and $t' \equiv \alpha(r)\sigma^\alpha$, where $\sigma^\alpha = \{X_1/\alpha(s_1), \ldots, X_2/\alpha(s_n)\}$. Now, by Proposition 4.13, $s \trianglelefteq^\sharp \alpha(s)$ w.r.t. $atext$, and consequently $s_i \trianglelefteq^\sharp \alpha(s_i)$ w.r.t. $atext$, $i = 1, \ldots, n$. Moreover by Fact (ii), we have $r \trianglelefteq^\sharp \alpha(r)$ w.r.t. $atext$. Hence, $r\sigma \trianglelefteq^\sharp \alpha(r)\sigma^\alpha$ w.r.t. $atext$. Finally,

$$t \equiv r\sigma \trianglelefteq^\sharp \alpha(r)\sigma^\alpha \equiv t' \text{ w.r.t. } atext$$

. □

Proposition 4.15 can be generalized to partial rewrite sequences using a simple inductive argument. More formally,

Proposition 4.16 *Let $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ be a term abstraction function with text abstraction function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$. Let $\mathcal{J} \equiv (I_N, I_M, R)$ be a Web specification, and $\mathcal{J}^\alpha \equiv (I^\alpha_N, I^\alpha_M, R)$ be the abstract version of $\mathcal{J}$.*

If $t_0 \rightharpoonup_{I_M} t_1 \rightharpoonup_{I_M} \ldots \rightharpoonup_{I_M} t_n$, $n \geq 0$, then

1. $\alpha(t_0) \rightharpoonup_{I^\alpha_M} t'_1 \rightharpoonup_{I^\alpha_M} \ldots \rightharpoonup_{I^\alpha_M} t'_n$;
2. $t_n \trianglelefteq^\sharp t'_n$ *w.r.t.* $atext$

PROOF. We proceed by induction on the length n of the concrete partial rewrite sequence.

Case $n = 0$**.** In this case, we trivially have $t_n \equiv t_0$, hence $t'_n \equiv \alpha(t_n)$ which directly implies claim 1. To prove claim 2, observe that, by Proposition 4.13, $t_n \trianglelefteq^\sharp \alpha(t_n) \equiv t'_n$ w.r.t. $atext$.

Case $n > 0$**.** We consider the following concrete partial rewrite sequence

$$t_0 \rightharpoonup_{I_M} t_1 \rightharpoonup_{I_M} \ldots \rightharpoonup_{I_M} t_n, n > 0.$$

By inductive hypothesis, there exists $\alpha(t_0) \rightharpoonup_{I^\alpha_M} t'_1 \rightharpoonup_{I^\alpha_M} \ldots \rightharpoonup_{I^\alpha_M} t'_{n-1}$ such that $t_{n-1} \trianglelefteq^\sharp t'_{n-1}$ w.r.t. $atext$. Since $t_{n-1} \trianglelefteq^\sharp t'_{n-1}$, $t'_{n-1} \equiv \alpha(t_{n-1})$. By Proposition 4.15, we obtain $t'_{n-1} \equiv \alpha(t_{n-1}) \rightharpoonup_{I^\alpha_M} t'_n$ such that $t_n \trianglelefteq^\sharp t'_n$. Therefore, by composing the computed abstract partial rewrite sequences, we obtain

$$\alpha(t_0) \rightharpoonup_{I^\alpha_M} t'_1 \rightharpoonup_{I^\alpha_M} \ldots \rightharpoonup_{I^\alpha_M} t'_n$$

with $t_n \trianglelefteq^\sharp t'_n$ w.r.t. $atext$.

□

Given an (abstract) partial rewrite sequence $\mathcal{S}^\alpha \equiv \alpha(t_1) \rightharpoonup_{I^\alpha_M} t'_2 \rightharpoonup_{I^\alpha_M} \ldots \rightharpoonup_{I^\alpha_M} t'_n$, we call abstract completeness requirement any term appearing in $\mathcal{S}^\alpha$. Now, our verification methodology works as follows: first, we compute the concrete completeness requirements and, then, we check whether such requirements are fulfilled in the considered Web site. As explained in Section 3.1, a completeness requirement r is a term which occurs in a partial rewrite sequence of the form $p \equiv t_0 \rightharpoonup t_1 \rightharpoonup t_2 \ldots \rightharpoonup t_n \equiv r$, where $p \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ is a Web page of the Web site W and $r \in \tau(\mathcal{T}ext \cup \mathcal{T}ag)$ is the computed (completeness) requirement. Therefore, by Proposition 4.16, we can directly conclude that each concrete requirement is safely approximated by its abstract description. More formally, the following corollary holds.

Corollary 4.1 *Let $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ be a term abstraction function with text abstraction function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$. Let W be a Web site, and W^α be the abstract version of W. Let (I_N, I_M, R) be a Web specification, and $(I_N^\alpha, I_M^\alpha, R)$ be the abstract version of (I_N, I_M, R).*

If r is a completeness requirement for W computed by (I_N, I_M, R), then there exists an abstract completeness requirement r^α for W^α computed by $(I_N^\alpha, I_M^\alpha, R)$ such that $r \trianglelefteq^\sharp r^\alpha$ w.r.t. $atext$.

The fact that any concrete completeness requirement is safely approximated by an abstract completeness requirement ensures that the abstract verification is safe, that is, whenever an abstract requirement is fulfilled in the abstract Web site, each concrete representation is fulfilled in the concrete domain. This allows us to conclude the absence of concrete errors in the case when no abstract errors are detected.

Formally, the following theorem holds.

Theorem 4.1 *Let $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ be a term abstraction function with text abstraction function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$. Let W be a Web site and W^α be the abstract version of W. Let $\mathcal{J} \equiv (I_N, I_M, R)$ be a Web specification, and $\mathcal{J}^\alpha \equiv (I_N^\alpha, I_M^\alpha, R)$ be the abstract version of $\mathcal{J}$. Then,*

W^α does not contain any abstract (universal/existential) completeness error w.r.t. $\mathcal{J}^\alpha$, then W does not contain any concrete (universal/existential) completeness error w.r.t. $\mathcal{J}$.

PROOF. **(sketch)** First of all, we show that if $\alpha(t) \trianglelefteq \alpha(p)$, $t \in \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$, $p \in W$, then $t \trianglelefteq p$ (*requirement safeness property*). By contradiction, we assume that $t \not\trianglelefteq p$. This amounts to saying that there is a path in t which is not recognized in p. On the other hand, by Proposition 4.13, we have $t \trianglelefteq^\sharp \alpha(t)$ w.r.t. $atext$, and hence all the paths in t are simulated (recognized) in $\alpha(t)$. Since $\alpha(t) \trianglelefteq \alpha(p)$, all the paths in t are recognized in $\alpha(p)$. And finally, by Proposition 4.13, $p \trianglelefteq^\sharp \alpha(p)$ w.r.t. $atext$, all the paths in t are recognized in p, which leads to a contradiction.

Now, we use this result to prove the theorem. In the following, we will distinguish three cases according to the kind of completeness errors we want to detect.

Existential completeness error. By contradiction, we assume there exists a concrete existential completeness error

$$(r, \{p_1, p_2, \ldots, p_n\}, \mathtt{E})$$

in W w.r.t. $\mathcal{J}$. That is, there exists $p \in W$ such that $p \rightarrowtail^+_{I_M} r$ and $\{p_1, p_2, \ldots, p_n\}$ is not existentially complete w.r.t. r (i.e. $\forall\ i = 1, \ldots, n, w \in O_{\mathcal{T}ag}(p_i),\ r \not\trianglelefteq p_{i|w}$). By Corollary 4.1, $r \trianglelefteq^\sharp \alpha(r)$ w.r.t. $atext$. On the other hand, W^α does not contain any abstract existential completeness error w.r.t. $\mathcal{J}^\alpha$. This implies that any computed abstract requirement r^α is embedded in some $p^\alpha \in W^\alpha$. In particular, the abstract requirement $\alpha(r)$ is embedded in some $\alpha(p_i) \in \{\alpha(p_1), \ldots, \alpha(p_n)\} \subseteq W^\alpha$ (in symbols, $\exists\, i = 1, \ldots, n, w \in O_{\mathcal{T}ag}(\alpha(p_i)),\ \alpha(r) \trianglelefteq \alpha(p)_{i|w}$). By the requirement safeness property, we then derive $r \trianglelefteq p_{i|w'}$, which leads to a contradiction, as we supposed that $\{p_1, p_2, \ldots, p_n\}$ is not existentially complete w.r.t. r.

Missing Web page error. Analogous to the first case.

Universal completeness error. Analogous to the first case.

□

Note that —whenever we detect an abstract completeness error— we are not able to guarantee the presence of a concrete completeness error. This is mainly due to the fact that the abstraction can enable partial rewriting steps over abstract descriptions which are not possible in the concrete domain. Thus, there might be an abstract requirement which does not correspond to any concrete requirement, as illustrated by the following example.

Example 4.17 *Consider the following set I_M of completeness rules of a Web specification*

$$r_1)\ f(X, Y) \rightharpoonup m\ \langle \mathtt{E} \rangle$$
$$r_2)\ f(a, b) \rightharpoonup m'\ \langle \mathtt{E} \rangle$$

and the Web site $W = \{f(f(a), f(b), h(c)), m\}$. Assume that the text abstract function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$ is defined as $atext(\_, t) = t$ for each $t \in \mathcal{T}ext$. Then,

$$W^\alpha = \{f(f(a, b), h(c)), m\}.$$

Moreover, the abstract description of I_M is equal to I_M (i.e. $I_M \equiv I_M^\alpha$). The set of concrete requirements of W w.r.t. I_M is $\{m\}$; while the set of abstract requirements of W^α w.r.t. I_M^α is $\{m, m'\}$. Note that m' is not fulfilled in W^α, as it is not embedded in any abstract page of W^α, that is, m' represents an abstract completeness error. On the other hand, requirement m' cannot be computed in the concrete domain, since rule r_2 cannot be applied to Web pages in W. Consequently, m' is not responsible for any concrete completeness error in W.

The approximation we considered allows us to establish a safe connection between abstract and concrete correctness errors as well. In particular, we are able to ensure that whenever an abstract correctness error is detected, a corresponding correctness error must exist in the concrete counterpart.

Given a concrete correctness error $e \equiv (p, w, l, \sigma)$, we define $\alpha(e) \equiv (\alpha(p), w^\alpha, \alpha(l), \alpha(\sigma))$, $w^\alpha \in O_{\mathcal{T}ag(\alpha(p))}$.

Theorem 4.2 *Let $\alpha :: \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V}) \rightarrow \tau(\mathcal{T}ext \cup \mathcal{T}ag, \mathcal{V})$ be a term abstraction function with text abstraction function $atext :: \mathcal{T}ag^* \times \mathcal{T}ext \rightarrow \mathcal{T}ext$. Let W be a Web site and W^α be the abstract version of W. Let $\mathcal{J} \equiv (I_N, I_M, R)$ be a Web specification such that α is correct w.r.t. I_N, and $\mathcal{J}^\alpha \equiv (I_N^\alpha, I_M^\alpha, R)$ be the abstract version of $\mathcal{J}$.*

If W^α contains an abstract correctness error

$$e^\alpha \equiv (p^\alpha, w^\alpha, l^\alpha, \sigma^\alpha)$$

w.r.t. $\mathcal{J}^\alpha$, then W contains a concrete correctness error $e \equiv (p, w, l, \sigma)$ w.r.t. $\mathcal{J}$ such that $\alpha(e) \equiv e^\alpha$.

PROOF. **(sketch)** By contradiction, we assume there exists no concrete correctness error in the Web site W such that $e^\alpha \equiv \alpha(e)$.
The fact that there exists an abstract correctness error

$$(p^\alpha, w^\alpha, l^\alpha, \sigma^\alpha)$$

in an abstract Web page p^α w.r.t. the abstract rule $rl^\alpha \equiv l^\alpha \rightharpoonup error | C \in I_N^\alpha$ implies that $l^\alpha \trianglelefteq p^\alpha_{|w^\alpha}$, for some $w^\alpha \in O_{\mathcal{T}ag}(p^\alpha)$, and $C\sigma^\alpha$ holds.
Now, by Proposition 4.13, we derive (i) $p \trianglelefteq^\sharp p^\alpha \equiv \alpha(p)$ w.r.t. $atext$, and (ii) $l \trianglelefteq^\sharp l^\alpha \equiv \alpha(l)$ w.r.t. $atext$. By (i), (ii), and $l^\alpha \trianglelefteq p^\alpha_{|w^\alpha}$, we can conclude that $l \trianglelefteq p_{|w}$, $w \in O_{\mathcal{T}ag}(p)$. Besides, α is correct w.r.t. I_N , and $C\sigma^\alpha$ holds. Thus, $C\sigma$ holds.
Summing up, there exists $l \rightharpoonup error | \overline{C}$ which detects a concrete correctness error (p, w, l, σ) w.r.t. $\mathcal{J}$, which contradicts the initial hypothesis. □

To conclude, by the approximation scheme formalized so far, we are able to apply the original verification framework to abstract data, providing an extremely efficient analysis which is able to locate correctness errors as well as to ensure the absence of completeness errors in the concrete descriptions quickly, saving time to the user.

In order to improve the efficiency of the analisys, the following optimization has been developed: whenever an abstract completeness error is found, the abstract verification process is stopped. Unfortunately, if we want to check exactly the location of the errors, the concrete methodology should be executed from scratch. Nevertheless, if we divide a Web site into modules we can perform the analysis modularly and execute the (heavier) concrete methodology only on those modules where the abstract analysis detects potential mistakes.

5 Implementation

An experimental implementation αVerdi of the abstract framework proposed in this paper has been developed and compared to the previous Verdi implementation for the realistic test cases given in [6], which are randomly generated by using the XML documents generator xmlgen (available within the XMark project [13]). The tool xmlgen is able to produce a set of XML data, each of which is intended to challenge a particular primitive of XML processors or storage engines.

Table 1 shows some of the results we obtained for the simulation of two different Web specification rules $WS1$ and $WS2$ for the on–line auction system in five different, randomly generated XML documents. Specifically, we tuned the generator for scaling factors from 0.01 to 0.1 to match an XML document whose size ranges from 1Mb –corresponding to an XML tree of about 31000 nodes– to 10Mb –corresponding to an XML tree of about 302000 nodes–.

Web specification $WS1$ aims at checking the verification power of our tool regarding data correctness, and thus includes only correctness rules. The specification rules of $WS1$ contain complex and demanding constraints which involve conditional rules with a number of membership tests and functions evaluation. The Web specification $WS2$ aims at checking the completeness of the randomly generated XML documents. In this case, some critical completeness rules have been formalized which involve the processing of a significant amount of completeness requirements.

The results shown in Table 1 were obtained on a personal computer equipped with 1Gb of RAM memory, 40Gb hard disk and a Pentium Centrino CPU clocked at 1.75 GHz running Ubuntu Linux 5.10.

For each Web specification $WS1$ and $WS2$, column $Verdi$ shows the runtime of the original WebVerdi-M tool. Column App shows the time used for the approximation of the Web site w.r.t. the corresponding abstract Web specification. Finally, column αVerdi shows the execution time of the abstract verification tool αVerdi.

The preliminary results that we have obtained demonstrate a huge speedup w.r.t. our previous methodology. At the same time, the abstraction times are affordable given the complexity and size of the involved data sets: less than 5 minutes for the largest benchmark (10 Mb), with a very reduced space budget. We note that the original

Nodes	Scale factor	Time					
		$WS1$			$WS2$		
		Verdi	App	αVerdi	Verdi	App	αVerdi
30 th	0.01	0.96 s	11 s	0.14 s	165 s	11 s	0.92 s
90 th	0.03	2.84 s	154 s	0.42 s	1768 s	154 s	3.01 s
150 th	0.05	5.94 s	732 s	0.75 s	4712 s	732 s	52.45 s
241 th	0.08	9.42 s	5,330 s	1.23 s	12503 s	5,330 s	186.22 s
301 th	0.10	12.64 s	8,132 s	1.52 s	21208 s	8,132 s	285.51 s

Table 1. Verdi-M **Benchmarks**

WebVerdi-M implementation was only able to process efficiently XML documents whose size was not bigger than 1Mb.

6 Conclusion

Web developing tends to create data that exhibit many commonalities which often result in extremely inefficient Web site verification models and methodologies. This paper describes a novel abstract methodology for Web sites analysis and verification which offsets the high execution costs of analyzing complex Web documents. The framework is formalized as a source-to-source transformation which is parametric w.r.t. the abstraction and translates the Web documents and their specifications into constructions of the very same languages, so that an efficient implementation can be easily derived with very little effort. The key idea for the abstraction is to exploit the sub–structure similarity that is commonly found in HTML/XML documents. Note that no automatic abstraction refinement is needed because no relevant parts are lost due to abstraction.

In this work we have developed the idea of an "horizontal compression". We are currently working on reducing terms also "vertically" by encoding tree paths into single nodes [25]. This is not straightforward if we want to achieve it by a source-to-source transformation, since the abstract embedding should be generalized in order to allow matching between explicit tree paths (i.e. sequences of tree nodes) and nodes of the Web page templates which implicitly represent tree paths.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web. From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

[2] Manar H. Alalfi, James R. Cordy, and Thomas R. Dean. A Survey of Analysis Models and Methods in Website Verification and Testing. In *Proc. 7th Int'l Conf. on Web Engineering (ICWE 2007)*, volume 4607 of *LNCS*, pages 306–311. Springer, 2007.

[3] Luca de Alfaro. Model checking the world wide web. In *Proc. 13th Int'l. Conf. Computer Aided Verification (CAV 2001), Paris, France, July 18-22, 2001*, volume 2102 of *LNCS*, pages 337–349, 2001.

[4] M. Alpuente, D. Ballis, and M. Falaschi. Rule-based Verification of Web Sites. *Software Tools for Technology Transfer*, 8:565–585, 2006.

[5] M. Alpuente, D. Ballis, M. Falaschi, and D. Romero. A Semi-automatic Methodology for Reparing Faulty Web Sites. In *Proc. of the 4th IEEE Int'l Conference on Software Engineering and Formal Methods (SEFM'06)*, pages 31–40. IEEE Computer Society Press, 2006.

[6] M. Alpuente, D. Ballis, M. Falaschi P. Ojeda, and D. Romero. A Fast Algebraic Web Verification Service. In *Proc. of First Int'l Conf. on Web Reasoning and Rule Systems (RR 2007)*, volume 4524 of *LNCS*, pages 239–248, 2007.

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

[8] D. Ballis and J. García 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)*, volume 157(2). ENTCS, Elsevier, 2005.

[9] 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*, 29(1):23–46, 2004.

[10] F. Bry and S. Schaffert. 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), volume 2401 of *LNCS*. Springer-Verlag, 2002.

[11] F. Bry and S. Schaffert. The XML Query Language Xcerpt: Design Principles, Examples, and Semantics. Technical report, 2002. Available at: `http://www.xcerpt.org`.

[12] P. Buneman, M. Grohe, and C. Koch. Path Queries on Compressed XML. In *Proceedings of the 29th Int'l Conference on Very Large Data Bases (VLDB'03)*, pages 141–152. Morgan Kaufmann, 2003.

[13] Centrum voor Wiskunde en Informatica. XMark – an XML Benchmark Project, 2001. Available at: `http://monetdb.cwi.nl/xml/`.

[14] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. The maude 2.0 system. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications (RTA 2003)*, number 2706 in LNCS, pages 76–87. Springer-Verlag, 2003.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA, 2nd edition, 2001.

[16] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[17] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282, 1979.

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

[19] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 244–320. Elsevier Science, 1990.

[20] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Verifying Integrity Constraints on Web Sites. In *Proc. of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, volume 2, pages 614–619. Morgan Kaufmann, 1999.

[21] M. F. Fernandez and D. Suciu. Optimizing Regular Path Expressions Using Graph Schemas. In *Proc. of Int'l Conf on Data Engineering (ICDE'98)*, pages 14–23, 1998.

[22] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing Simulations on Finite and Infinite Graphs. In *IEEE Symposium on Foundations of Computer Science*, pages 453–462, 1995.

[23] N. El Kadhi and H. El-Gendy. Advanced Method for Cryptographic Protocol Verification. *Journal of Computational Methods in Science and Engineering*, 6:109–119, 2006.

[24] J.W. Klop. Term Rewriting Systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume I, pages 1–112. Oxford University Press, 1992.

[25] Temur Kutsia. Context sequence matching for xml. In *Proc. of 1st Int'l Workshop on Automated Specification and Verification of Web Sites (WWV'05)*, volume 157(2), pages 103–119. ENTCS, Elsevier, 2005.

[26] T. Legall, B. Jeannet, and T. Jhon. Verification of Communication Protocols Using Abstract Interpretation of FIFO Queues. In *Proceedings of Algebraic Methodology and Software Technology, 11th International Conference (AMAST'06)*, volume 4019 of *LNCS*, pages 263–274. Springer, 2006.

[27] M. Leuschel. Homeomorphic Embedding for Online Termination of Symbolic Methods. In T. Æ. Mogensen, D. A. Schmidt, and I. H. Sudborough, editors, *The Essence of Computation*, volume 2566 of *LNCS*, pages 379–403. Springer, 2002.

[28] S. Lucas. Rewriting-Based Navigation of Web Sites: Looking for Models and Logics. In *Proc. of 1st Int'l Workshop on Automated Specification and Verification of Web Sites (WWV'05)*, volume 157(2). ENTCS, Elsevier, 2005.

[29] B. Michael, F. Juliana, and G. Patrice. Veriweb: automatically testing dynamic web sites. In *Proc. of 11th Int'l WWW Conference*. ENTCS, Elsevier, 2002.

[30] N. Polyzotis, M.N. Garofalakis, and Y. E. Ioannidis. Approximate XML Query Answers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (ICMD'04)*, pages 263–274. ACM, 2004.

[31] TeReSe, editor. *Term Rewriting Systems*. Cambridge University Press, Cambridge, UK, 2003.